

Gjenkjenning av trafikkskilt i sanntid ved hjelp av YOLOv3

IDATT 2502 - Anvendt maskinlæring med prosjekt

Eirik Steira og Mads Lundsgaard

28. november 2021

Sammendrag

En av de større problemstillingene tilknyttet fagfeltet datasyn er intelligente systemer som kan detektere og gjenkjenne trafikkskilt. Praktiske anvendelser av slike løsninger krever at systemet pålitelig og nøyaktig klarer å lokalisere og identifisere trafikkskilt i utfordrende omgivelser.

Denne rapporten presenterer en tilnærming til systemer for gjenkjenning av trafikkskilt basert på YOLOv3 og klassifisering basert på konvolusjonelle nevrale nettverk. Førstnevnte utgjør nettverket for deteksjonen av trafikkskilt, mens sistnevnte står for gjenkjenningen av skiltene. Trening og evaluering av nettverkene blir gjort på German Traffic Sign Detection Benchmark-(GTSDB) [1] og German Traffic Sign Recognition Benchmark-datasettene (GTSRB) [2].

Resultatene viser potensielle både for deteksjon og klassifisering, men etterlater rom for forbedringer.

Nøkkelord— Trafikkskiltgjenkjenning, YOLOv3, konvolusjonelle nevrale nettverk

1 Introduksjon

I den moderne verden har systemer for trafikkskiltgjenkjenning mange anvendelser, deriblant autonome biler, overvåking av trafikk, samt sjåfør- og trafiksikkerhet. Vanligvis har slike systemer to bestanddeler, nemlig detektering og klassifisering. Førstnevnte fokuserer på å lokalisere skiltet i et bilde, mens sistnevnte foretar en detaljert klassifisering for å identifisere dens type.

Trafikkskilt er fundamentalt i veinettverket og har til hensikt å være lett tilgjengelige, synlige og forståelige for trafikanter. Det faktum at veiskiltingen er standardisert og vidt brukt har tilrettelagt for oppbygging av datasett som kan brukes til å trenne et konvolusjonelt nettverk. Likevel er det en signifikant utfordring å produsere et system som pålitelig kan detektere og klassifisere skilt i den virkelige verden som en følge av dens

variabilitet, der lysforhold, vær og andre faktorer varierer konstant.

To hyppig brukte datasett er GTSRB og GTSDB som tilrettelegger for trening av klassifisering og deteksjon av trafikkskilt.

Nyere studier har benyttet konvolusjonelle nevrale nettverk (refereres videre til som CNN¹) for å løse problemet med deteksjon og klassifisering av trafikkskilt. Såkalte to-trinnsdetektorer bestående av et regionforslagsnettverk² og klassifiseringsnettverk som Faster-RCNN [3] har vist seg å gi god nøyaktighet og hastighet, samt fremstår som en lovende løsning for virkelige problemer. Mens Faster-RCNN fokuserer på å øke hastigheten til tidligere R-CNN-rammeverket, tilbyr ett-trinnsdetektorer som You Only Look Once (refereres videre til som YOLO³) forbedret ytelse i sanntid [4]. Disse lader likevel innenfor generalisering av nye objekter, nøyaktighet og romslige begrensninger i antallet objekter i nærhet til hverandre. I tillegg er de ikke effektive til å detektere mindre objekter, noe som er kritisk for gjenkjenning av trafikkskilt. Nyere forbedringer på vegne av YOLOv2 [5] og YOLOv3 [6] har håndtert de øvrige problemene til en viss grad.

I denne rapporten brukes YOLOv3 objekt-detekterende nettverk og CNN til det formål å detektere og klassifisere trafikkskilt i sanntid. Klassifikatoren er basert på nettverket foreslått av Li og Wang [7].

2 Tidligere arbeid

2.1 Trafikkskiltdeteksjon

Problemet med automatisk deteksjon av trafikkskilt er svært utfordrende som en følge av naturlige forstyrrelser. Eksempler på dette er varierende lysforhold, uskarphet forårsaket av bevegelser, og blokkeringer i og rundt veien. Utendørs lysforhold påvirker i stor grad fargen til trafikkskilt og fargeinformasjonen kan bli fullstendig upålitelig som et primært trekk for deteksjon. Dette har ført til problemer for tidligere metoder som er basert på å trekke ut skiltlokasjoner og klassifisering basert på en fargesannsynlighetsmodell og histogram av orienterte

¹CNN - Convolutional Neural Network

²RPN - Region Proposal Network

³YOLO - You Only Look Once

gradienter⁴[8]. Videre har nyere studier begynt å benytte seg av CNN for objektdeteksjon. Flere forskere har derfor forsøkt å løse problemet med trafikkskiltgjenkjenning ved hjelp av ulike CNN-modeller.

Et regionsbasert CNN ved navn R-CNN (refereres videre til som R-CNN⁵), introdusert i 2014 [9], kom med en løsning til den trege prosesseringen av bilder. I stedet for å forsøke å klassifisere flere tusen regioner så har dette antallet blitt redusert til 2000. Følgelig ble det oppnådd raskere prosessering sammenlignet med tidligere modeller, men kunne fortsatt ikke anvendes på video. For eksempel kunne det ta 53 sekunder for å prosessere et bilde med prosessor (videre referert til som CPU⁶) og 13 sekunder med skjermkort (videre referert til som GPU⁷) [9].

En metode for trafikkgjenkjenning basert på dyp læring er blitt forslått av Y. Zhu et al. [10]. Forslaget inneholder et rammeverk med komponenter for dyp læring som inkluderer et Fully Conventional Network⁸ for veiledede forslag til trafikkskilt og CNN for objektklassifisering. Denne metoden er i stand til å produsere og predikere mer diskriminative kandidater, som bidrar til å gjøre hele deteksjonssystemet raskt og nøyaktig. Samtidig ble det oppnådd høyere presisjon enn R-CNN med høyere hastighet som en følge av et lavere antall forslag til regioner å klassifisere. Etter R-CNN ble introdusert i 2014 [9] er det blitt introdusert en forbedring på vegne av en av skaperne, Ross Girshick. Metoden Girshick la frem ble kalt Fast R-CNN [11].

Z. Zhu et al. [12] la frem et forslag til et CNN som kan detektere og klassifisere trafikkskilt samtidig. Sammenlignet med tidligere CNN-baserte løsninger oppnår denne bedre resultater på objekter som opptok mindre porsjoner av bildet, som for eksempel trafikkskilt. I arbeidet sammenlignes også resultatene med Fast R-CNN [11].

Videre påpeker S. Ren et al. [3] en større utfordring vedrørende beregning av forslag til regioner i forbindelse med Fast R-CNN og introduserer Faster R-CNN. Denne metoden har en forbedret ytelse sammenlignet med forgjengeren, men kommer fortsatt til kort på enkelte områder [13].

