Ansiktsigenkänningsbaserad dörrkontroll för katter

Andreas Lindgren

Examensarbete

Informationsteknik

2023

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	
Författare:	Andreas Lindgren
Arbetets namn:	Ansiktsigenkänningsbaserad dörrkontroll för katter.
Handledare (Arcada):	Jonny Karlsson
Uppdragsgivare:	

Sammandrag:
Examensarbetet behandlar undersökningen av olika metoder för igenkänning av ansikten på enskilda katter. You Only Look Once (YOLO) är enligt utförda forskningar den snabbaste och noggrannaste djupinlärningsmetoden för bildigenkänning och därmed valdes för arbetet. I arbetet beskrivs utvecklingen av ett ansiktsigenkänningssystem som innebär träningsdatasettets och -miljöns förberedelse samt utförandet av träningsprocessen av en anpassad YOLO version 7 igenkänningsmodell för identifiering av två olika katter. I arbetet beskrivs också metoderna för testande av systemets prestanda i form av noggrannhet och hastighet både på en dator i medelklass och på en Raspberry Pi enkortsdator.
Resultaten presenteras i form av en tabell och bilder på utdata av detekteringsskriptet. I slutet av examensarbetet diskuteras hur ansiktsigenkänningssystems prestanda kunde förbättras i framtiden för att kunna användas på en Raspberry Pi dator med bättre snabbhet.

Nyckelord:	YOLO, djupinlärning, Raspberry Pi
Sidantal:	34
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information Technology
Identification number:	
Author:	Andreas Lindgren
Title:	Face recognition-based door control for cats.
Supervisor (Arcada):	Jonny Karlsson
Commissioned by:	

Abstract:
This thesis deals with the investigation of different methods for recognizing faces of individual cats. You Only Look Once (YOLO) is, according to research, the fastest and most accurate deep learning model for image recognition. This work describes the development of a face recognition system which involves the preparation of the training dataset and environment and the execution of the training process of a custom YOLO version 7 recognition model for the identification of two individual cats. This work also describes the methods for testing the performance of the system in terms of accuracy and speed both on a midrange computer and on a Raspberry Pi single board computer.
The results are presented in the form of a table and images of the output data of the detection script. At the end of the thesis, it is discussed how the performance of the recognition system could be improved in the future to be used on a Raspberry Pi computer with improved speed.

Keywords:	YOLO, deep learning, Raspberry Pi
Number of pages:	34
Language:	Swedish
Date of acceptance:	

INNEHÅLL / CONTENTS

Figurer / Figures

Tabeller / Tables

1 INLEDNING

1.1 Bakgrund

I ett katthushåll kan man lätt råka ut för problemet där katterna äter upp varandras mat. Detta kan vara problematiskt speciellt ifall katterna följer en bestämd diet. För att hindra katterna att komma åt varandras skålar kunde man försöka lära dem att endast få äta från sin egen skål. Att träna katter är dock inte alltid lika lätt som till exempel att träna en hundvalp och kan kräva mycket mer tid och tålamod. Fastän man hade en tränad katt, kan man ändå inte vara säker på att den håller sig vid sin egen kopp då ägaren inte är hemma.

En självklar lösning är att ha skålarna gömda då det inte är matdags och ta fram dem några gånger om dagen. Problemet med detta förutom att hamna gömma och ta fram skålarna, är att upprätthålla en konstant övervakning av utrymmet där matskålarna är placerade. För att eliminera behovet av övervakning forskar examensarbetet möjligheten att med hjälp av datorseende och bildigenkänning känna igen och identifiera specifika kattansikten för användning i ett automatiserat system.

1.2 Syfte och målsättningar

Syftet med detta arbete är att undersöka och jämföra olika metoder för ansiktsigenkänning av katter samt att utveckla ett ansiktsigenkänningsbaserat system som kunde implementeras på hårdvara med mindre datorkraft.

Målet med arbetet är att utveckla ett system som med hjälp av datorseende kan identifiera en specifik katt med så bra noggrannhet och hastighet som möjligt. Även då hastigheten spelar en roll i examensarbetet och igenkänningssystemet ska kunna användas på ett mindre system, är arbetets fokus i igenkänningen av kattansikten.

1.3 Metoder

För att nå lösningen för problemet undersöktes och jämfördes olika utvecklade algoritmer som används för igenkänning av kattansikten samt olika system som använder sig av dessa algoritmer. Valet av metoden som användes i arbetet för igenkänning gjordes genom undersökning av relaterad forskning inom området.
Ansiktsigenkänningsmetodens prestanda mättes med att testa både noggrannheten på en uppsättning av testbilder samt hastigheten på en bordsdator och på ett system med mindre datorkraft.

1.4 Struktur

Andra kapitlet av arbetet innehåller teoretiska delen av arbetet där olika utvecklade algoritmer och lösningar för igenkänning av katter undersöks och diskuteras. I tredje kapitlet motiveras och beskrivs valet av igenkänningsmetoden för arbetet, förberedelserna för träning av igenkänningsmodeller samt träningsprocessen. I fjärde kapitlet beskrivs olika metoderna som använts för testning av noggrannheten och hastigheten av modellerna samt testens resultat. I femte kapitlet beskrivs kort en ny variant av igenkänningsalgoritmen som användes i examensarbetet och hur den kunde användas för att förbättra resultaten i framtiden.

2 RELATERAD FORSKNING OCH UTVECKLING

En del forskningar har gjorts och olika system har byggts upp för att känna igen människoansikten, men betydligt färre för att känna igen ansikten på katter, för att inte tala om andra djur.

I det här kapitlet behandlas några av de väsentliga teknikerna samt system och apparater som utvecklats för igenkänning av katter.

2.1 Igenkänning av katter

Forskare har redan specificerat en funktionsvektor som kan användas för att känna igen människans ansikte. Samma tillvägagångssätt gäller dock inte för husdjur eftersom vi inte vet vilka egenskaper vi behöver använda och hur de ska tolkas. När det gäller människor kan forskarna till exempel använda Variational Autoencoder (VAE) arkitektur för att extrahera egenskaper från människoansikten. I denna metod komprimeras foton av en person till vektorer som innehåller de önskade egenskaperna, till exempel hudton och ansiktsuttryck.
När det gäller ansiktsigenkänning av husdjur finns det ännu ingen tillförlitlig funktions vektor. (Nadejda 2022).

Enligt rapporten ”Pet Cat Face Verification and Identification“ av Adam Klein från Stanfords universitet undersökte de hur de kan använda djupinlärning med faltningsnätverk för att upptäcka, verifiera och identifiera individuella ansikten på digitala bilder av tamkatter. De rapporterade om träning av YOLOv3 för att upptäcka kattansikten och EfficientNet för verifiering och identifiering. EfficientNet är en arkitektur för ett faltningsnätverk och en skalningsmetod som på ett enhetligt sätt skalar alla dimensioner av djup, bredd och resolution med hjälp av en sammansatt koefficient. Till skillnad från konventionell praxis, där dessa faktorer skalas godtyckligt, skalar skalningsmetoden EfficientNet på ett enhetligt sätt nätverkets bredd, djup och upplösning med en uppsättning fasta skalningskoefficienter. (Papers With Code 2019).

För träning av neurala nätverket på EfficientNet skapade de ett nytt dataset med ca 130 000 individuella bilder av ca 23 500 olika katter från profiler för adoption av husdjur på nätet. Efter förberedelsen av data slopades 33 536 oanvändbara bilder och nya datasettet delades in i 86 867 bilder på 15 949 olika katter för träning, 4803 bilder på 886 olika katter för validering samt 4794 bilder på 887 olika katter för testande.
De skapade 10 000 par av bilder på kattansikten från testdata för att utföra verifieringsuppgiften. (Klein 2019).

