



BACHELORPROEF VOORGEDRAGEN TOT HET
BEHALEN VAN DE GRAAD VAN BACHELOR IN DE
INFORMATICA

Object Tracking met Unsupervised Learning

Ilias Bachiri

Promotor

Dr. Philippe BEKAERT

Academiejaar 2017-2018

Abstract

Van AI leren auto's te besturen tot de beste Go-spelers onder ons verslaan. Van het afleiden of er fraude wordt gepleegd bij online advertenties tot het herkennen van gesproken taal.[3, 8, 17] Dit zijn een paar nieuwe gebieden die Machine Learning ons heeft kunnen bezorgen.

Er zijn echter niet enkel onderzoeksgebieden bijgekomen, maar ook verbeterd. Een voorbeeld hiervan is tracking, het detecteren en volgen van een bepaald object in een video. Er zijn verscheidene neurale netwerken die dit kunnen realiseren: YOLO, ROLO... Dit zijn echter meestal supervised learning algoritmen en vergen dus een groot aantal gelabelde data om het model te trainen.

Unsupervised algoritmen voorkomen dit door verborgen structuren te zoeken in ongelabelde data. Deze algoritmen zijn echter nog sterk in onderzoek en bijgevolg is er niet een specifiek algoritme waarvoor wordt geopteerd voor tracking. In wat volgt bespreken we een voorgestelde methode om unsupervised learning toe te passen op tracking.

Inhoudsopgave

1 Inleiding	5
1.1 Probleemstelling	5
1.2 Machine Learning	6
1.3 Unsupervised Learning	6
1.4 Doel	7
1.5 Toepassingen	8
1.6 Overzicht	8
2 Machine Learning	9
2.1 Oorsprong, methode en doel	9
2.1.1 Oorsprong	9
2.1.2 Methode	10
2.1.3 Doel	11
2.2 Klasse van problemen	12
2.2.1 Regressie	12

2.2.2	Classificatie	13
2.3	Types van Machine Learning	13
2.3.1	Supervised Learning	14
2.3.2	Unsupervised Learning	14
2.3.3	Reinforcement Learning	14
2.4	Evaluatie methoden	15
2.4.1	Cross-validation	15
2.4.2	Precision en Recall	15
2.4.3	ROC curve	17
2.4.4	Confusion matrix	18
3	Unsupervised Learning	20
3.1	Inleiding	20
3.2	Clustering	20
3.3	K-means clustering	22
3.4	Evaluatie	23
3.4.1	Purity	23
3.4.2	Rand Index	24
4	Artificiële Neurale Netwerken	25
4.1	Principe	25
4.1.1	Perceptron	25

4.1.2	Training	26
4.2	Convolutionele Neurale Netwerken	27
4.2.1	Convolutionele laag	27
4.2.2	Pooling laag	28
4.2.3	Architecturen	29
4.3	Autoencoders	30
5	Unsupervised Tracking: Eigen contributie	32
5.1	Principe	32
5.2	Training	33
5.3	Experimentele resultaten	35
6	Conclusie	39

Hoofdstuk 1

Inleiding

1.1 Probleemstelling

Tracking is een interessant gebied dat terug kan gevonden worden in verschillende applicaties. Neem bijvoorbeeld de tracking van een voetbal in belangrijke wedstrijden (zoals het wereldkampioenschap) om interessante statistieken bij te houden, scheidsrechters te helpen...[19]

Er zijn verschillende algoritmen bedacht om tracking af te handelen. Zo is er mean-shift tracking, contour tracking...[29, 31] Het probleem is echter vaak dat deze algoritmen niet kunnen omgaan met onnatuurlijkheden in een gegeven video en bijgevolg verbeterd moeten worden voor deze fouten.