I 2016 kom YOLO som et resultat av at daværende algoritmer, som R-CNN, brukte for lang tid på å detektere objekter [4]. De regionsbaserte algoritmene, som Fast-RCNN og Faster-RCNN, så på deler av bildet hvor det var høy sannsynlighet for at det fantes kandidater. Med andre ord så de ikke på bildet som en helhet, men heller i mindre deler. YOLO trenget derimot se på hele bildet én gang, derav navnet **You Only Look Once**. Algoritmen tar inn et bilde, deler det opp i et rutennett, regner ut sannsynligheten for at et objekt finnes i hver ruta og en forskynnings-verdi for hver av disse. Naborutene som har en verdi over en predefinert terskel velges ut og blir brukt for å lokalisere objektet. Følgelig kan objektdeteksjonen foregå ved hjelp av et enkelt konvolusjonelt nevralt nettverk, noe som igjen resulterer i betraktelig bedre prosesse-

ringstid. Sammenlignet med R-CNN som bruker 13 sekunder per bilde ved bruk av GPU [9], kan YOLO prosessere 45 bilder i sekundet [4].

J. Redmon og Ali Farhadi bygget videre på YOLO og kom frem til de nyere utgavene YOLOv2 [5] og YOLOv3 [6].

Trafikkskiltgjenkjenning blir fortsatt forsket på som en følge av nytteverdien i praktiske applikasjoner. Nylig har flere forskningsartikler oppnådd solide resultater innen trafikkskiltgjenkjenning ved bruk av YOLOv3 med CNN-baserte klassifikatorer. S. P. Rajendran et al. [14] presenterte en slik tilnærming for trafikkskiltgjenkjenningssystemer og sammenliknet resultatene med et system basert på Faster R-CNN. Resultatene viste god ytelse basert på GTSDB-datasettet.

2.2 Trafikkskiltklassifikator

Tidligere løsninger for klassifisering av trafikkskilt benyttet seg av ulike metoder som baserte seg på Support Vector Machines [15] og Sparse-Representation-Based Graph Embeddings [16]. Denne type løsninger har i stor grad blitt erstattet av CNN.

I nyere tid, med GTSRB-datasettet som fundament, har det blitt oppnådd høy grad av nøyaktighet ved bruk av CNN innenfor dette området [2]. Kombinasjoner av sylinder fra dype nevrale nettverk (refereres videre til som DNN⁹) og flersøylet DNN (refereres videre til som MCDNN¹⁰), foreslått av Semanet et al. [17], har resultert i lave feilrater og en nøyaktighet på 99.46%. En annen tilnærming av A. Arcos-García et al. [18] ved bruk av CNN og Spatial Transformer Networks (refereres videre til som STN¹¹) og T. L. Yuan [19] rapporterte en nøyaktighet på henholdsvis 99.71% og 99.59%. Andre tilnærninger har vært bruk av en såkalt Committee of CNNs [20] med 99.15% nøyaktighet og Hinge Loss-trente CNN [21] på 99.65% på GTSRB.

CNN med asymmetriske filtre ser også gode resultater. J. Li og Z. Wang [7] brukte et effektivt CNN med asymmetriske filtre til klassifiseringen av trafikkskilt og Faster R-CNN for deteksjon med en nøyaktighet på 99.66%. Den nevrale klassifikatoren de har brukt er utgangspunktet til nettverksarkitekturen vi presenterer senere. X. Ding et al. [22] viste i 2019 at denne arkitekturen kan gi forbedret ytelse og oppnå bedre nøyaktighet enn nettverk med standard symmetriske filtre.

3 Metode

I denne seksjonen presenterer vi metoden brukt for klassifisering og gjenkjenning av trafikkskilt basert på henholdsvis CNN og YOLOv3. Systemet består av en YOLOv3 detektor, trent på GTSDB for å detektere trafikkskiltkandidater, og en CNN trent på GTSRB som klassifiserer kandidatene. Sistnevnte vil altså fremlegge prediksjoner blant de 43 klasse-i dette datasettet.

⁴HOG - Histogram of Oriented Gradients

⁵R-CNN - Region-based Convolutional Network

⁶CPU - Central Processing Unit

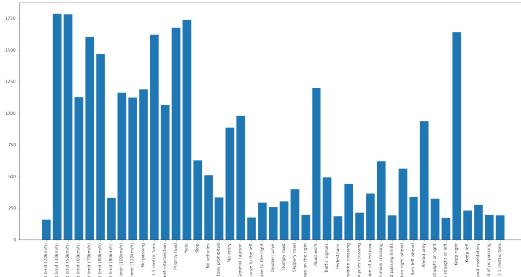
⁷GPU - Graphical Processing Unit

⁸FCN - Fully Convolutional Network

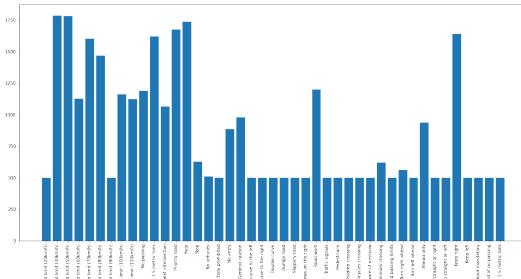
⁹DNN - Deep Neural Network

¹⁰MCDNN - Multi-column DNN

¹¹STN - Spatial Transformer Networks



Figur 1: GTSRB før balansering



Figur 2: GTSRB etter balansering

3.1 Datasett og preprosessering

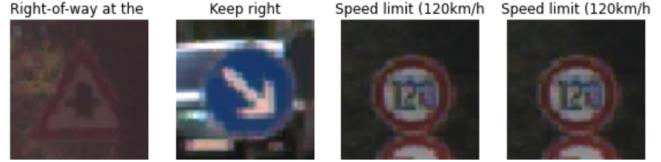
GTSRB inneholder mer enn 50 000 livsvirkelige bilder av trafikkskilt og brukes til flerklasse-klassifisering av enkeltbilder [2]. 39 209 av disse brukes til trening og validering, mens 12 630 brukes til testing. Av det førstnevnte brukes det her 80% til testing og 20% til validering. Bildene er delt opp i 43 klasser og det er en viss grad av variabilitet i lysforhold, kontrast, vinkler og rotasjon.

Selv om vi har tilgang på et stort antall bilder å trenne modellen på, er det en viss ubalanse i datasettet som kan føre til partisk klassifisering. For eksempel er det kun 210 bilder av klassen *Speed limit (20km/h)* i treningsdatasettet, mens det er hele 2250 av klassen *Speed limit (70km/h)*.

Følgelig har vi tatt i bruk ulike metoder for å prosessere bildene i forkant av treningen for å skape mer balansert treningsdata. Dette innebærer å identifisere klassene med bilder under et visst antall og supplementere med modifiserte versjoner av disse i treningsdatasettet. Utvidelser vi gjennomfører er beskjæring, skalering, forskyvning, rotering med +15 grader og dybdetransformasjon på +-5.7 grader. Vi erfarte at å sette en nedre grense for antall bilder for hver enkeltklasse til 500 bilder gir gode resultater.

Bildene varierer også i størrelse fra 15×15 til 250×250 . Under preprosesseringen blir størrelsen til samtlige bilder skalert eller beskjært til å være på formen 48×48 .

Datasettet brukt til trening av detektoren er GTSDB, som er hyppig brukt av forskere innen datasyn, mønstergenkjenning og bildebaseret førerassistanse. Dataset-



Figur 3: Stikkprøver fra GTSRB



Figur 4: Strikkprøver fra GTSRB etter utvidelser

tet består av 900 bilder delt inn i 600 treningsbilder og 300 evalueringssbilder. Det ble brukt en tilpasset versjon av dette datasettet, som inkluderer en tektsfil med informasjon om hvilke av fire kategorier skiltet hører til og dens koordinater for avgrensningsbokser. De fire kategoriene er; pålagt, fare, forbud og annet. Kategoriene brukes for å skille på skiltenes hovedtrekk.