Figur 1. Exempel på kattpar. (Klein 2019).

Den resulterande matrisen visar en noggrannhet på cirka 94,6 % på data som inte innehåller några tidigare observerade kattklasser, se figur 2. (Klein 2019).

	Same	Different
Accept	47.31%	2.46%
Reject	2.93%	47.30%

Figur 2. Matris med falskt positiva (different/accept) och falskt negativa (same/reject) procentandelar. (Klein 2019).

Enligt resultaten av Adam Kleins forskning kan man dra slutsatsen att ansiktsigenkänningsmetoden gynnar goda resultat i noggrannheten på testbilderna. Forskningen tar dock inte ställning till hur deras modell skulle prestera i realtid på nya bilder som kommer från en videoström.

Arkaitz Garro, en ingenjör från Nederländerna byggde upp ett videoövervakningssystem som meddelar honom då hans katt står vid ytterdörren. Garro använde en Raspberry Pi-dator med en kameramodul, tillsammans med motionEyeOS som programvara för rörelsedetektering. (Price 2018). MotionEyeOS är en Linuxdistribution som förvandlar en enkortsdator till ett videoövervakningssystem. (Github, 2017. *MotionEyeOS).* För bild-

igenkänning använde han Amazon Web Services (AWS) molnbaserade programvara: Amazon Rekognition. (Price 2018).

Amazon Rekognition erbjuder olika tjänster beroende på användningsfallet. Amazon Rekognition Image är en tjänst för bildigenkänning som bygger på djupinlärning och som upptäcker objekt, scener och ansikten, extraherar text, känner igen kändisar och identifierar olämpligt innehåll i bilder. Tjänsten gör det också möjligt att söka och jämföra ansikten mot bilder uppladdade till tjänsten. (AWS 2022).

Vid en intervju med Business Insider nämnde Garro att ”När systemet upptäcker en rörelse skickar den en bild till igenkänningsprogrammet, som i sin tur kontrollerar kattens identitet mot tidigare bilder av katten.” (Price 2018). Källkoden för Garros system är inte tillgänglig för allmänheten, vilket gör det svårt att veta precis hur han byggde upp systemet men enligt Amazon kan man med Rekognition Image hitta liknande ansikten i en stor samling bilder. Man kan skapa ett index över ansikten som upptäcks i bilderna. Rekognition Image snabba och exakta sökning ger ansikten som bäst matchar referensansiktet. (AWS 2022).
Fördelen med att använda molnbaserad programvara för bildigenkänning är användarvänligheten. Man klarar sig med minimalt programmeringsarbete då man inte själv behöver välja, programmera och konfigurera teknologin för igenkänningen. En annan fördel med molnbaserad programvara är att all tung beräkning sker i molnet, vilket gör det möjligt att använda hårdvara med mindre datorkraft. Det är dock sällan gratis att använda molnbaserad programvara och kan anses som en nackdel. Dessutom måste systemet vara kopplat till internet då systemet ska skicka bilden av katten till molnet för igenkänning.

2.2 Automatiska produkter för individuell matdosering för djur

År 2014 startade entreprenören Mu-Chi Sung med sitt företag 42ARK en folkfinansierings kampanj för produkten: ”CatFi”.

Figur 3. En bild på 42ARKs produkt: ”CatFi” (Indiegogo 2014, Bistro: A Smart Feeder Recognizes Your Cat's Face)

Då en katt stiger på vågen, aktiveras systemets inbyggda kamera och skickar en videoström på katten till företagets server där algoritmen för igenkänningen av katten körs. Katternas portionsstorlekar är definierade i en databas och identifierade kattens rätta matdos distribueras från matbehållaren. (Indiegogo 2014, Bistro: A Smart Feeder Recognizes Your Cat's Face).

Företaget samlade in över 200 000 euro under kampanjen och fick mycket positiv publicitet. Det har dock inte funnits någon uppdatering om utvecklingen från företaget sedan år 2017 och många kräver sina pengar tillbaka. Ingen information finns om huruvida utvecklingen av CatFi fortfarande pågår eller inte, och varför utvecklingen skulle ha avbrutits.

Under Consumer Electronics Show (CES) år 2019 presenterade italienska företaget Volta produkten ”Mookkie”.

Figur 4. En bild på voltas produkt: ”mookkie”. (Businesswire 2019)

Enligt Voltas verkställande direktör Silvio Revelli lär ska systemet kunna identifiera individuella hundar samt katter. (Businesswire 2019). Apparaten har en kamera på framsidan ungefär vid höjden av husdjurets ansikte. Då djuret närmar sig apparaten, aktiveras kameran och bilderna analyseras av ett neuronnät. Vid en lyckad igenkänning öppnas luckan över skålen och djuret kommer åt maten. (Marchese 2019).

Volta lovade att automatiserade matskålen Mookkie skulle vara tillgänglig för konsumenterna från och med september år 2019 men apparatens hemsidor (www.mookkie.com) är inte tillgängliga samt på Voltas hemsidor (www.volta.ai) nämns inte produkten överhuvudtaget. Liksom för CatFi-produkten som nämndes i början av detta kapitel verkar det som om företaget för Mookkie har upphört apparatens tillverkning.

2.3 Sammanfattning

När det gäller algoritmer för igenkänning av katter, ser det enligt Adam Kleins forskning (figur 2) ut som att djupinlärning med konvolutionella neurala nätverk åstadkommer bra resultat. Med att använda molnbaserad programvara för igenkänning, så som

Arkaitz Garro gjorde, kan man med liten insats åstadkomma en fungerande lösning. Igenkänningen kan dock inte köras lokalt och kommer att kräva extra kostnader för att upprätthålla tjänsten.

Apparaterna CatFi (figur 3) och Mookkie (figur 4) verkar på papper som bra koncept men av en eller annan anledning har tillverkningen av båda systemen avslutats. Tyvärr har produkternas tillverkare inte delat med sig källkoden för igenkänningsalgoritmerna och därmed är det omöjligt att veta exakt vad som använts.

Efter en utförlig undersökning av de olika utvecklade teknikerna och systemen för igenkänning av katter, verkar det som om det inte finns någon fullt fungerande kombination av en igenkänningsalgoritm med ett automatiserat system. Detta examensarbete går in på en närmare analys och jämförelse av olika metoder för igenkänning av katter i strävan efter att bygga upp en funktionell ansiktsigenkänningsbaserad dörrkontroll för katter med högsta möjliga prestanda.

3 UTVECKLING AV SYSTEMET

3.1 Val av igenkänningsmetod

Enligt utförda undersökningar av igenkänningsmetoder i kapitel 2.1, verkar djupinlärning vara bästa alternativet då det kommer till noggrannhet. Målet är dock att slutliga igenkänningssystemet ska kunna användas också på en enkortsdator med mindre datorkraft och därför spelar snabbheten en viktig roll i valet av metoden för igenkänning.

I Srivastava et al. (2021) artikel ”Comparative analysis of deep learning image detection algorithms” jämförs de tre populäraste djupinlärningsmodellerna för bildigenkänning: Single Shot Detection (SSD), Faster Region based Convolution Neural Networks (Faster R-CNN) och You Only Look Once (YOLO). YOLO använder endast ett faltningsnätverk för förutsägelse och klassificering av flera olika objekt i en bild och använder logistisk regression för att lösa objektsdetekteringen. Igenkänningsprocessen slutförs

med ett enda system från början till slut genom att placera utdata från originalbilden i en klass och position.