Hier is waar Machine Learning van te pas komt. Zoals vermeld zijn er variabelen die problemen kunnen vormen wanneer we aan tracking willen doen. Machine Learning laat ons echter toe om de variabelen te laten afhandelen door algoritmen aan de hand van voorafgaande data. De data wordt door het algoritme gebruikt om te leren hoe het aan tracking kan doen op een efficiënte manier.[32]

Jammer genoeg introduceert dit een nieuw probleem: goed gelabelde en representatieve data. Het verzamelen van data is al wat minder een probleem dan vroeger. Desalniettemin is het labelen wel nog steeds een complicatie

waar men op moet letten.[25] De labels zijn namelijk hetgeen waar het algoritme zich op moet baseren. Slechte labels kunnen dus een negatief effect hebben.

Er is niettemin een manier om labels te vermijden: Unsupervised Learning. Deze manier van Machine Learning wordt echter nog onderzocht. Het doel van deze thesis is het onderzoeken naar een methode die dit mogelijk maakt en evalueren hoe bruikbaar deze methode is.

1.2 Machine Learning

Machine Learning is een discipline binnen AI, waarbij data wordt gebruikt om een uitspraak te doen over een bepaald gegeven. Neem als voorbeeld dat we een systeem willen om de soort van een bloem te laten herkennen. De manier waarop Machine Learning dit realiseert, is door de belangrijkste elementen uit een gegeven te halen. Zo kan men op basis van de afmetingen, kleur... van miljoenen bloemen bepalen tot welke soort een gegeven bloem behoort.

Er zijn verschillende vormen van Machine Learning: Supervised, Unsupervised, Reinforcement, Semi-supervised... Al deze vormen van Machine Learning doen wel iets verschillend (Reinforcement maakt bijvoorbeeld gebruik van beloningen en afstraffingen om een algoritme iets te leren). Degene waar wij ons op zullen focussen is *Unsupervised Learning*.

1.3 Unsupervised Learning

Unsupervised Learning maakt geen gebruik van labels of aansporingen, zoals Supervised Learning en Reinforcement Learning respectievelijk gebruiken. In plaats daarvan proberen dit soort algoritmen onderliggende structuur te vinden in de data die men geeft (merk op dat dit de noodzaak van een goede hoeveelheid data niet doet verdwijnen).[6]

Een bekende toepassing hiervan is het groeperen van foto's van Google

Foto's.[7, 13] Wanneer men afbeeldingen upload naar Google Foto's worden deze achterliggend aan de hand van gelijkaardige elementen gegroepeerd. Zo kunnen foto's waar de gebruiker in voorkomt in dezelfde groep gestoken worden. Op die manier krijgen alle afbeeldingen in de groep hetzelfde label, dankzij de gebruiker die één enkel gezicht labelt.

1.4 Doel

Er bestaan verschillende algoritmen om aan object tracking te doen in video's. Zo hebben we *Background Subtraction*, *Temporal Differencing*, *Optical Flow*... Deze methodes werken vooral op basis van hoe de tracking en detectie van het object is gedefinieerd. Dit geeft vooral problemen wanneer de visuele afhankelijkheden waarop het identificeren steunt sterke afwijkingen heeft. Zo kunnen *Occlusion*, *Clutter*, *Noise*... problemen veroorzaken, aangezien ze interferentie zorgen met de afhankelijkheden waarop de algoritmes op steunen.[23]

Machine Learning is een principe dat deze problemen op een relatief slimme wijze heeft weten te verbeteren. De reden hiervoor is dat we het systeem laten leren welke data relevant is om ons doel te bereiken in plaats van zelf hier regels voor te verzinnen.

Men heeft enkel een grote hoeveelheid data nodig om een machine learning model te trainen en men kan correcte regressie of classificatie uitvoeren op nieuwe data. Dit kan onder andere gedaan worden met *Supervised Learning*. Op hoog niveau werkt dit door data die al gekend is te geven aan het algoritme en de output te laten vergelijken met het verwachte antwoord.