3.2 Trafikkskiltdetektor

3.2.1 YOLOv3

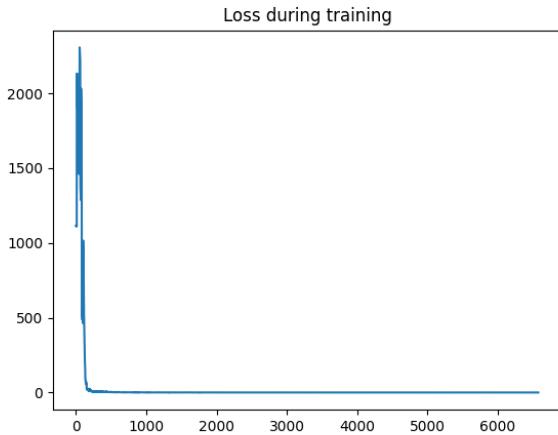
YOLO kommer til kort på enkelte punkter når den sammenlignes med tidligere algoritmer, hvorav detektering av små objekter er et av disse. YOLOv3 søker å nå bedre treffsikkerhet, men det går på bekostning av hastigheten sammenlignet med forgjengerne YOLO og YOLOv2. Ved å bruke en ny underliggende arkitektur oppnår YOLOv3 en mye bedre treffsikkerhet på trekkuthenting, spesielt på mindre objekter hvor den oppnår 3 ganger bedre resultater sammenlignet med forgjengeren [6]. YOLOv3 eigner seg derfor til bruk for detektering av trafikkskilt på video, da den har en tilstrekkelig treffsikkerhet på små objekter, samtidig som ytelsen holder mål.

3.2.2 Arkitektur

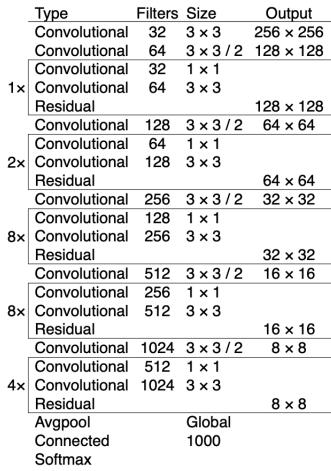
Den nye underliggende arkitekturen som YOLOv3 baserer seg på kalles Darknet-53 og består av 53 konvolusjonelle lag som figur 7 illustrerer. Arkitekturen legger til rette for bedre ut-

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	171
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

Figur 5: Darknet-53 sammenlignet med andre arkitekturen



Figur 6: Illustrasjon av loss under treningen til YOLOv3 basert på **feil kodebase**



Figur 7: Arkitekturen til YOLOv3 [6]

henting av særtrekk sammenlignet med Darknet-19, som YOLOv2 er basert på. Selv om Darknet-53 oppnår lavere fps¹² enn forgjengeren, så utnytter den GPU-en bedre ved å ha et vesentlig høyere antall flyttall-operasjoner i sekundet, samtidig som den oppnår høyere presisjon ved deteksjon [6], som figur 5 viser. Videre bygger YOLO på Darknet-53 ved å legge til 53 lag ekstra for bedre deteksjon. Et særtrekk med dette er at nettverket predikrer på tre ulike nedskaleringer, hvor den skalerer ned med 32, 16 og 8 henholdsvis. Dette gjør at nettverket lettere kan detektere objekter av ulik størrelse.

3.2.3 Trening av YOLO-modellen

YOLOv3 modellen som brukes har blitt trent på datasettet benyttet av Sichkar V.N. og Kolyubin S.A. i forbindelse med

¹²Frames per second/Bilder per sekund

deres forskningsprosjekt på datasyn for autonome fartøy [23]. Det tilsvarer GTSDB på et format som kan tolkes av YOLOv3, der hvert bilde har en tekstfil med riktig informasjon om klasse og avgrensningsboksens koordinater.

Det ble benyttet to ulike kodebasar for trening av detektor, hvorav den ene var utdatert og ga ingen informasjon om presisjon oppnådd. Loss fra trening med feil kodebase er illustrert i figur 6. Det ble begrenset med trening på rett kodebase, visualisering av treningen ble derfor ikke mulig å anskaffe.

3.3 Trafikkskiltklassifikator

Den nevrale nettverksklassifikatoren brukt til klassifisering av trafikkskilt i denne rapporten er basert på [14] og originalt hentet fra [7]. Med enkelte modifiseringer oppnådde vi et raskt og nøyaktig nettverk vist i tabell 1. Nøyaktighet benyttes som målet for å evaluere ytelsen til klassifikatoren.

3.3.1 Arkitektur

Arkitekturen består av et konvolusjonslag med 32 filtre av størrelsen 3×3 i første omgang etterfulgt av $n \times 1$ konvolusjonslag og et nytt lag med $1 \times n$ konvolusjonslag. Sistnevnte er i motsetning til mer standard arkitekturen, som gjerne består av $n \times n$ konvolusjonslag. Asymmetriske konvolusjonslag på formen $1 \times n$ og $n \times 1$ blir brukt som erstatning for mer standard symmetriske $n \times n$ konvolusjonslag. Det ble erfart at en slik sammensetning gir økt nøyaktighet i prediksjonene på både valideringsdatasettet og testdatasettet. Denne tilnærmingen har vist seg å redusere både antall konvolusjonelle operasjoner og antallet parametre i kjernene, som fører til raskere komputasjoner og økt hastighet [7]. For eksempel vil et konvolusjonslag med filtre av dimensjonen 5×5 tilsvare et redusert antall konvolusjonelle operasjoner med $(5 \times 5 - 1 \times 5 - 5 \times 1) / (5 \times 5) \approx 60\%$ (her ses det bort ifra dimensjonene på utdataen, da denne ikke vil endres etter den kovolusjonelle operasjonen).

Nettverket består også en Inception Module [25], som oppgitt i figur 8, bestående av tre konvolusjonslag som opererer på samme nivå, med forskjellig storrelse på filtrene, fremfor sekvensielt. Disse består av to lag med asymmetriskefiltre på henholdsvis $1 \times 3 + 3 \times 1$, $1 \times 5 + 5 \times 1$ og $1 \times 7 + 7 \times 1$. Disse blir så sammenkoblet og ført videre til etterfølgende lag. Her har det dog blitt utført enkelte modifiseringer og forenklinger sammenlignet med den originale arkitekturen til Inception Module. 1×1 konvolusjonslag er byttet ut med 7×7 og alle lagene er i stedet 2 lag med asymmetriskefiltre for å redusere antall konvolusjonelle operasjoner ytterligere. Følgelig reduseres kostnaden ved beregning. Grenen med Max-Pool-laget er heller ikke inkludert, da den også var ekskludert fra [7].