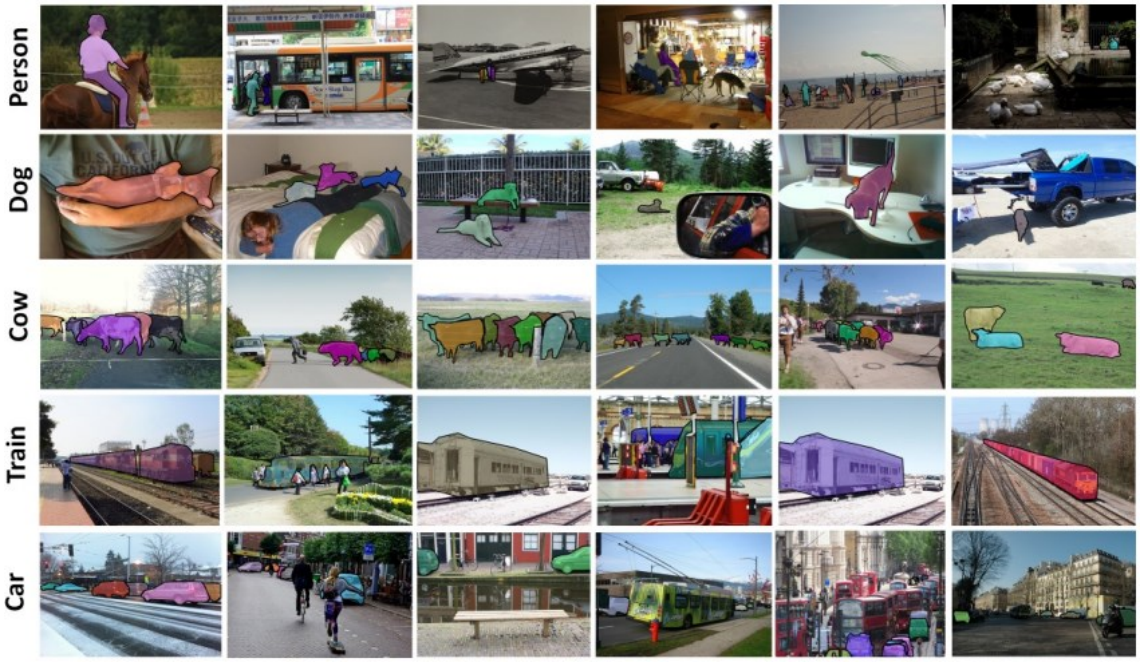
Det visade sig att YOLOv3 (Version 3 av YOLO) var snabbast, SSD följer tätt efter och Faster R-CNN kommer på sista plats. Användningsfallet påverkar dock vilken algoritm man vill använda; om man jobbar med ett relativt litet dataset och inte behöver resultat i realtid är det bäst att välja Faster R-CNN. YOLOv3 är den man ska välja ifall man behöver analysera en videoström från en kamera. (Srivastava et al. 2021).

Även om forskningen av Srivastava et al. (2021) är relativt ny, har YOLOv7 (Version 7 av YOLO) redan utvecklats sedan artikeln publicerats. YOLOv7 överträffar alla tidigare djupa inlärningsmodeller för bildigenkänning samt versionerna av YOLO både när det gäller hastighet och noggrannhet. Den kräver flera gånger billigare hårdvara än andra neurala nätverk och kan tränas mycket snabbare på små dataset utan förtränade vikter. (Boesch G. 2023).

Resultaten av ovan nämnda forskningar pekar tydligt i riktning mot YOLO som val för bildigenkänning för detta arbete.

3.2 Träningsdatasettets förberedelse

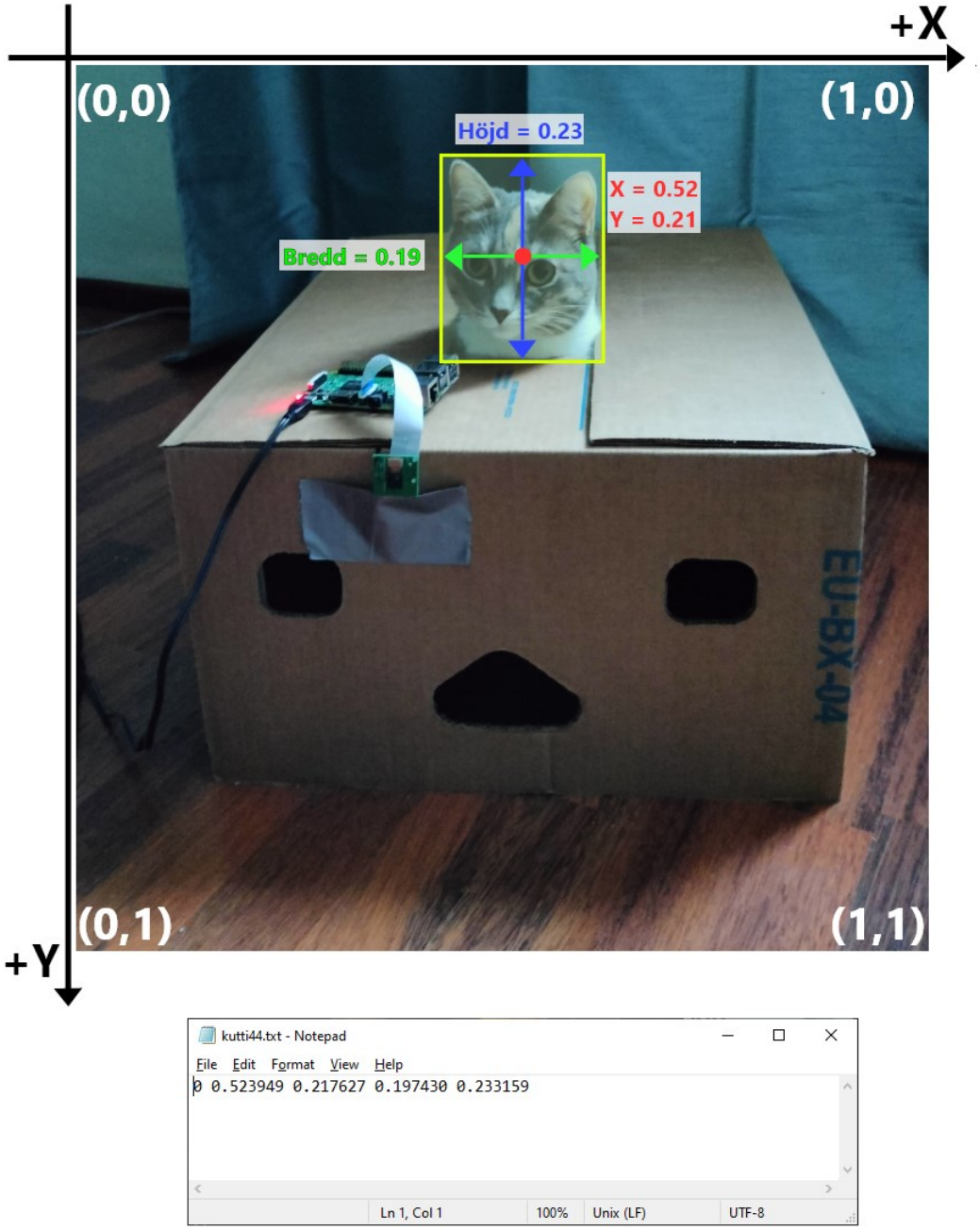
För att använda YOLOv7 för examensarbetets syfte måste en anpassad igenkännings-modell byggas upp och tränas. YOLOv7 kommer med färdiga modeller som är tränade för att känna igen 80 olika objekt. Modellerna är tränade på The Microsoft Common Objects in COntext (MS COCO) dataset som innehåller 91 kategorier av allmänna objekt och 82 av dem har mer än 5 000 märkta instanser, se figur 5. Totalt har datasetet 2 500 000 märkta instanser i 328 000 bilder. (Lin et al. 2015).



Figur 5. Exempel på märkta bilder i datasetet MS COCO. (Lin et al. 2015)

En anpassad modell för YOLO tränas med att skapa en egen dataset av märkta bilder. För att inte själv behöva ta tusentals bilder kan man med hjälp av överföringsinlärning återanvända en förtränad modell för ett nytt syfte.

Innan en ny anpassad modell kan tränas, måste man samla en uppsättning av bilder på objektet som modellen ska lära sig känna igen. Indexet, koordinaterna på mittpunkten och normaliserade bredden samt höjden på objektet i bilderna ska sedan märkas och sparas i en textfil i samma katalog med bilderna, se figur 6. Det finns flera olika sätt och verktyg att märka bilder. I detta arbete användes LabelImg, ett Python verktyg med öppen källkod som automatiskt sparar informationen på markerade objektet i en textfil.



Figur 6. Indexet, koordinaterna och storleken på objektet i bilden ”kutti44.jpg” sparad i textfilen ”kutti44.txt”

Datasettet består av sammanlagt 100 bilder varav hälften är av katten ”Kutti” med klassindex 0 och andra hälften av katten ”Saku” med klassindex 1. Uppsättningen av bilderna är enligt god djupinlärningspraxis delad i en katalog som innehåller 80% av bilderna som används för träningen av modellen och i en annan katalog som består av 20% av bilderna som används för validering av modellens noggrannhet under träningsprocessen. Valideringsprocessen ger information som hjälper i justering av modellens konfigurationer.

3.3 Förberedelser av träningsmiljön

För att träna en ny modell kan antingen ett grafikkort eller en processor användas för att utföra träningsarbetet men även om det är möjligt med en processor, är det betydligt snabbare att använda ett grafikkort. Grafikkorten kan utföra flera parallella beräkningar, vilket gör det möjligt att distribuera träningsprocesser och kan avsevärt påskynda djupinlärningsprocesser. Med ett grafikkort kan man lägga samman många kärnor som använder färre resurser utan att offra effektivitet eller kraft. (Run.ai 2023). För att spara tid och resurser utnyttjades Google Colab i detta arbete. Colaboratory, eller Colab, är en produkt från Google Research. Colab gör det möjligt för vem som helst att skriva och köra Pythonkod via webbläsaren, och är särskilt lämpad för djupinlärning, dataanalys och utbildning. Mer tekniskt sett är Colab en Jupyter-notebook-värdtjänst som inte kräver någon installation för att användas och utöver detta ger också gratis tillgång till beräkningsresurser, inklusive kraftfulla grafikkort. (Google Research 2023).

Innan träningsprocessen kunde startas, måste miljön först ställas upp. Detta började med att installera YOLOv7 först lokalt genom att klona källkoden från YOLOv7s Github kodförråd med kommandot: ”*git clone https://github.com/WongKinYiu/yolov7.git*”. Det är möjligt att installera verktyg och paket direkt till en Google Colab session men problemet är att ifall kopplingen till sessionen avslutar, rensas lagret kopplat till den sessionen. För att undvika detta, laddades installationskatalogen till molntjänsten Google Drive som i sin tur användes som lagringsutrymme för Google Colab sessionen.

Före YOLOv7 kunde användas, måste alla de Python moduler och paket installeras som verktyget använder. För att inte själv hamna lista ut vilka paket och versioner ska installeras är dessa listade i en textfil: ”*requirements.txt*” i installationskatalogen. Textfilen innehöll två föråldrade paket för ramverket PyTorch; ”*torch>=1.7.0,!=1.12.0*” och ”*torchvision>=0.8.1,!=0.13.0*” som inte än hade fullständigt stöd för modellträning med ett grafikkort med nyare CUDA version. PyTorch är ett ramverk för att bygga djupinlärningsmodeller och är känt för sitt utmärkta stöd för grafikkort. CUDA är en parallell beräkningsplattform och programmeringsmodell som möjliggör dramatiska

ökningar av beräkningsprestanda genom att utnyttja kraften av processorerna i ett grafikkort. (Nvidia 2023). Textfilens gamla PyTorch versioner ersattes med nyare och en hänvisning till CUDA version 11.3 lades till, se figur 7.

```
#torch>=1.7.0,!=1.12.0
#torchvision>=0.8.1,!=0.13.0
-i https://download.pytorch.org/whl/cu113
torch==1.11.0+cu113
torchvision==0.12.0+cu113
```

Figur 7. Gamla PyTorch versionerna ersatta med version som stöder version 11.3 av CUDA.

Eftersom meningen var att träna en anpassad modell för att känna igen objekten i bilderna som märktes tidigare med verktyget LabelImg, var nästa steg att redigera konfigurationsfilen ”*coco.yaml*”. Allt annat kunde raderas från filen förutom kataloghänvisningen till egna dataset, antalet klasser samt namnen på klasserna, se figur 8. För tydlighetens skull ändrades filnamnet till ”*custom_cfg.yaml*”.

```
# path to data:
train: data/images/train
val: data/images/validation

# number of classes
nc: 2

# class names
names: [ 'kutti', 'saku' ]
```

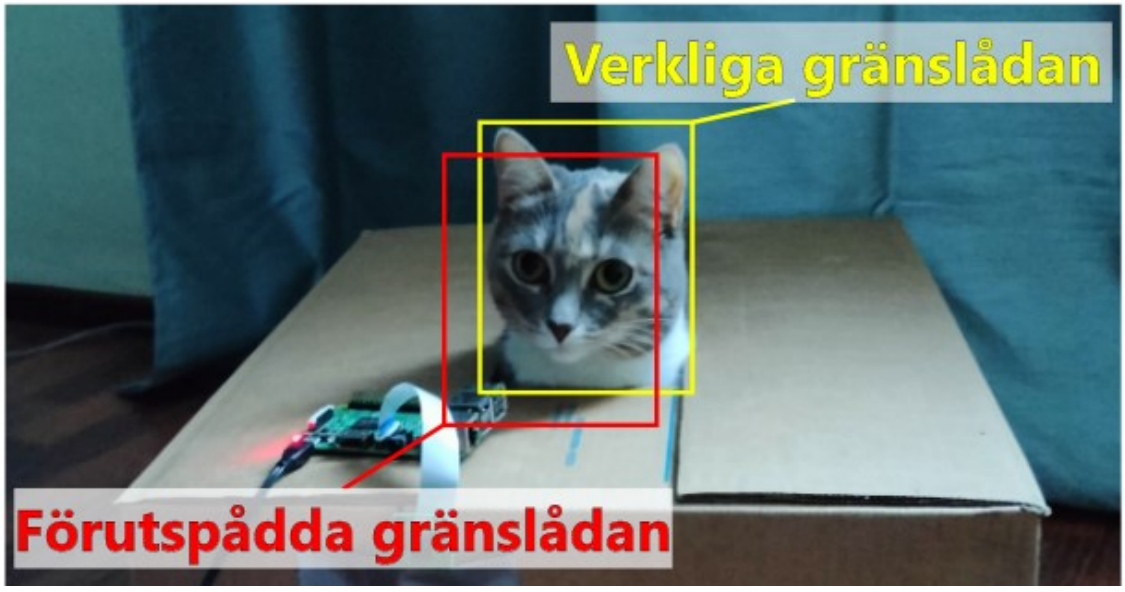
Figur 8. Redigerad konfigurationsfil ”custom_cfg.yaml”.