Deze benadering bracht echter een ander probleem met zich mee. Het aantal gelabelde data dat nodig is om een zeer grote precisie te halen kan in sommige situaties hoog oplopen. Dit maakt Unsupervised Learning een interessant domein voor onderzoek, aangezien Unsupervised Learning geen nood heeft aan labels (wel nog steeds data).

Op vlak van tracking brengt dit echter een ander probleem teweeg: feature extraction.[14] Neem bijvoorbeeld een giraffe in een video. Om aan tracking

te kunnen doen moet men goede features hebben, opdat niet een vlek van de giraffe wordt gegroepeerd, maar de volledige giraffe. Deze thesis stelt een methode voor om dit soort video tracking te realiseren.

1.5 Toepassingen

Toepassingen liggen vooral in gebieden waar men aan tracking wil doen, maar het labelen van video's moeilijk is. Een voorbeeld is een winkel die het koopgedrag van mensen wilt nagaan. Met een unsupervised tracking systeem zou het mogelijk zijn om terugkerende bezoekers te groeperen en hun koopgedrag te identificeren aan de hand van welk gangpad ze bezoeken in de winkel. Deze data kan onder andere handig zijn om bepaalde producten anders te ordenen, zodat kopers ze zeker passeren en misschien kopen.

1.6 Overzicht

Hoofdstuk 2 handelt over algemene principes in Machine learning en vaak gebruikte evaluatie methoden. Hoofdstuk 3 bespreekt wat dieper de methodes die worden gebruikt voor Unsupervised learning en belangrijke principes rond deze vorm van learning. Hoofdstuk 4 bespreekt op hoog niveau de werking van neurale netwerken uit die gebruikt worden in de voorgestelde methode (de lezer kan hoofdstuk 2, 3 en 4 overslaan indien men al met deze principes vertrouwt is). Hoofdstuk 5 bespreekt de nieuwe voorgestelde unsupervised tracking methode en legt de algemene werking uit en in hoofdstuk 6 staat de conclusie over het hele verhaal rond unsupervised tracking.

Hoofdstuk 2

Machine Learning

2.1 Oorsprong, methode en doel

2.1.1 Oorsprong

De term "Machine Learning" werd voor het eerst gebruikt in 1959 door Arthur Samuel.[11] Alhoewel deze term al langer dan vandaag bestaat, is het pas recent dat deze vorm van Artificial Intelligence voor verscheidene toepassingen gebruikt wordt: AlphaGo, Stock Prediction...

Een definitie voor Machine Learning gaat als volgt: een machine learning algoritme leert uit ervaring E als de performantie P van taken T met E omhoog gaat. Zowel de ervaring als de performantie-maat hangt af van wat taak T precies inhoudt. Zo kan de taak "Het sorteren van auto's hun kleuren" met ervaring "Miljoenen foto's van auto's" als performantie-maat "Het aantal juist geïdentificeerde auto's" hebben.

2.1.2 Methode

Een machine learning algoritme gebruiken neemt meestal bepaalde stappen aan. Eerst en vooral zal men data moeten verzamelen. In de meeste gevallen zal een grotere hoeveelheid aan data de predictie correcter laten verlopen dus men wenst vaak zoveel data als maar kan (dit is tegenwoordig niet meer zo een probleem als vroeger, aangezien er enorm veel datasets op het internet zijn te vinden op sites zoals Kaggle).¹

De volgende stap is deze data voorbereiden voor training. Het kan zich namelijk voordoen dat er data is in de dataset die het algoritme fout kan beïnvloeden. Zo kunnen uitschieters die zich ver van het gemiddelde bevinden best verwijderd of gelimiteerd worden door een bovengrens. In classificatieproblemen kunnen alle klassen best ook evenveel data krijgen om bias te voorkomen (tenzij er daadwerkelijk meer van een bepaalde klasse voorkomt dan een andere).