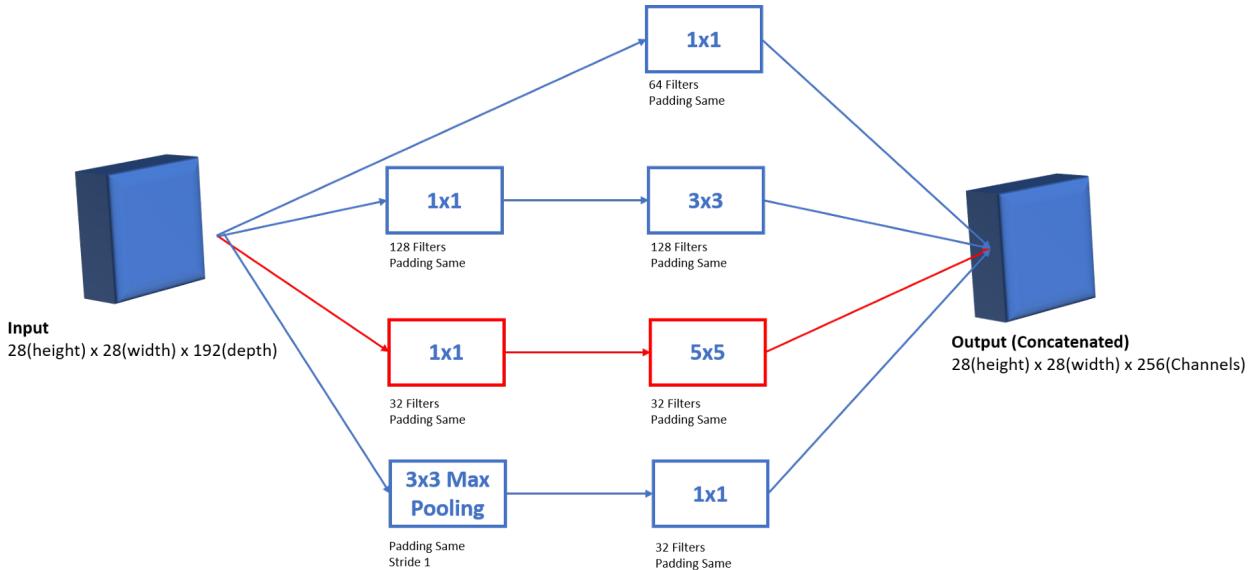
Arkitekturen i sin helhet er angitt i tabell 1 og det benyttes følgende forkortelser:

Conv: vanlig konvolusjonslag

s1: lagets stride er 1×1

s2: lagets stride er 2×2

I likhet med vanlig praksis innen standard CNNs, er hvert konvolusjonslagslag etterfulgt av Batch Normalization- og



Figur 8: Komplett Inception Module [24]

ReLU-lag. En slik tilnærming vil føre til at nettverket konvergerer raskere, mens det forenkler optimaliseringen av modellen, samt gir bedre nøyaktighet.

Dropout brukes for å oppnå bedre generalisering i modellen og økt nøyaktighet under testing. Dropout-ratene i tabell 1 er satt til henholdsvis 0.2, 0.2, 0.3 and 0.4.

Siden det skal klassifiseres 43 klasser i GTSRB-datasettet, gir det siste Dense-laget 43 outputs med Softmax-aktivering.

Modellen trenes på 200 epoker med en batch size på 60, med og uten utvidelser av datasettet. Utvidelsene er beskrevet i seksjon 3.1

3.3.2 Gradient Descent Optimaliseringsalgoritme

Gradient descent er den foretrukne måten å optimalisere neurale nettverk på, samt en av de mest populære algoritmene for å utføre optimalisering [26]. Det er her blitt brukt implementasjonen av Adaptive Moment Estimation (refereres videre til som Adam) fra biblioteket Keras versjon 2.7.0. Implementasjonen benytter seg av en læringsrate på 0.001 og β_1 , β_2 og epsilon på henholdsvis 0.9, 0.999 og 1e-07. Det er verdiene som brukes under trening av modellen.

Adam beregner tilpasningsdyktige læringsrater for hvert parameter. S. Ruder [26] viste empirisk at denne algoritmen fungerer godt i praksis og er å favorisere sammenlignet med andre optimaliseringsalgoritmer som RMSprop og Adagrad.

3.3.3 Loss-funksjoner

Klassifikatoren bruker log (categorical cross entropy) som tapsfunksjon i problemet ved flerkasse-klassifisering. Det benyttes en kategorisk tapsfunksjon siden det er 43 klasser i GTSRB. Denne tapsfunktjonen er gitt ved følgende formel:

$$Loss = \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i \quad (1)$$

Hvor \hat{y}_i er den i -ende skalaren i utdataen fra modellen, y_i er den tilsvarende målverdien og *output size* er antallet skalarer i modell-utdataen.

For hver ramme i videoen som mates til detektoren, lokaliserer denne skiltene i bildet og tegner en avgrensningsboks. Disse seksjonene blir så beskjæret til å ha dimensjoner på 48 x 48 slik at klassifikatoren så kan predikere hvilket skilt dette er. Den beste prediksjonen blir så vist i utdataen sammen med avgrenningsboksen.

4 Resultater

Under prosjektets forløp har det blitt eksperimentert med forskjellige implementasjoner og justeringer for å se om det ga utslag i resultatene. I begynnelsen ble det benyttet relativt enkle nettverksarkitekturen som utviklet seg i takt med et økt kunnskapsnivå og kjennskap til relevant forskning. Resultatene sammenlignes også med tidligere løsninger nevnt i seksjon 2.2.

Klassifikatoren er trent og evaluert ved bruk av Keras og TensorFlow backend. Den er trent og evaluert på GTSRB, mens for detektoren er det brukt GTSDDB. Samtlige kjøringer under dette prosjektet er gjennomført i Kaggle¹³ som i skrivende stund tilbyr maskiner med Nvidia Tesla P100 skjermkort GPU med 16 GB minne.

¹³<https://www.kaggle.com/>

Tabell 1: Trafikkskiltklassifikatorens arkitektur

Lag	Type	Filterstr/ Parameter	Filtere/ Stride	Form på output
1	Conv	3 x 3	32, s1	48 x 48 x 32
2	Conv	7 x 1	48, s1	48 x 48 x 48
3	Conv	1 x 7	48, s1	48 x 48 x 48
4	MaxPool	2 x 2	s2	24 x 24 x 48
5	Dropout	0.2		24 x 24 x 48
Inception-modul				
6-1	Conv	3 x 1	64, s1	24 x 24 x 48
7-1	Conv	1 x 3	64, s1	24 x 24 x 48
6-2	Conv	7 x 1	64, s1	24 x 24 x 48
7-2	Conv	1 x 7	64, s1	24 x 24 x 48
6-3	Conv	5 x 1	64, s1	24 x 24 x 48
7-3	Conv	1 x 5	64, s1	24 x 24 x 48
8	Concatenate	-	-	24 x 24 x 128
Sammensatte Feature Maps				
9	MaxPool	2 x 2	s2	12 x 12 x 128
10	Dropout	0.2	-	12 x 12 x 128
11	Conv	3 x 3	128, s1	12 x 12 x 128
12	Conv	3 x 3	256, s1	12 x 12 x 256
13	MaxPool	2 x 2	s2	6 x 6 x 256
14	Dropout	0.3	-	6 x 6 x 256
15	Dense	256	-	256
16	Dropout	0.4	-	256
17	Dense - Softmax	43	-	43

4.1 Trafikkskiltdetektor

Selv om modellen brukt for deteksjon er basert på Valentyn Sichkar og Sergey Kolyubin sin forskning [23], inkluderes likevel resultater fra trening av modellen da det er fordelaktig å vite for å gi en helhetlig forventning til det oppnådde resultatet.

4.1.1 Mean Average Precision

Når man oppgir treffsikkerhet på objektdeteksjon så er det vanlig å måle det i gjennomsnittlig presisjon (videre referert til som AP¹⁴) og gjennomsnittlig presisjon fra hver klasse (videre referert til som mAP¹⁵). For å forstå enhetene må man først forstå målenheten IoU¹⁶ og hvordan lokalisering av objekter på et bilde fungerer. Når man forsøker å detektere objekter på et bilde så lager modellen et område hvor den antar at objektet er. Samtidig så har man et område som på forhånd er definert i henhold til hvor det faktiske objektet ligger. Ved å se hvor mye det predikerte området overlapper det faktiske området så regner man ut IoU. Deretter sammenlignes IoU mot en forhåndsatt terskel, som da klassifiserer deteksjonen som sann positiv (videre referert til som TP¹⁷), hvis IoU er over eller lik eller falsk positiv (videre referert til som FP¹⁸).

Videre benyttes to formler, *precision* (2) og *recall* (3), som tar utgangspunkt i klassifiseringen av TP, FP og FN¹⁹. Disse to

¹⁴Average Precision

¹⁵mean Average Precision

¹⁶Intersection over union

¹⁷True Positive - riktig gjetting på et tilstedeværende objekt

¹⁸False Positive - Gjettet på et objekt som ikke var tilstede

¹⁹False Negative - Ikke gjettet på et objekt som er tilstede



Figur 9: Illustrasjon av AP [28]

settes opp i en graf hvor precision er på y-aksen mens recall er på x-aksen, illustrert i figur 9. AP er så regnet utifra grafen ved bruk av formel 4, som baserer seg på summen av formel 5. For å regne ut mAP tar man så gjennomsnittlig AP fra hver klasse [27].

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \dots, 1.0\}} p_{interp}(r) \quad (4)$$

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \quad (5)$$

4.1.2 Trening

Til tross for at GTSDB-datasettet er relativt lite å trenere på, ble det erfart at resultatene var lovende. Ut ifra artikelen som YOLOv3-modellen er basert på [23] så leses det at 97.22% mAP ble oppnådd ved 5700 epoker. Dette er et godt utgangspunkt for deteksjon av trafikkskilt. Resultatet som ble oppnådd etter trening i denne rapporten var 83.83% mAO ved 4000 epoker, noe som er vesentlig lavere enn hva som ble oppnådd i [23].

4.2 Trafikkskiltklassifikator

Ytelsen til modellen ble evaluert basert på nøyaktigheten til klassifiseringen av testdatasettet i GTSRB, bestående av 12 630 bilder. Det ble ikke oppnådd en like høy nøyaktighet som [14] og [7], men resultatene er likevel tilfredsstillende med 99.12% nøyaktighet på testdatasettet. Uten preprosessering var dette tallet på på 98.99%.

Resultatet ovenfor ble oppnådd ved å trenere modellen med 200 epoker og en batch size på 60. Som illustrert i figur 10 konvergerer modellen relativt hurtig. Det ble eksperimentert med ulik batch size til det formål å se om det hadde innvirkning på resultatene.

4.2.1 Arkitektur

På letingen etter en god nevral nettverksarkitektur som ga god nøyaktighet var undertegnede gjennom flere iterasjoner.

Resultatene er gitt som en sammenligning mot vårt endelige resultat, gjengitt i tabell 3.

Første iterasjon var et relativt standard oppsett for CNNs, med tre konvolusjonslag med ReLU-aktivering og filterstørrelse på 3×3 etterfulgt av Max Pool og Dropout med rate 0.25. Her ble også Softmax brukt i siste Dense-lag. Senere ble modellen gjort mer kompleks, med 4 konvolusjonslag, også her med filterstørrelse 3×3 , etterfulgt av ReLU-aktivering og Batch Normalization. Lag med Max Pool fulgte etter konvolusjonslagslag 2 og 4. Et Fully Connected-lag utgjør de siste lagene i denne modellen, med et Dense-lag med parameter 128 etterfulgt av ReLU og Batch Normalization. Det ble erfart at denne versjonen konvergerte raskere enn tidligere og ga bedre nøyaktighet enn de tidligere modellene.

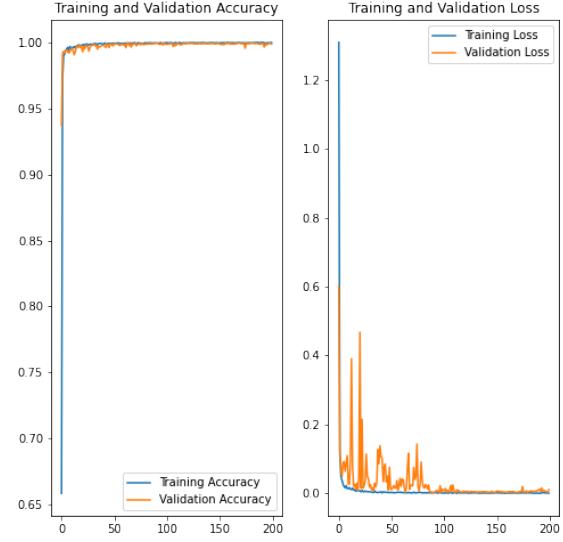
Siste og endelige versjon er avgitt i tabell 1. Her ble det eksperimentert med tre grener i Inception-modulen, hvert med et konvolusjonslag etterfulgt av ReLU-aktivering og Batch Normalization og filterstørrelser på $1 \times 3 + 3 \times 1$, $1 \times 5 + 5 \times 1$ og $1 \times 7 + 7 \times 1$. Andre tester gikk ut på bruk av kun ett konvolusjonslag i Inception-modulen med fire grener, altså 1×3 , 3×1 , 1×7 og 7×1 . Den endelige modellen presterer best blant de øvrige modellene, med 98.99% nøyaktighet uten utvidelser av datasettet og 99.12% med.

4.2.2 Trening

Generelt gir modellene bedre nøyaktighet ved flere epoker den blir trent med. For eksempel ga den endelige versjonen 98.99% nøyaktighet ved 50 epoker og 99.12% med 200 epoker. Resultatene fra trening av denne modellen med 200 epoker og batch size på 60 er oppgitt i figur 10. Det endelige losset/tapet kom på $9.7107e-04$ for treningsdatasettet og 0.0011 for valideringsdatasettet.

Et viktig hyperparameter i nevrale nettverk er størrelsen batchene det blir trent på. Her ble det erfart at en høyere batch size ga bedre nøyaktighet under testingen. Resultatene viste at å trenne i større batches på 60 førte til bedre ytelse enn en lavere størrelse på 16 hvor førstnevnte ga en differanse i nøyaktigheten på 0.03%. Selv om akkurat denne forskjellen kan sies å være minimal, ble det observert gjenomgående høyere nøyaktighet med en høyere batch size ytterligere nøyaktighet.

Batch size henger sammen med gradient descent optimaliseringssalgoritmen og kan forklares ved at tilstanden til modellen blir oppdatert når den gjør en prediction. Den vil da sammenligne dette med de forventede verdiene i valideringsdatasettet og bruke denne differansen som et estimat på feilgradienten. Feilgradienten blir så brukt til å oppdatere vektene til modellen. Altså desto mer eksempler modellen har under trening, desto presist vil dette estimatet bli. Da er det mer sannsynlig modellen yter bedre når vektene til nettverket blir justert. Motsatt vil færre eksempler per batch kan resultere i et mindre nøyaktig estimat av feilgradienten som også er avhengig på de spesifikke eksemplene brukt til trening. Dette vil igjen føre til upresise estimer, men kan resultere i raske re trening og en mer robust modell. Testingen utført i denne rapporten tilsa likevel at modellen trente raskere med batch size på 60 enn 16.



Figur 10: Visualisering av trening

Som tidligere presisert så har YOLOv3 byttet vekk litt av hastigheten mot bedre treffsikkerhet. Tiltross for dette proseserer den fortsatt 30fps [6], noe som anses som tilstrekkelig for sanntiddeteksjon. Arkitekturen er dog noe mer kompleks enn bare deteksjon, som igjen betyr at ytelse vil være dårligere. Vi opplevde å bruke mellom 0.1 og 0.2 -sekund per bilde/frame. Dette vil da tilsi mellom 5 og 10 fps, noe som er en del lavere en hva deteksjon med YOLOv3 bruker. Vi vil gå nærmere inn på årsaker til dette i 5.4.

Til tross for at resultatene ikke er banebrytende, har vi integrert YOLOv3 med en CNN for å lage et helhetlig system for trafikkskiltgjenkjenning. Resultatene kan sies å være gode, basert på tidligere arbeid beskrevet i seksjon 2.

5 Diskusjon

I dette prosjektet er det blitt gjennomført ulike tester og undersøkelser for å finne frem til et fungerende og pålitelig system for trafikkskiltgjenkjenning ved hjelp av YOLOv3 og konvolusjonelle nevrale nettverk. Disse utgjør viktige komponenter innenfor fagfeltet datasyn og forsøker å etterstrebe menneskelig syn. Mennesker har den fordel at de kan se en gjenstand og automatisk lære å gjenkjenne gjenstanden uansett om den varierer i størrelse, farge, plassering og lignende. Maskiner fokuserer derimot mer på deskriptive særtrekk som er gjengående i bilder ved trening. Med bakgrunn i dette skal denne rapportens resultater tolkes og sammenlignes for å se nærmere på hva som utgjør en god løsning for trafikkskiltgjenkjenning.

Underveis har det blitt erfart at fagområdet datasyn er

Tabell 2: Resultater fra klassifisering på GTSRB

Metode	Størrelse på input	Ant. parametre	Nøyaktighet
Asymmetric Kernels [7]	48 x 48	2.92M	99.66%
STN [18]	48 x 48	14.6M	99.71%
Committee of CNNs [20]	48 x 48	90M	99.15%
Hinge Loss Trained CNN [21]	47 x 47	1.16M	99.65%
Menneskelig ytelse [29]	-	-	98.84%
Vår	48 x 48	2.92M	99.36%

Tabell 3: Sammenlikning av iterasjoner av CNN-strukturer på GTSRB med og uten data-augmentering.

Metode	Data-augmentering	Ant. parametre	Nøyaktighet
Symmetriske filtre 3 x Conv	Nei	124,715	92.40%
Symmetriske filtre 4 x Conv	Ja	124,715	97.0%
Asymmetriske filtre - 4 grener m/1 x Conv ¹	Nei	3.1M	98.66%
Asymmetriske filtre - 4 grener m/1 x Conv ¹	Ja	3.1M	98.93%
Asymmetriske filtre - 4 grener m/2 x Conv	Nei	3.1M	98.97%
Asymmetriske filtre - 4 grener m/2 x Conv	Ja	3.1M	99.06%
Endelig versjon	Nei	3.02M	98.99%
Endelig versjon	Ja	3.02M	99.12%

¹ Ett enkelt konvolusjonslag med filterstørrelse på enten n x 1 eller 1 x n.

svært omfattende, noe som betyr at omfanget på prosjektet blir begrenset. Det ble tidlig i prosjektet bestemt seg for å holde seg til et system bestående av en detektor og en klassifiser. Selv om det antageligvis finnes bedre systemer enn det presentert her, ble det nødvendig å følge et forhåndsdefinert omfang som ble avgjort sammen med veileder.

5.1 Datasett

5.1.1 GTSRB

I og med at dette datasettet var spesielt ujevnt fordelt på de 43 klassene, ble klassifikatoren trent både med og uten balansering av datasettet ved hjelp av datautvidelser. Her kommer det tydelig frem at modellen oppnår bedre resultater ved bruk av datasett som inneholder mer data, samtidig som det er en jevnere fordeling mellom klassene. Dette er i tråd med en generell oppfattelse om at mer data å trenne på gir bedre klassifisering. På denne måten unngår vi også partisk klassifisering til en viss grad, da det ble erfart at modeller trent på et balansert datasett oftere klarte å skille på liknende skiltter. Et godt eksempel på dette er at modeller trent på det originale datasettet oftere klassifiserte et 60km/h-skilt feil og oppga disse som 80km/h-skilt. Dette er kritisk for pålitelige klassifikatorer i praktiske anvendelser.

Balanseringen av datasettet besto av å legge til utvidede versjoner av tilfeldig valgte bilder fra underrepresenterte klasser. Følgelig er det mulig for at dette kan ha hatt innvirkninger i nøyaktigheten modellen presterer. I tillegg ble observert at dette enkelte ganger førte til bedre nøyaktighet uten utvidelser, men det er usikkert om dette var påvirket av andre faktorer. Denne ulikheten ble betraktelig redusert og resultatene var mer konsistente ved flere epoker. Likevel ble det besluttet å oppgi resultatene og sammenligne disse, da det var forskjellene fra ulike kjøringer var minimale og det ble observert tydelige tendenser i resultatene.

Denne ulikheten ble betraktelig redusert og resultatene var mer konsistente ved flere epoker. Likevel ble det besluttet å oppgi resultatene og sammenligne disse, da det var forskjellene fra ulike kjøringer var minimale og det ble observert tydelige tendenser i resultatene.

5.1.2 GTSDB

Datasettet spiller en stor rolle i det endelige sluttresultatet da deteksjonsmodellen avhenger av tilstrekkelig mengder med data til trening. Under trening av deteksjonsmodellen så ble det benyttet en versjon av datasettet som var satt opp for trening ved bruk av YOLO. Siden datasettet var ferdig satt opp til formålet i dette prosjektet så ble det ikke gjort særlig mange erfaringer rundt anvending av datasettet.

5.2 Trafikkskiltdetektor

Valget av objekt-deteksjons algoritme var ikke noe som var naturlig, da tidligere erfaring og kunnskap til feltet var svært begrenset. Det endelige valget ble YOLOv3, da algoritmen har vist gode resultater samtidig som hastigheten er overlegen sammenlignet med R-CNN, Fast R-CNN og Faster R-CNN. Underveis i oppgaven har det derimot blitt oppdaget at det finnes nyere algoritmer som skal tilby enda bedre presisjon kombinert med bedre ytelse. I retrospektiv er dette noe som burde blitt testet og inkludert så man kunne sammenligne ulike objekt-deteksjons algoritmer opp mot hverandre.

Underveis i prosjektet ble erfart at det eksisterer flere ulike kodebasar som kan benyttes til trening av YOLO, hvorav mange er utdaterte. Dette ble uheldigvis oppdaget dagen før,



Figur 11: Rett deteksjon og klassifikasjon - Deteksjonsmodell med 83.83% mAP + Klassifiseringsmodell med 99.12% presisjon



Figur 12: Rett deteksjon og klassifikasjon - Deteksjonsmodell med 97.22% mAP + Klassifiseringsmodell med 99.12% presisjon

noe som betyddet et begrensning i disponibel tid til testing med korrekt kodebase. I tillegg ble det erfart at dokumentasjonen var noe mangelfull og ikke veldokumentert sammenlignet med rammeverk som Keras.

En ulempe som ble erfart med modellen var at den var omfattende å trenne. Siden modellen ble trent på datasettet fra rapporten til Sichkar V.N. og Kolyubin S.A. [23], var det allerede tilgjengelige ressurser for å lettere komme igang. Det var derimot en begrensning på tilgang til maskinvare, s som gjorde treningen en del vanskeligere. I artikkelen ble det brukt Nvidia Tesla V100 skermkort med særlig god prosessingskraft, mens treningen her i utgangspunktet ble begrenset av en Intel i5 prosessor. Den siste dagen ble det benyttet et Nvidia GTX 1060 GPU, noe som gjorde det mulig å trenne bedre.

Dessverre ble det ikke nok tid til mer omfattende trening med den rette kodebasen, noe som resulterte i høyeste resultat var på 83.83% mAP. Sammenligner man det med rapporten som detektormodellen er basert på, så ser man de oppnådde 97.22% mAP ved 5700 iterasjoner. Det er derfor å anta at man kunne forventet en høyere presisjon ved enda flere epoker, enn de 4000 som ble kjørt i dette prosjektet.

Årsaken til at det ikke ble trennt i Kaggle, som tidligere har blitt benyttet, er fordi Darknet er skrevet i C++ mens Kaggle bare støtter bruk av Jupyter notebook. Deteksjon og klassifisering har derfor blitt utført med YOLOv3 modellen som ble trennt i forbindelse med dette prosjektet og modellen som ble benyttet i [23]. Vi erfarer da at deteksjon fungerer til sin hensikt, men faller til kort sammenlignet med modellen til Sichkar V.N. og Kolyubin S.A. ved utfordrende miljøer som illustrert ved figur 13 og figur 14.

5.3 Trafikkskiltklassifikator

Det eksisterer en myriade av arkitekturen for CNNs, hvor uavhengig forskning har kommet med gode resultater. Det er her blitt testet ut både simple og mer komplekse modeller.

Fra tabell 2 kan vi også se at det er oppnådd dårligere nøyaktighet enn [7]. Undertegnede har benyttet seg av samme optimaliseringsalgoritme, men differensierer i andre faktorer som kan få utslag i resultatene.

For det første kan dette skyldes forskjeller i datasettet mo-



Figur 13: Delvis feil deteksjon og klassifikasjon - Deteksjonsmodell med 83.83% mAP + Klassifiseringsmodell med 99.12% presisjon



Figur 14: Delvis rett deteksjon og klassifikasjon - Deteksjonsmodell med 97.22% mAP + Klassifiseringsmodell med 99.12% presisjon

dellen er trent på, da Li og Wang bygget opp dette datasettet ved å velge et tilfeldig bilde for hvert 30. bilde i treningsdatasettet for å bygge opp valideringsdatasettet. På denne måten endte de opp med et valideringsdatasett på 1307 trafikkskiltbilder blant totalen på 39 209 bilder, som vil tilsi en fordeling på 3% for valideringsdatasettet. I denne rapporten ble det derimot brukt en 80/20% fordeling mellom treningsdatasettet og valideringsdatasettet. Dette ble det heller ikke eksperimentert med, men basert på den empiriske erfaringen nevnt over kunne det ytet bedre resultater. Dette kan derimot øke risikoen for overfitting.

Samtidig ble det benyttet funksjonen `train_test_split` fra python-modulen `sklearn.model_selection` som deler datasettet brukt til trening og validering tilfeldig i to deler, nemlig trening- og valideringsdatasettet. Det er trolig at dette hadde minimal innvirkning i forskjellige kjøringer i testene enn de større forskjellene satt opp mot [7]. Det er dog observert at det er vanlig praksis å benytte seg av denne funksjonaliteten.

Under dette punktet faller også inn variansen i nettverkssarkituren, eller algoritmen. Det vil si hvor sensitiv den er til det spesifikke datasettet brukt under trening [30]. Resultatene kan indikere at algoritmen har en viss grad av varians og kunne vært testet ved å endre på hyperparametre eller øke størrelsen på datasettet. Sistnevnte forsøkte vi å gjøre ved utvidelser av dataene og selv om det resulterte i bedre nøyaktighet, kommer det fortsatt til kort sammenliknet med modellen denne rapporten baserer seg på.

For det andre kan man få forskjellige resultater når man kjører samme algoritmen på samme datasett grunnet naturen til læringsalgoritmen. Dette ble også erfart i implementasjonen i denne rapporten, da resultatene varierte fra kjøring til kjøring. For eksempel ble treningsdatasettet supplementert med tilfeldige bilder fra underrepresenterte trafikkskiltklasser, noe som kan forklare nettopp denne variasjonen både i undertegnedes tester og den i [7]. En mulig løsning på dette problemet kan være å legge til ekstra tilfeldighet, for eksempel tilfeldig initiale vektorer og tilfeldig omstokking av data for hver epoke. Det kunne for eksempel blitt brukt en pseudotilfeldig (pseudorandom) tallgenerator i oppdelingen av trenings- og valideringsdatasettet. På den andre siden kan dette igjen føre til nye problemer, som det å finne det optimale seed til generatoren.

En tredje forklarende faktor er at en kan få forskjellige resultater når samme algoritme kjøres på samme data på forskjellige maskiner. Sammenlignet med [7] bruker vi en nyere og bedre GPU til trening og evaluering og vi kan derfor si at det er lite sannsynlig at denne faktoren er av stor betydning for resultatene.

Fra visualiseringen av treningen til den endelige modellen i figur 10 fremkommer det en rask konvergens både for nøyaktigheten og loss på trenings- og valideringsdatasettet innen de første 2-5 epokene. Dette kan forklares med bruken av Batch Normalization og ReLU-aktivering etter hvert enkelt konvolusjonslag.

Som nevnt i seksjon 3.3 ble det gjort enkelte modifiseringer til den originale Inception-modulen. Dette inkluderte å redusere antall grener og bytte ut 1 x 1 konvolusjonslag med 7

x 7 asymmetriske filtre. Fordeler med 1 x 1 konvolusjoner er at de lærer mønstre på tvers av dybdekanalene, som bidrar til den helhetlige evnen til trekkuthenting til modellen, samt evnen til å redusere dimensjonene til inputen sendt gjennom Inception Modulen [24]. På den andre siden kan såpass små filtre indikere at lavnivå trekk ikke påvirker piksler i nærhet til hverandre, noe som ikke er tilfellet for trafikkskilt. Her vil formen og farger til en piksel være tett avhengig av de nærmeste, som danner det totale trafikkskiltet. Derfor ble det tatt i bruk 7 x 7 asymmetriske filtre for å lære de større trekkene i bildene. Læring av større særtrekk i bildene ble derfor vektlagt høyere i vurderingen med hensyn til praktiske anvendelser.

5.4 System

Ved sammensetting av deteksjon- og klassifikasjonmodellen så ble det erfart at ytelse ikke var like bra som forventet. Ytelsen til YOLOv3 er tidligere blitt dokumentert til å være på omtrent 30 fps [6], noe som ikke ble oppnådd i dette prosjektet. Høyeste målte ytelse var på 8 fps, noe som er betydelig lavere enn forventet ytelse. En viktig faktor i dette er oppløsningen på videoene som blir testet. Videoene som det har blitt testet med har en høyere oppløsning enn hva COCO datasettet, som har blitt benyttet som til å måle ytelsen til YOLOv3. COCO datasettet har oppløsning på 640x480, mens en av videoene som har blitt mye brukt til testing er på 1280 x 720. Dette er en stor forskjell i oppløsning og spiller antageligvis en stor rolle i ytelsen av den endelige modellen. Dessverre ble det ikke nok tid til å teste og sammenligne opp mot ulike oppløsninger, noe som ville et bedre utgangspunkt til å sammenligne ytelses forskjell på de ulike oppløsningene.

Den endelige løsningen består av to deler, deteksjon og klassifisering, og kan derfor delvis sammenlignes med regionbaseerte rammeverk. Formålet til YOLO var å eliminere tidsforbruket til disse rammeverkene ved å sørge for deteksjon og klassifikasjon i samme nettverk. Dette betyr at modellen vår ikke nødvendigvis tar for seg prinsippene til YOLO i den forstand at den er todelt. Det hadde derfor vært interessant å sammenligne vårt endelige system opp mot en enkel YOLO modell som er trent for å detektere og klassifisere trafikkskilt fra samme datasett.

For å redusere antall etiketter som blir vist på videoen etter klassifisering av enkeltrammer, blir minimum sannsynlighet for kandidater satt til 0.6. Dette kunne vært eksperimentert med og tatt beslutninger på hvilken grense som gir best resultater sett i sammenheng med den helhetlige gjenkjenningen. På grunn av tidsbegrensninger ble dette ikke gjennomført.

6 Konklusjon og fremtidig arbeid

I denne rapporten ble det presentert et forslag til et trafikk-gjenkjenningsystem ved bruk av en YOLOv3-basert detektor og en tilpasset CNN-basert klassifikator. Detektoren er i stand til å detektere alle klassene i trafikkskilt-kategoriene og producere korrekte avgrenningsbokser for et flertall av de detekterte skiltene. Enkelte skilt ble detektert og avrensningsboks produsert mer enn én gang, som etterlater rom for forbedring

både her og når det kommer til hastighet. Klassifikatoren av trafikkskilt er i stand til å pålitelig identifisere hvilken type skiltet er med høy nøyaktighet. Til sammen utgjør disse to komponentene både helheten i systemet for gjenkjennning av trafikkskilt som med videre arbeid kunne blitt forsøkt anvendt i praksis.

Når det gjelder fremtidig arbeid bør muligheten for samtidig deteksjon og klassifisering av trafikkskilt undersøkes nærmere. På denne måten kan en unngå å måtte ta i bruk et ekstra nettverk for å klassifisere trafikkskilt. Nyere versjoner av YOLOv3 eksisterer i skrivende stund, som YOLOv4, YOLOv5 og YOLOv6, som med fordel kunne vært benyttet som en utvidelse av dette arbeidet.

Vår endelige CNN er relativt stort, på 39.6 MB, og det kan tenkes at et mindre nettverk med fordel kan utforskes til bruk i mobile- og bærbare enheter til bruk i intelligente systemer. Disse kan så evalueres ved å utplasseres i disse, som har begrenset prosesseringskraft sammenlignet med PC'er. Den endelige størrelsen på detektor modellen er på 246 MB, noe som også kan anses som suboptimalt for mobile enheter.

Annet fremtidig arbeid kan inkludere å trenne på nye observasjoner underveis i anvendelser for å kontinuerlig forbedre ytelsen til modellene.

Avslutningsvis vil takke Martin Johannes Nilsen som har veiledet oss gjennom hele prosjektet og stilt seg disponibel til å hjelpe ved alle døgnets tider.

Referanser

- [1] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2013.
- [2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," in *The 2011 International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.
- [3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.
- [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.
- [5] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.
- [6] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.
- [7] J. Li and Z. Wang, "Real-time traffic sign recognition based on efficient cnns in the wild," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 975–984, 2019.
- [8] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 2022–2031, 2016.
- [9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.
- [10] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.
- [11] R. Girshick, "Fast r-cnn," 2015.
- [12] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [13] Z. Zuo, K. Yu, Q. Zhou, X. Wang, and T. Li, "Traffic signs detection based on faster r-cnn," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 286–288, 2017.
- [14] S. P. Rajendran, L. Shine, R. Pradeep, and S. Vijayaraghavan, "Real-time traffic sign recognition using yolov3 based detector," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2019.
- [15] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras, "Road-sign detection and recognition based on support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 264–278, 2007.
- [16] K. Lu, Z. Ding, and S. Ge, "Sparse-representation-based graph embedding for traffic sign recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1515–1524, 2012.
- [17] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *The 2011 International Joint Conference on Neural Networks*, pp. 2809–2813, 2011.
- [18] Álvaro Arcos-García, J. A. Álvarez García, and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Networks*, vol. 99, pp. 158–165, 2018.
- [19] T. L. Yuan, "GTSRB_Keras_STN." https://github.com/hello2all/GTSRB_Keras_STN, 2017. Hentet 23. november 2021.
- [20] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *The 2011 International Joint Conference on Neural Networks*, pp. 1918–1921, 2011.
- [21] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1991–2000, 2014.

- [22] X. Ding, Y. Guo, G. Ding, and J. Han, “Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks,” in *Proceedings of the IEEE-E/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [23] V. Sichkar and S. Kolyubin, “Real time detection and classification of traffic signs based on yolo version 3 algorithm,” *Naučno-tehničeskij Vestnik Informacionnyh Tehnologij, Mehaniki i Optiki*, vol. 20, pp. 418–424, 05 2020.
- [24] “Deep learning: Understanding the inception module.” <https://towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652>. Accessed: 2021-11-27.
- [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2014.
- [26] S. Ruder, “An overview of gradient descent optimization algorithms,” 2017.
- [27] R. J. Tan, “Breaking down mean average precision (map).”
- [28] J. Hui, “map (mean average precision) for object detection,” Mar 2018.
- [29] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” *Neural Networks*, vol. 32, pp. 323–332, 2012. Selected Papers from IJCNN 2011.
- [30] “Why do i get different results each time in machine learning?.” <https://machinelearningmastery.com/different-results-each-time-in-machine-learning/>. Accessed: 2021-11-27.