Sista steget före installationskatalogen och datasettet laddades upp till Google Drive var att ladda ner en förtränad YOLOv7 modell för överföringsinlärningen. Som förtränade modell valdes den minsta och snabbaste modellen ”*YOLOv7*” enligt prestandatabellen från YOLOv7s Github kodförråd, se figur 9.

Model	Test Size	AP^{test}	AP_{50}^{test}	AP_{75}^{test}	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

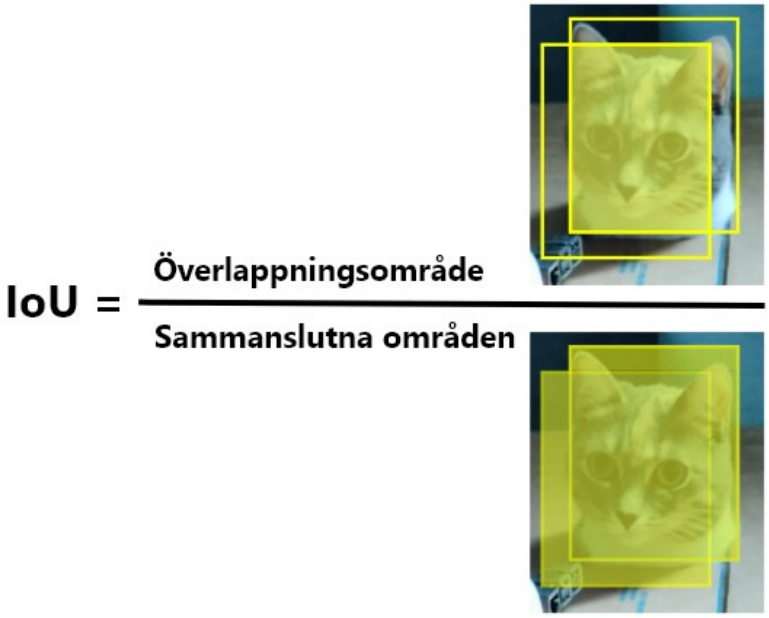
Figur 9. Prestandatabellen från YOLOv7s Github kodförråd (Github 2023. Official YOLOv7)

Modellen har en genomsnittlig noggrannhet (AP) på 51,4 % på data som inte har använts för träning. AP^{test} är ett medelvärde av 10 olika ”*Intersection over Unions*” (IoU) värden från 0.50 till 0.95 med steg på 0.05. IoU används för att utvärdera objektdetekteringens prestanda genom att jämföra den verkliga gränslådan med den förutspådda gränslådan, se figur 10. (PyImageSearch 2016). Hastigheten på modellerna är framställda prestandatabellen på två olika sätt, ”*batch 1 fps”* och ”*batch 32 average time”*. ”*Batch 1 fps”* är hastigheten på själva igenkänningen av en bild, mätt i bilder per sekund. ”*Batch 32 average time”* tyder på hastigheten i millisekunder på träning av 32 sats av träningsbilder per epok innan modellen uppdateras. Dessa är mätt med ett Nvidia V100 Tensor grafikkort vilket är det mest avancerade grafikkort för datacenter som någonsin byggts för att påskynda AI, högpresterande beräkningar (HPC), datavetenskap och grafik. (Nvidia 2023. NVIDIA V100 TENSOR CORE GPU).



Figur 10. Ett exempel på en objektdetektering där gula är den verkliga gränslådan me-dan röda är gränslådan som anpassade modellen har förutspått.

Själva IoU värdet beräknas enligt ekvationen demonstrerad i figur 11 där området på lådorna som överlappar varandra delas med området på båda lådornas sammanslutna område.



Figur 11. Ekvation för beräkning av IoU.

Förberedelserna av träningsmiljön var klara och installationskatalogen av YOLOv7 samt märkta bilderna laddades upp till Google Drive inför träningen av modellen.

3.4 Träningsprocess

Det första steget av träningsprocessen var att skapa en anslutning till katalogen för YOLOv7 installationen och datasettet från Google Drive till Google Colab genom att köra koden, se figur 11.

```
from google.colab import drive
drive.mount('/content/drive')
```

Figur 12. Kod för att skapa Google Drive anslutning till Google Colab miljön.

Andra steget innebar att installera de paket som YOLOv7 är beroende av för att fungera. Alla paket listade i ”*requirements.txt*” filen installerades till miljön med hjälp av kommandot ”*pip install*”, se figur 12. På andra raden i samma figur byts nuvarande arbetskatalog med hjälp av kommandot ”*cd*” till YOLOv7s installationskatalog.

```
!pip install -r drive/MyDrive/yolov7/requirements.txt
%cd drive/MyDrive/yolov7/
```

Figur 13. Kommandot för att installera paketen listade i ”requirements.txt” filen samt för att byta nuvarande katalog.

I tredje steget påbörjades själva träningen av anpassade igenkänningsmodellen med hjälp av kommandot med smått redigerade parametrar demonstrerat på YOLOv7 kodförråd, se figur 14. Träningsprocess upprepades tre gånger med epokerna 50, 150 och 200. En träningsprocess med 300 epoker påbörjades men gränsen på användningen av Google Colabs gratis grafikkort kom emot och processen avslutades automatiskt.

```
!python train.py --workers 8 --device 0 --batch-size 16 --data data/custom_cfg.yaml
--epochs 100 --img 640 640 --hyp data/hyp.scratch.custom.yaml
--name yolov7-kutti-ja-saku-100epochs_100imgs --weights yolov7.pt
```

Figur 14. Smått redigerat kommando för att påbörja träningen. (Github 2023. Official YOLOv7).

Efter en total träningstid på ca 8 timmar blev fyra anpassade YOLOv7-modeller tränade på 80 bilder.

4 TESTNING AV MODELLERNA

I detta kapitel presenteras testmetoderna och -resultaten av anpassade modellernas noggrannhet och hastighet.

4.1 Noggrannhet

För testning av modellens noggrannhet togs ytterligare 25 bilder av katten ”Kutti” med klassindex 0 och 25 bilder av katten ”Saku” med klassindex 1 varefter bilderna laddades upp till Google Drive. YOLOv7 kodförråd innehåller också ett Python skript ”*test.py*”, vilken används för att testa noggrannheten av en YOLO modell mot en uppsättning av bilder i en katalog. Skriptet användes på 50 testbilderna med kommandot demonstrerat i figur 15.

```
!python test.py --data data/custom_test_set.yaml --img 640 --batch 16 --conf 0.001
--iou 0.65 --device 0 --weights yolov7-kutti-ja-saku-50epochs_100imgs.pt
--name yolov7-kutti-ja-saku-50epochs_100imgs_test
```

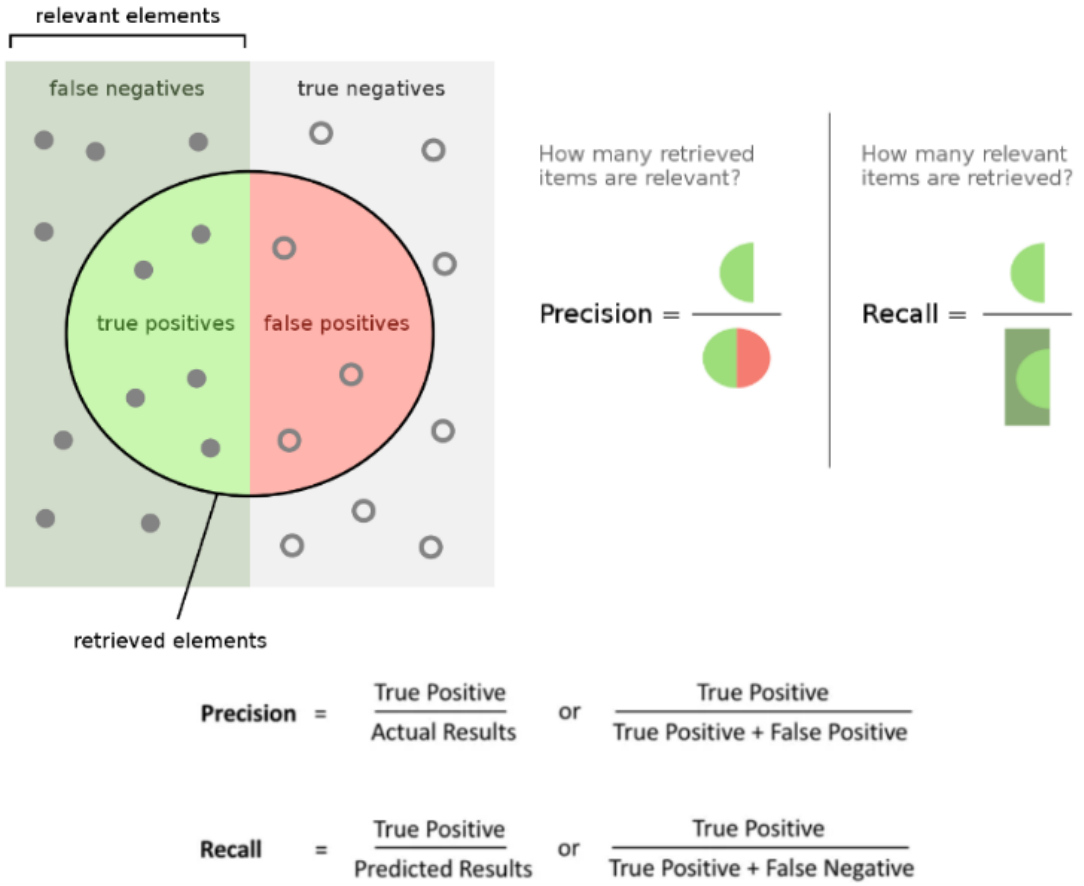
Figur 15. Smått redigerat kommando för att påbörja testningen av en modell. (Github 2023. Official YOLOv7).

Samma kommando upprepades enskilt för alla fyra modeller och resultaten dokumenterades och är presenterade i tabell 1.

Tränings-epok	Klass	Antal bilder	Precision	Återkallelse	F1-värde	AP^{test}
50	Båda (0,1)	50	64 %	64 %	64 %	40,7 %
50	Kutti (0)	25	72,5 %	72 %	72,2 %	52,9 %
50	Saku (1)	25	55,5 %	56 %	55,7 %	28,4 %
100	Båda (0,1)	50	71,5 %	79,9 %	75,5 %	66,6 %
100	Kutti (0)	25	68,7 %	87,8 %	77,1 %	67,8 %
100	Saku (1)	25	74,2 %	72 %	73,1 %	65,4 %
150	Båda (0,1)	50	91,2 %	86 %	88,5 %	80,2 %
150	Kutti (0)	25	88 %	88 %	88 %	81,6 %
150	Saku (1)	25	94,4 %	84 %	88,9 %	78,8 %
200	Båda (0,1)	50	87,5 %	60 %	71,2 %	60,7 %
200	Kutti (0)	25	82,7 %	72 %	77,0 %	63,8 %
200	Saku (1)	25	92,3 %	48 %	63,2 %	57,6 %

Tabell 1. Resultaten av anpassade modellernas precision på 50 testbilder.

Precisionen i tabellen fås genom att dela sant positiva resultaten med de sant positiva resultaten adderat med de falskt positiva resultaten. Återkallelsen (*recall*) räknas i sin tur med att de sant positiva resultaten delas med de sant positiva resultaten adderat med de falskt *negativa* resultaten. F1-värdet tar i hänsyn både precision och återkallelsen för att mäta modellens noggrannhet. Falskt positiva och falskt negativa resultat har en stor inverkan då man mäter noggrannheten av en modell, medan sant negativa resultat ofta är mindre viktiga. F1-värdet försöker ta hänsyn till detta genom att ge större vikt åt falska negativa och falska positiva resultat samtidigt som man inte låter ett stort antal sant negativa resultat påverka F1-värdet, se figur 16. (Riggio C. 2019).



$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Figur 16. Illustration på precision och återkallelse samt beräkning av F1-värdet. (Riggio C. 2019).

När man tittar på tabellen 1, kan man se att den modell som tränades för 150 epoker har den högsta noggrannheten jämfört med de andra modellerna. Modellen som tränats för 200 epoker har redan betydligt (ca 20%) mindre noggrannhet vilket kan tyda på överanpassning. Överanpassning inträffar när en modell presterar bra på träningsdata, men generaliserar inte bra på nya och okända data. Med andra ord har modellen lärt sig mönster som är specifika för träningsdata och som är irrelevanta för andra data. (Carremans B. 2018).

Redan på basis av dessa resultat kan man dra slutsatsen att det räcker med ca. 150 epoker för att träna en modell på ett träningsdataset med bara 80 bilder. För att ytterligare förbättra noggrannheten kunde modellen tränas med ett betydligt större träningsdataset

men på grund av begränsade resurser, räcker den uppnådda noggrannheten för arbetets syfte.

4.2 Hastighet

Modellens hastighet spelar en stor roll i arbetet eftersom slutprodukten kräver att igenkänningen sker i realtid från en videoström. Genom att använda skriptet ”*detect.py*” i YOLOv7s installationskatalog kan man köra objektdetektering på enskilda bilder, video eller en videoström, se exemplet på kommandot i figur 17. Modellen med bästa noggrannheten valdes för att köra kommandot.

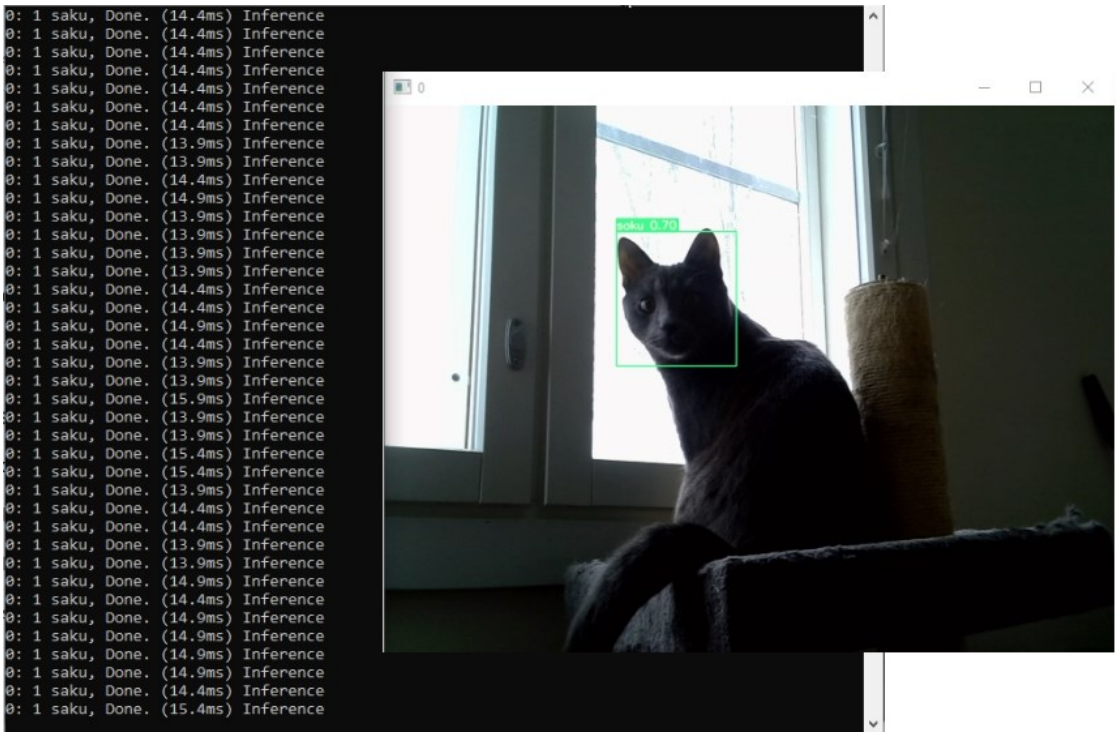
```
python detect.py --weights yolov7-kutti-ja-saku-150epochs_100imgs.pt --conf 0.25 --img-size 640 --source 0
```

Figur 17. Kommando för att köra detektering på en videoström. (Github 2023. Official YOLOv7).

4.2.1 Detektering med en dator i mellanklass

Eftersom skriptet ska komma åt en videoström från en kamera, måste kommandot köras lokalt på ett system utrustad med en kamera.

Utdata av skriptet ”*detect.py*” innehåller igenkända klassens namn och index samt hastigheten på igenkänningen per bild i millisekunder. Kommandot för detekteringen kördes först på en bordsdator med ett CUDA-utrustat NVIDIA 2070 S grafikkort. Genomsnittliga hastigheten på igenkänning per bild var enligt utdata ca. 0,014 sekunder (14 millisekunder), se figur 18. Hastigheten på denna dator verkade vara lämplig för att användas i scenariot där en katt närmar sig kameran och ska kännas igen. Igenkänningssystemets hastighet måste dock ännu testas på ett mindre system med mindre datorkraft.

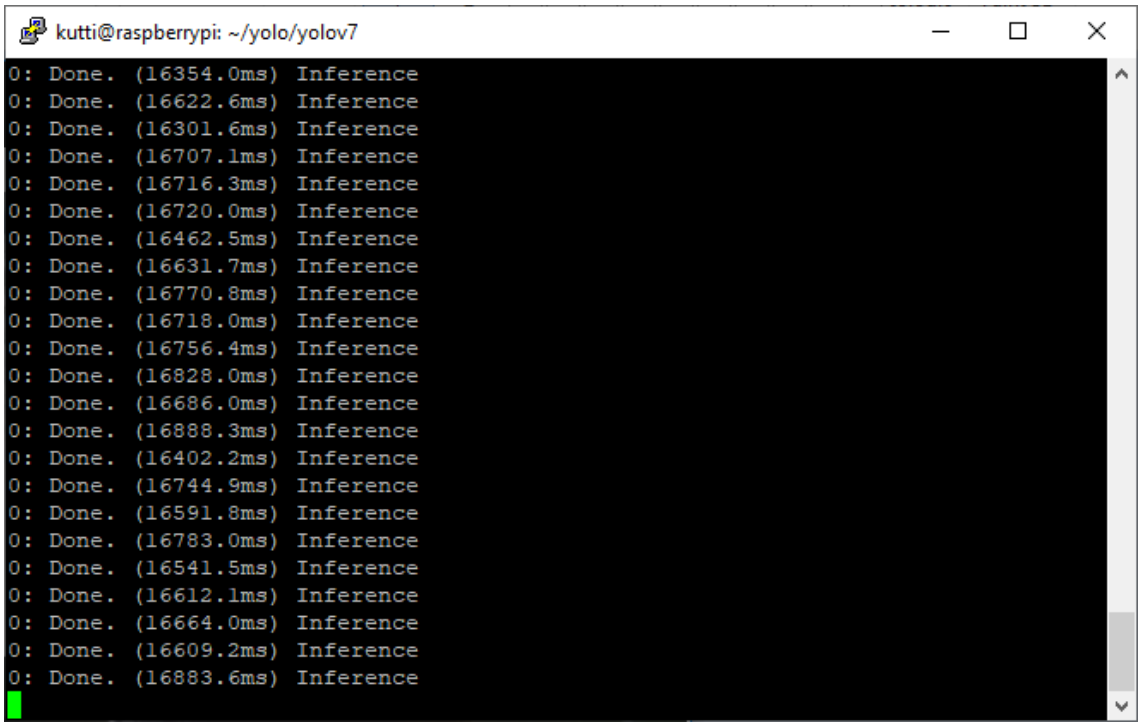


Figur 18. Bild på utdata och fönstret på videoflödet för objektdetekteringen via en webbkamera.

4.2.2 Prestandan på en Raspberry Pi enkortsdator

Raspberry Pi är en billig dator i storlek av ett kreditkort som kan anslutas till en datorskärm eller TV. Den kan göra allt man förväntar dig att en stationär dator ska göra, från att surfa på Internet och spela upp högupplöst video till att göra kalkylark, ordbearbetning och spela spel. (Raspberry Pi 2023).

Detekteringen kördes på Raspberry Pin med samma kommando som demonstrerats i figur 17. Raspberry Pin som användes i detta arbete använder ett operativsystem utan ett grafiskt användargränssnitt för att förbättra prestandan och på grund av det, kördes detekteringskommandot utan ett fönster för förhandsgranskning av videoströmmen. Raspberry Pin saknar ett grafikkort med möjlighet för CUDA och därmed används processorn för grafiska arbeten vilket är betydligt långsammare. Detta kan ses också tydligt på detekteringens utdata, se figur 19.



Figur 19. Bild på detekteringsskriptets utdata av en videoström, körd på en Raspberry Pi dator.

Den genomsnittliga hastigheten på en detektering per bild var ca. 16,5 sekunder (16 500 millisekunder). Resultatet var alldeles för långsamt för att kunna använda igenkännings-systemet på ett system med begränsad datorkraft.

4.3 Slutsatsen av modellernas testning

Efter en granskning av testfasens resultat kan man dra slutsatsen att även om modellen var tillräckligt noggrann uppnåddes inte arbetets önskade hastighet. YOLO visade sig vara en igenkänningsmetod som man med tack vare överföringsinlärning och färdigt tränade modeller kan lätt skapa egna anpassade modeller med relativt små dataset och ändå uppnå god noggrannhet. Nackdelen med YOLOv7s färdigt tränade modeller (figur 9) tycktes dock vara att för att nå god hastighet för igenkänning rekommenderas det att köras på ett system med ett CUDA utrustat grafikkort.

För att utveckla en anpassad igenkänningsmodell vars noggrannhet är av samma klass som modellen som utvecklats under arbetet men vars snabbhet är betydligt bättre krävs ytterligare forskning inom andra YOLO modeller.

5 FRAMTIDA FORSKNING

Under tiden detta examensarbete skrevs publicerade företaget Ultralytics en egen version av YOLO som de kallar Ultralytics YOLOv8. Ultralytics YOLOv8 är en ledande, sista skrikets modell som bygger på tidigare versioner av YOLO och introducerar nya funktioner och förbättringar för att ytterligare öka prestanda och flexibilitet. YOLOv8 är utformad för att vara snabb, noggrann och enkel att använda, vilket gör den till ett utmärkt val för ett stort antal uppgifter inom objektsdetektering, bildsegmentering och bildklassificering. (Github 2023. Ultralytics).

Eftersom Ultralytics YOLOv8 är så nytt kunde inga hastighetsjämförelser mellan YOLOv8 och YOLOv7 hittas. Ultralytics har dock publicerat färdigt tränade modeller på deras kodförråd inklusive en tabell på prestandan, se figur 20. Till skillnad från YOLOv7s minsta modell ”YOLOv7” (figur 9) har Ultralytics publicerat deras minsta och snabbaste modell, ”YOLOv8n” (YOLOv8 Nano). Medan modellen har en 27,43 % sämre genomsnittlig noggrannhet på data som inte använts för träning är den 98,22% snabbare än YOLOv7. Procentuella ökningen i snabbheten är dock beräknat endast med att jämföra snabbheterna i figurerna 9 och 20. Snabbheten i figur 20 är mätt med ett snäppet nyare Nvidia A100 Tensor grafikkort och har troligen också en liten inverkan på ökningen av snabbheten.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figur 20. Prestandatabell på modellerna från Ultralytics Github kodförråd (Github 2023. Ultralytics)

6 SLUTSATSER

Bakgrunden för examensarbetet var att utveckla en lösning för att automatisera övervakningen av huskatter i ett utrymme. För att nå lösningen var arbetets mål att med hjälp av datorseende och en ansiktsigenkänningsalgoritm kunna identifiera en specifik katt med tillräcklig noggrannhet och hastighet. Hastigheten spelade en viktig roll eftersom kattigenkänningen skulle också kunna köras på ett datorsystem med mindre datorkraft.

Före den praktiska delen av igenkänningssystemets utveckling undersöktes andra liknande lösningar och metoder för igenkänning av katter. Metoden som användes i examensarbetet för igenkänning av specifika kattansikten valdes på basis av resultat från relaterad forskning inom jämförelser av olika igenkänningsmetoder och -algoritmer. Den praktiska delen, utvecklingen av en anpassad modell för igenkänning av två olika kattansikten, utfördes helt och hållet i en Google Colab miljö för att spara resurser. Utvecklade igenkänningsmodellens noggrannhet och hastighet testades för att fastställa om de var lämpliga för examensarbetets syfte. Hastighetstesterna utfördes på två olika datorsystem, en mellanklassig bordsdator och en Raspberry Pi enkortsdator med låg datorkraft. En lämplig noggrannhet uppnåddes, men igenkänningsmodellens hastighet på en Raspberry Pi dator var långsam.

Målet med examensarbetet nåddes delvis. Ett system utvecklades som med hjälp av datorseende och en igenkänningsalgoritm kunde med bra noggrannhet identifiera och skilja åt två olika katter men på grund av dålig snabbhet kunde igenkänningssystemet inte användas på en dator med mindre datorkraft. Att implementera igenkänningssystemet på en lägre presterande maskin kräver mer forskning och testande med andra djupinlärningsmodeller, och YOLOv8s Nano modell verkar vara ett bra ställe att fortsätta på.

KÄLLOR / REFERENCES

Nadejda A., 2022. *Pet Face Recognition: Are We There Yet?*

Tillgänglig: https://readwrite.com/pet-face-recognition-are-we-there-yet/

Hämtad 30.1.2023

Klein A., 2019. *Pet Cat Face Verification and Identification*

Tillgänglig: http://cs230.stanford.edu/projects_fall_2019/reports/26251543.pdf

Hämtad 30.1.2023

Price R., 2018. *This engineer built a facial recognition gadget to notify him when his cat wants to come inside.*
Tillgänglig: https://www.businessinsider.in/this-engineer-built-a-facial-recognition-gadget-to-notify-him-when-his-cat-wants-to-come-inside/articleshow/63098474.cms
Hämtad 5.2.2023

Amazon Web Services (AWS), 2022. *Amazon Rekognition Image.*
Tillgänglig: https://aws.amazon.com/rekognition/image-features/?nc=sn&loc=3&dn=2
Hämtad 5.2.2023

Indiegogo, 2014. *Bistro: A Smart Feeder Recognizes Your Cat's Face.*
Tillgänglig: https://www.indiegogo.com/projects/bistro-a-smart-feeder-recognizes-your-cat-s-face#/
Hämtad 6.2.2023

Marchese K., 2019. *the mookkie pet bowl uses facial recognition to spot your cat and feed it food.*
Tillgänglig: https://www.designboom.com/technology/mookkie-smart-pet-bowl-01-07-2019/
Hämtad 6.2.2023

Businesswire, 2019. *Italian Firm Volta Wins Innovation Award With Mookkie, AI-Driven Pet Feeder That Recognizes Your Cat or Dog.*
Tillgänglig: https://www.businesswire.com/news/home/20190104005295/en/Italian-Firm-Volta-Wins-Innovation-Award-With-Mookkie-AI-Driven-Pet-Feeder-That-Recognizes-Your-Cat-or-Dog
Hämtad 6.2.2023

Github, 2017. *MotionEyeOS.*
Tillgänglig: https://github.com/motioneye-project/motioneyeos/wiki
Hämtad 13.2.2023

Papers With Code, 2019. *EfficientNet.*
Tillgänglig: https://paperswithcode.com/method/efficientnet
Hämtad 13.2.2023

Srivastava S., Divekar Vishvas A., Anilkumar C., Naik I., Kulkarni V. & Pattarbiramn V., 2021. *Comparative analysis of deep learning image detection algorithms.*
Tillgänglig: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00434-w
Hämtad 16.2.2023

Boesch G. 2023. *YOLOv7: The Most Powerful Object Detection Algorithm (2023 Guide)*
Tillgänglig: https://viso.ai/deep-learning/yolov7-guide/
Hämtad 16.2.2023

Lin T., Maire M., Belongie S., Bourdev L., Girshick R., Hays J., Perona P., Ramanan D., Zitnick C. L., Dollár P. 2015. *Microsoft COCO: Common Objects in Context*
Tillgänglig: https://arxiv.org/pdf/1405.0312.pdf
Hämtad 17.2.2023

Run.ai 2023. *Making the Most of GPUs for Your Deep Learning Project*
Tillgänglig: https://www.run.ai/guides/gpu-deep-learning
Hämtad 17.2.2023

Google Research 2023. *What is Colaboratory?*
Tillgänglig: https://research.google.com/colaboratory/faq.html
Hämtad 17.2.2023

Nvidia 2023. *CUDA FAQ*
Tillgänglig: https://developer.nvidia.com/cuda-faq
Hämtad 18.2.2023

Github 2023. *Official YOLOv7*
Tillgänglig: https://github.com/WongKinYiu/yolov7
Hämtad 19.2.2023

PyImageSearch 2016. *Intersection over Union (IoU) for object detection*
Tillgänglig: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
Hämtad 20.2.2023

Nvidia 2023. *NVIDIA V100 TENSOR CORE GPU*
Tillgänglig: https://www.nvidia.com/en-us/data-center/v100/v
Hämtad 20.2.2023

Riggio C. 2019. *What's the deal with Accuracy, Precision, Recall and F1?*
Tillgänglig: https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021
Hämtad 22.2.2023

Carremans B. 2018. *Handling overfitting in deep learning models*
Tillgänglig: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e
Hämtad 22.2.2023

Raspberry Pi. 2018. *What is a Raspberry Pi?*
Tillgänglig: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/
Hämtad 24.2.2023

Github 2023. *Ultralytics*
Tillgänglig: https://github.com/ultralytics/ultralytics
Hämtad 13.3.2023