De derde stap is het kiezen van een model. Een model kan men zien als het hoofdgedeelte dat de data gaat behandelen. Er bestaan verschillende modellen en meeste modellen zijn meer geschikt dan andere in bepaalde situaties. Een model kiest men dus afhankelijk van het probleem dat men wil oplossen.

Decision trees zijn bijvoorbeeld toe te passen op zowel classificatie- als regressie problemen. Een groot nadeel van decision trees is echter dat ze niet kunnen geüpdatet worden met nieuwe data. Als we bijvoorbeeld een decision tree hebben die spam kan classificeren op basis van gebruikte woorden en er komt ineens een nieuw woord vaak voor in spam, dan moet men een nieuwe decision tree trainen met de nieuwe data in plaats van de huidige tree incrementeel te updaten.

Na het model te kiezen, gaat men het gekozen model moeten trainen. Dit trainen kan men zeer letterlijk nemen. Stel dat we opnieuw een decision tree nemen om appels en bananen te classificeren. Deze decision tree weet in het begin compleet niet wat een appel of banaan is wanneer er één aan hem wordt gepresenteerd. We geven hem echter nu de voorbereide data waarvan

¹<https://www.kaggle.com>

hij nu gaat leren. Door fouten te maken en het interne model te verbeteren aan de hand van de data kan hij identificeren dat met een bepaalde zekerheid een geel object geklassificeerd kan worden als banaan en een rode als appel.

Achteraf kan men nog eens het getrainde model testen aan de hand van test-data. Dit doet men om inzichten op te doen over eventuele parameters waar men misschien niet had aan gedacht, fouten in de predictie vroegtijdig op te sporen... Hier zijn een aantal mogelijkheden voor: confusion matrices, ROC curves... Belangrijk hierbij is dat validatie niet hetzelfde is als testen. Bij validatie worden de parameters van het model nog eens extra aangepast om overfitting te voorkomen aan de hand van validatie-data die verschillend is van de test-data en training-data.

Soms is het mogelijk om een getrainde model nog te verbeteren door te doen aan parameter tuning. Bepaalde models gebruiken namelijk parameters die nodig zijn tijdens het trainen. Zo kan "hoe vaak de dataset wordt doorlopen" één maal, twee maal... gedaan worden. Goed gekozen parameters kunnen in sommige gevallen de accuraatheid van een predictie model in zekere mate doen stijgen.

2.1.3 Doel

Machine Learning is vooral interessant voor problemen waar het programmeren van bepaalde herkenningsregels een relatief moeilijke taak wordt. Neem dat we bijvoorbeeld een spam filter willen maken voor e-mails. Spam heeft bijvoorbeeld bepaald woorden die gebruikt worden ("Nigeriaanse prins"), bepaalde namen die vaak gebruikt worden, afzenders...

Deze regels kunnen vrij ver gaan en voor iedere regel moet men testen of deze voldaan is om te weten of de e-mail spam was. Niet enkel het programmeren maar ook het onderhouden van deze code zou een karwei zijn. Neem namelijk dat de volgende dag een hele nieuwe trend zich voordoet: "Afghaanse prinsessen", de stijl van schrijven wordt aangepast... In deze gevallen zouden de vele complexe regels aangepast worden wat soms vrij moeilijk kan blijken.

Een Machine Learning algoritme pakt dit veel slimmer aan: het algoritme

wordt getraind tegen voorbereide data en leert hier automatisch patronen uit (het kan zelfs zijn dat de analist hier zelf nieuwe inzichten in op doet). Dit vermindert de nood aan complexe regels die moeten bedacht worden. Ook kunnen sommige algoritmen geüpdatet worden tegen nieuwe data om hier nieuwe trends uit te leren en het onderhouden een gemakkelijke zaak te maken. Deze algoritmen kunnen ook problemen aan waar er niet echt algoritmen voor aangewend kunnen worden zoals spraak detectie.[9]

2.2 Klasse van problemen

2.2.1 Regressie

Regressie beslaat de klasse van problemen waarbij een waarde moet geschat worden voor een bepaald element. Stel dat we huizen verkopen en we zijn niet zeker hoeveel een nieuw huis dat we willen verkopen ongeveer zou moeten kosten. Een Machine Learning algoritme zou dan met vorige huizen en hun prijzen een uitspraak kunnen doen over het huis dat men wil verkopen. Zo zou het algoritme misschien kunnen leren dat een hoger aantal badkamers meestal in correlatie staat met een hogere prijs.

Regressie wordt meestal gedaan door een bepaalde functie af te stellen op basis van gelabelde data. Neem bijvoorbeeld een eerstegraads-functie met voorschrift $ax+b$. Met de data kan dan door het Machine Learning algoritme nagegaan worden wat de richtingscoëfficiënt is, waar de y-as snijdt...

Bij een predictie moet men dan enkel het functievoorschrift invullen om de overeenkomstige waarde te krijgen. Het functievoorschrift hoeft echter niet altijd een eerstegraads te zijn en kan even complex zijn als de data is. Als het voorschrift echter te complex is, kan dit echter voor problemen zorgen en wordt in de subsectie *overfitting* besproken. Tevens is een te simpele curve problematisch aangezien deze dan vaak ver buiten hun foutenmarge zal gepresterd worden. [18]

Net zoals bij classificatie bestaan er verschillende toepassingen voor regressie. Zo kan men aan het voorspellen van koersprijzen doen aan de hand

van Machine Learning. Het mooie hieraan is dat deze algoritmen kunnen gedefinieerd worden om wijzigingen op beurzen te anticiperen en hun model hiervoor aan te passen.[28]

2.2.2 Classificatie

Classificatie problemen zijn problemen waarbij men een bepaald element heeft en deze moet classificeren tot een bepaalde groep. Zo zou men bijvoorbeeld spam kunnen filteren aan de hand van gelabelde spam en ham (echte e-mails). Classificatie is ook nuttig voor extreem kritische zaken zoals het tijdig detecteren van tumoren (iets wat dokters grotendeels op het oog moeten doen en dus foutgevoelig).[2]

Classifiers zijn relatief gevarieerd afhankelijk van wat ze moeten doen. Zo bestaan er, naast het spam en ham voorbeeld, classifiers die tussen meerdere klassen moeten onderscheiden: multiclass classifiers. Dit kan bijvoorbeeld het geval zijn wanneer men een stuk fruit wil identificeren (appel, peer, banaan...).

Wanneer men meerdere labels moet toekennen met een classifier (bijvoorbeeld meerdere mensen in één foto) spreekt men van multilabel classification. Gelijkaardig hieraan is multioutput classification, waarbij een label meerdere waardes kan hebben (bijvoorbeeld wanneer men een foto indeelt in elke pixel en elke pixel een bepaalde intensiteit heeft).

Een groot probleem bij classificatie-algoritmen is accuraatheid. Om honderd procent accuraatheid te bereiken is er vaak bijzonder veel invoerdata vereist.[16]

2.3 Types van Machine Learning

Algemeen onderscheid men drie types learning: unsupervised, supervised en reinforcement (deze onderscheiding is echter niet altijd van toepassing; zo bestaat er ook semi-supervised learning). Welk type gebruikt moet worden

hangt af van welk probleem er precies mee opgelost moet worden.[21]

2.3.1 Supervised Learning

Supervised Learning is veruit de meest gebruikte vorm van learning. Het concept is relatief simpel: het algoritme krijgt een dataset waarvan de uitkomst al bekend is en leert hieruit de functie die gebruikt moet worden voor verdere predictie. Het doel van dit soort learning is dus een label aanbrengen op nieuwe data op basis van vroegere data. Deze soort van learning heeft vooral problemen met het aantal labels dat moet worden voorzien aan de data.

2.3.2 Unsupervised Learning

Clustering of Unsupervised Learning is relatief minder gemakkelijk te begrijpen in vergelijking met Supervised Learning. Het idee is om een onderliggend patroon te vinden in de data die gebruikt kan worden om nieuwe data te beoordelen op basis van dit patroon. Zo zou een Unsupervised Learning algoritme in staat zijn om verschillende soorten van bestek te groeperen zonder dat dit algoritme een besef heeft van wat precies wat is. Meer hierover volgt in hoofdstuk 3.

2.3.3 Reinforcement Learning

Reinforcement Learning is in een zekere zin een vorm van learning die wij als men ook kennen. Wanneer het algoritme iets correct doet krijgt hij een beloning en als niet wordt hij afgestraft. Het doel van het algoritme is dan om zijn strategie zodanig aan te passen dat hij altijd zal beloont worden. De AlphaGo AI van Google maakte hier gebruik van om een zeer groot aantal keren tegen zichzelf te spelen, het spel Go te leren en de Europese kampioen in Go te verslaan.[24]

2.4 Evaluatie methoden

Een belangrijk proces na het implementeren van een machine learning algoritme is evalueren hoe goed het algoritme zijn werk doet. Wat hieronder volgt zijn een paar van de methodes die men kan toewenden om een algoritme te evalueren (deze methoden zijn vooral van toepassing op supervised learning).

2.4.1 Cross-validation

Cross-validation is een methode die vaak gebruikt wordt om accuraatheid van een model na te gaan. Er bestaan verschillende varianten op cross-validation, maar het principe is in de meeste gevallen gelijkaardig: splits de trainingsdata op in trainingsdata en validatiedata. De validatiedata zal dan gebruikt worden om te kijken hoe accuraat het model is.

Het opsplitsen van de trainingsdata betekent echter dat men minder data heeft om mee te trainen en kan het model dus beïnvloeden. Varianten zoals *K-fold cross-validation* lossen dit op door K aantal training- en validatiedata groepen te halen uit de trainingsdata.[27]

Deze methode is echter problematisch wanneer men een dataset heeft die niet zoals gewenst is. Wanneer er namelijk een dataset van 90 gele auto's en 10 rode auto's is en het model gokt in elke instantie "gele auto", dan is er een accuraatheid van 90 percent (ook al gokt het model constant geel).

2.4.2 Precision en Recall

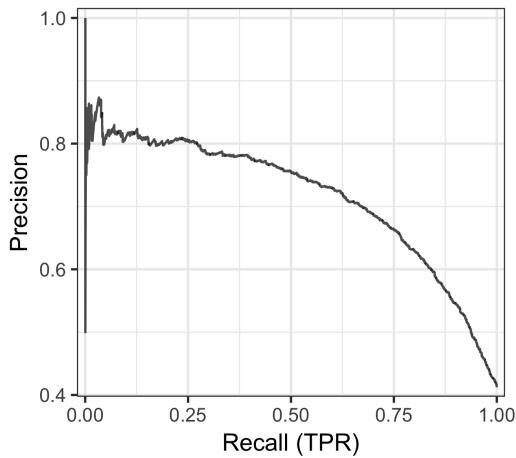
Precision en Recall zijn twee termen die praktisch lijnrecht tegenover elkaar staan en het toenemen van één heeft meestal als gevolg dat de andere zal afnemen. Recall is de maat voor "True Positives", oftewel hoeveel van de elementen die tot een bepaalde groep hoorden ook daadwerkelijk juist werden aangeduid. Als er bijvoorbeeld acht gele auto's zijn en tien rode auto's zijn en ons model kan vier gele auto's correct identificeren, dan is de recall voor

de gele auto's $4/8$.

Precision, daarentegen, duid aan hoeveel van de geïdentificeerde elementen ook daadwerkelijk juist zijn. Stel we nemen hetzelfde voorbeeld als hierboven, maar voegen tevens toe dat twee rode auto's fout werden geïdentificeerd als een gele auto. