

AI baserad mätteknik för Jordbruk

Andreas Borgström, Te4 (Gymnasieingenjör)

Abstrakt	2
Förord	2
Introduktion	2
Metod	3
Object Detection Model	3
Optisk mätalgoritm	4
Optisk uppskattning av djup	5
Resultat	7
Diskussion	10
Sammanfattning	10
Appendix	11

Växthus Nordic Greens Trelleborg AB



Abstrakt

Med ett ständigt växande behov av mat behöver växthus nya metoder för att bli mer effektiva. Den här artikeln beskriver ett projekt med syftet att övervaka tillväxten av tomatplantor i ett växthus med hjälp av AI och andra optiska tekniker. Artikeln beskriver också hur insamlad-mätdata kan användas för att förutse tillväxt och därmed kunna reglera relevanta miljöparametrar.

Nyckelord: AI, Växthus, Regulation

Förord

Jag vill tacka Göran Hasse (Raditex Control AB) och Thomas Wildig (Arvalla AB) för att jag fick vara en del av detta projekt och med det en chans att förbättra mina färdigheter inom programmering, artificiell intelligens och konsultyrket.

Introduktion

Erfarna växthusodlare är väldigt bekanta med plantornas behov och för att kunna öka effektiviteten i produktionen måste denna insikt appliceras på varje planta för att på ett precist sätt mäta hälsa och tillväxt.

Att ofta mäta hur stora plantorna är kan säga hur väl de växer och vilka åtgärder som är nödvändiga ifall de inte växer som de ska. I stora växthus blir detta tröttsamt snabbt om man måste göra det manuellt med ögonmått och mätsticka och sedan skriva ner alla mått. I en del moderna växthus används robotar som rör sig över golvet och kan utföra lättare eller mer komplexa uppgifter automatiskt.

Parametrar som till exempel jordfuktighet och näringsnivåer kan mätas med IoT sensorer, medan tillväxt och detektering av skadedjur och sjukdom kräver visuell avläsning. Något som är enkelt för växthusodlare, men som blir väldigt komplicerat för automatiska system.

Den här artikeln beskriver ett koncept där en kamera används med en robot i ett växthus i Trelleborg och en object detection modell används för att bearbeta bilderna tagna med kameran för att kunna mäta dimensionerna på tomaterna, samt volym och vikt.

Metod

Utvecklingen av mätsystemet med Computer Vision sker i tre processer.

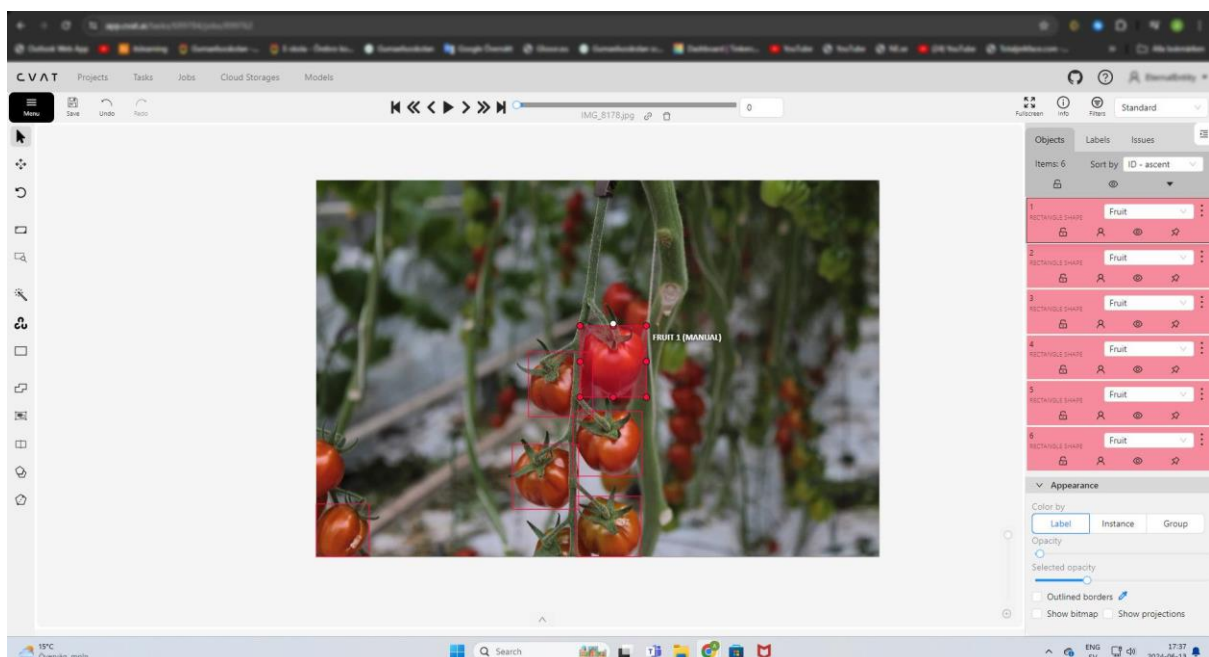
- Träning av en Object Detection modell med relevant plant data såsom bilder på tomater, blad och blommor.
- Mätning och konvertering av objekt från pixlar till verkliga värden.
- Uppskattning av avstånd från kameran till objekten.

Object Detection Model

För att få mått från tomatplanter behöver en object detection modell kunna känna igen object som tomater, blommor och blad i en bild. Då inget sådant dataset gick att hitta online behövdes det utvecklas ett nytt dataset för projektets behov.

Under ett besök användes en DSLR kamera för att ta ungefär 160 bilder med storleken 5184x3456 i .jpg format och med olika brännvidder.

Under två veckor annoterades bilderna med den kostnadsfria webbtjänsten CVAT.ai. Den resulterande modellen i det populära dataset formatet YOLO använde 150 av bilderna.



Figur 1. Annoteringsverktyget CVAT.ai.

Genom att använda en NVIDIA L4 Tensor Core GPU via Google Collaboratory kunde modellen tränas på sju minuter och 44 sekunder under 30 epoker där 8 bilder med storleken 864x864x1 behandlades samtidigt.



Optisk mätalgoritm

1. En måttsticka skulle behöva hållas bredvid varje planta eller en monterad på roboten som rör sig.
2. Perspektivet har betydelse då en mätsticka i förgrunden inte är en bra referens för att uppskatta mått på objekt i bakgrunden.

Med dessa brister i åtanke behöver mätningen utföras endast med bilder.

Under undersökning av olika metoder för att hitta rätt mått hos objekt i bilder så förekom denna formel ofta som en föredragen metod.

Objekt Bredd

$$= \frac{\text{avstånd} \times \text{Pixelbredd} \times \text{Kamerasensor bredd}}{\text{Brännvidd} \times \text{Bildbredd}}$$

För att räkna ut objektets höjd byter man ut alla bredd variabler med höjd variabler.

I detta projekt användes en Canon EOS 1300D dslr kamera med ett 18-55 mm objektiv. Kamerasensorn har dimensionerna 22.3x 14.9 mm och upplösningen var satt till 5184 x 3456 pixlar.

Alla parametrar utan avstånd är enkla att få tag i: Pixel bredden/höjden går att få från bounding boxes returnerade av object detection modellen och brännvidden är tillgänglig genom bildens metadata.

Formeln ovan, med avstånds parametern kan returnera bredd och höjdvärden som kan användas för att uppskatta tomaternas volym och vikt, och dessutom en ungefärlig area från blad och blommor.

Metoden för att returnera avstånd beskrivs i nästa steg.



Optisk uppskattning av djup

Att returnera ett avståndsvärden från objekten i bilden visade sig bli den mest tidskrävande delen i projektet, med många försök på metoder som misslyckades.

En metod som fungerade var utvecklingen av en funktion som beskriver korrelationen mellan den diagonala längden på ett objekt i en bild och avståndet från kameran när den rör sig längre bort.

Utvecklingen av funktionen sker relativt enkelt där man noterar brännvidden kameran använder vid tillfället och sedan tas flera bilder på ett objekt (helst ett rektangulärt) som för varje bild flyttas längre ifrån kameran. Dessa avstånd i varje bild måste då också noteras.

Med ett bildprogram beskars bilderna så att de bara består av objekten, så att diagonalerna enklare kan räknas ut från den nya bildstorleken. Diagonalerna associeras sedan med avstånden från kameran. Nedan finns några av referensbilderna med avståndet till objekten utskrivet.



Figur 3. Referensbilderna används för att skapa en avstånds uppskattningsfunktion.

The diagonals along with their associated distances were input in the code below.

```
x = [1901, 1748, 1645, 1511, 1459, 1388, 1307,
1245, 1198, 1117, 1051]
y = [55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]
coefficients = np.polyfit(x, y, 2)
```

Figur 4. Kodavsnitt där x och y värden skapar koefficienterna till en andragradsekvation.

X-listan innehåller diagonalerna i pixlar, Y-listan innehåller de mätta avstånden från kameran och heltalet 2 i funktionen np.polyfit bestämmer vad för slags koefficienter som ska returneras. I det här fallet blir funktionen en andragradsfunktion.

Funktionen returnerar en array med A, B och C koefficienterna för en andragradsfunktion:

$$f(x) = Ax^2 + Bx + C$$

I det här fallet: A = 0.0000357666869, B = 0.165897917, C = 246.

Med brännvidden 48 mm ser funktionen ut så här:

$$f(x) = 0.0000357666869x^2 - 0.165897917x + 246$$

Nu med avståndsvärdena går det att räkna ut den verkliga storleken på objekten. Dock så finns det en stor svaghet, i form av förändringar i brännvidd. En funktion beräknad från bilder med en brännvidd på 48 mm är inte tillämplig när till exempel bilder med 46 mm brännvidd ska beräknas.

Resultat

Projektet består av 3 python filer, run_main.py (Ska senare ändras till main.py), inference.py, measurements.py.

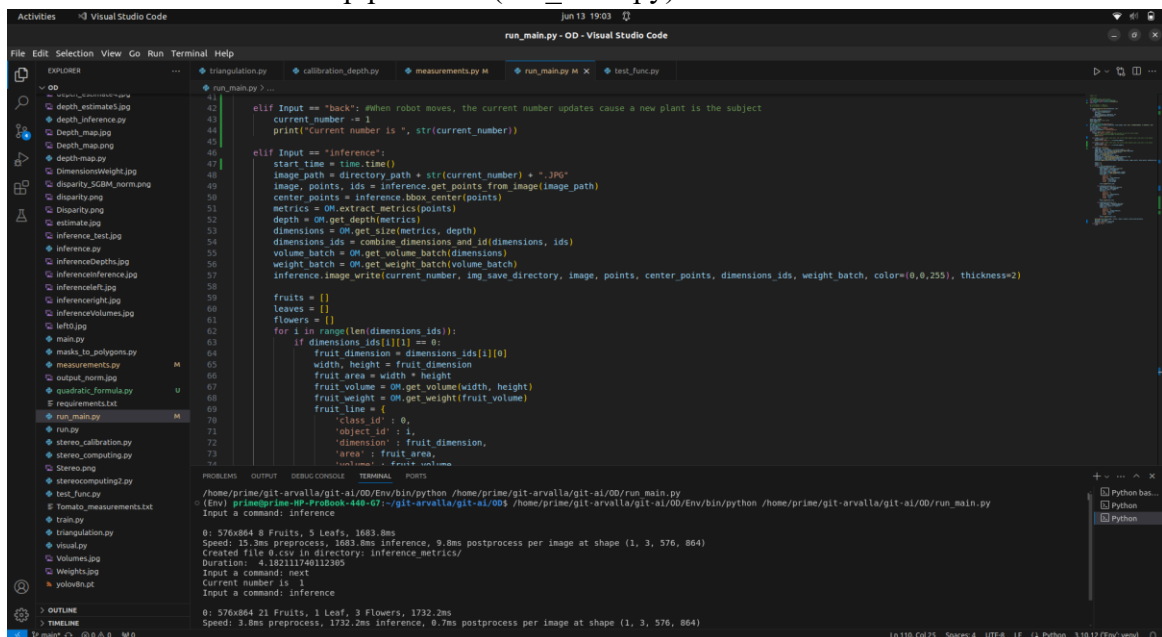
Där:

- run_main.py tar emot användarinmatning i form av kommandona “inference”, “next”, “back” och “kill”. "Inference" matar object detection modellen med en aktiv bild och därefter lagras resultaten från modellen i form av en bild med bounding boxes och en .csv-fil. “Next” ändrar vilken bild i katalogen som ska användas till den följande. “Back” ändrar den aktiva bilden till föregående bild i katalogen. “Kill” stoppar koden.

Med en Raspberry Pi 5 kan koden bli modifierad för att kunna använda en Raspberry Pi Camera Module v3 för att kunna ta bilder. Detta sker förmodligen med ett kommando såsom “capture” eller “shoot”. Den här funktionaliteten kommer ha sin egen python-fil.

- inference.py är en fil innehållande klassen Inference som hanterar bildbehandlingsoperationer och modifieringar med ett större beroende av det populära biblioteket OpenCV.
- measurements.py är en fil innehållande klassen Optical_measurements som hanterar enkla geometri beräkningar.

Nedan finns en skärmdump på koden (run_main.py) när den körs.



```
run_main.py
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
0: 576x864 8 Fruits, 5 Leafs, 1683.8ms
Speed: 15.3ms preprocess, 1683.8ms inference, 9.8ms postprocess per image at shape (1, 3, 576, 864)
Created file 0.csv in directory: inference_metrics/
Duration: 4.18211749112385
Input a command: next
Current number is 1
Input a command: inference
0: 576x864 21 Fruits, 1 Leaf, 3 Flowers, 1732.2ms
Speed: 3.8ms preprocess, 1732.2ms inference, 0.7ms postprocess per image at shape (1, 3, 576, 864)
```

Figur 5. Kommandon körd från run_main.py.

Vid skrivandet har funktionen en relativt smidig pipeline som kan hantera inference och export av bilder och csv filer på ungefär 1.88 sekunder.

Nedan är ett exempel på denna output.



Figur 6. Modellen skapar bilden 0.jpg med inferens värdena.

De gröna numrena är bredd och höjd i millimeter där numret till vänster om rektanglarna är höjd och numret under botten är bredden. Det vita numret i det övre vänstra hörnet är class_id* indexet. Det blåa numret är vikten i gram.

```
1 class_id,object_id,dimension,area,volume,weight
2 0,0,"(64.72, 64.86)",4197.7392,142250.1,153.05
3 0,1,"(37.41, 77.27)",2890.6706999999997,56621.97,60.92
4 0,2,"(72.06, 66.64)",4802.0784,181184.95,194.95
5 0,3,"(65.98, 74.2)",4895.716,169132.53,181.98
6 0,4,"(71.35, 70.11)",5002.348499999999,186881.6,201.08
7 0,5,"(52.6, 41.78)",2197.628,60525.52,65.12
8 0,7,"(40.97, 56.55)",2316.8534999999997,49700.77,53.48
9 0,12,"(42.66, 45.15)",1926.0989999999997,43022.74,46.29
10 1,6,"(63.44, 54.06)",3429.5664,,
11 1,8,"(49.45, 32.52)",1608.1140000000003,,
12 1,9,"(75.38, 39.22)",2956.4035999999996,,
13 1,10,"(72.05, 66.24)",4772.592,,
14 1,11,"(69.09, 53.79)",3716.3511000000003,,
```

CSV ▾ Tab Width: 8 ▾ Ln 14, Col 43 ▾ INS

Figur 7. Mätvärdena från inferens sparade i filen 0.csv.

*class_id säger vilket plant objekt som hittades i bilden. 0 är Tomat, 1 är Blad, 2 är Blomma. 2 dyker inte upp i filen eftersom det inte fanns några blommor i originalbilden. (Alternativt så lyckas inte modellen upptäcka en blomma i ursprungsbilden fast den finns.

*object_id är en numrering för värdena i en bild. IDt är designerat av object detection modellen, men det betyder inte att modellen har förmågan att känna igen och skilja individuella tomater från objekt av samma klass från varann.

Diskussion

Klass	Bilder	Instanser	Precision
Alla	15	390	0.754
Frukt (Tomat)	15	108	0.873
Blad	15	217	0.542
Blommor	8	65	0.847

Figur 8. Object detection modellens prestanda på valideringsbilder.

Tabellen i figur 6 visar prestandan hos object detection modellen när den testats på validering bilderna. I 15 bilder upptäcktes 390 objekt. Precisions Kolumnen visar att 75.4 % av dessa objekt klassificerats korrekt. Mer specifikt klassificerades tomaterna korrekt mest, och blad minst. Detta orsakas troligen av att modellen har svårt att skilja på blad och den gröna bakgrunden. Framtida förbättringar involverar en högre representation av blad som går att skilja från bakgrunden i datasetet. Dessutom skulle ett dataset med mer av samma typ av bilder sannolikt öka precisionen.

När det kommer till uppskattningen av djup så är det tydligt att funktionen inte fungerar optimalt då uppskattningarna generellt blir för stora och inte inom acceptabel marginal. Anledningen tros vara att de flesta av bilderna tagits med brännvidden 46 mm, medan djup algoritmen genom ett misstag utvecklats med bilder som har brännvidden 48 mm. Det här felet går enkelt att åtgärda med nya referensbilder i en matchande brännvidd. Alternativt så kan en ny algoritm utvecklas som skapar en andragsfunktion som tar brännvidden i åtanke för ökad dynamism.

Projektet fungerar bra som ett proof of concept och nästa steg skulle vara att integrera det med en robot i ett växthus och koppla systemet till en databas. En ytterligare algoritm skulle också kunna utvecklas som förutser tillväxt med data över tid.

Sammanfattning

Utvecklingen av mätteknik med AI visade sig framgångsrik, om än med ett par mindre förfiningar att ordna, som att implementera en felhantering i koden, förbättring av object detection modellens precision och att skapa en dynamisk former för avstånd uppskattning med brännvidden i åtanke.

Framåt behöver projektet integreras med en rörlig robot i ett växthus för att kunna utveckla ett praktiskt system.

Appendix

Begrepp använda i artikeln i alfabetisk ordning:

AI, Bounding boxes, Brännvidd, CPU, Computer Vision, Dataset, DSLR kamera, Epoker, GPU, IDE, Modell, Object Detection, OpenCV, RAM, Raspberry Pi 5, Tidsseriedata, Valideringsdata, YOLO

- AI: AI är en förkortning för Artificiell Intelligens, intelligenta datorsystem som kan utföra uppgifter likt människor.
- Bounding boxes: I kontexten av denna artikel refererar bounding boxes till rektanglarna genererade runt objekt detekterade av object detection modellen.

- Brännvidd: Brännvidden är avståndet mellan änden av objektivet och kamerasensorn. Beroende på avståndet kan synvilken vara bredare eller smalare.
- CPU: En Central Processing Unit är en komponent i datorer som skickar ut alla instruktioner för att kunna köra program.
- Computer Vision: Ett område inom AI som fokuserar på datorers förmåga att förstå innehåll i bilder.
- Dataset: Dataset innehåller organiserad data (bilder, text, ljus, osv) och är nödvändiga för att utveckla en AI- modell.
- DSLR Kamera: Digital Single Lens Reflex kameror kombinerar digitala sensorer med en spegel och en bred repertoar av utbytbara objektiv. Dessa kameror tillåter användaren med kontroll över inställningar och bildkvalitet än point and shoot kameror och smartphones.
- Epoker: En epok inom kontexten maskininlärning är en enhet som mäter hur många gånger en träningsprocess har gått igenom all träningsdata.
- GPU: En Graphics Processing Unit är en komponent in datorer gjord för visuella beräkningar. Den utmärker sig i parallella beräkningar vilket leder till snabbare utföranden.
- IDE: En miljö att skriva, testa, och felsöka kod med stöd för olika programmeringsspråk.
- Inferens: Inferens är processen där ny data behandlas av en AI modell för att få en output.
- Modell: En modell är i AI kontext är ett program kapabelt att utföra specifika uppgifter genom bekantning med massor av data.
- Object Detection: Ett område inom Computer Vision där objekt i en bild detekteras med dess position i bilden.
- OpenCV: Open Source Computer Vision Library är ett gratis bibliotek med funktioner för bildhantering.
- RAM: Random Access Memory är ett korttidsminne för datorer som lagrar aktiva processer och data åt CPU:n.
- Raspberry Pi 5: Raspberry Pi 5 (RPI 5) är en enkortsdator som ofta används för DIY projekt där hög beräkningskraft för mindre pengar och mindre strömkonsumtion behövs.
- Tidsseriedata: Data över tid.
- Valideringsdata: Reserverad data i ett dataset till för att testa prestandan hos maskininlärningsmodeller.
- YOLO: En stor object detection modell tränad på tusentals bilder av vanliga objekt.
- Pythagoras sats: En formel för att räkna ut diagonalen i en rät triangel med hjälp av längderna på de kortare sidorna. Formel: $a^2 + b^2 = c^2$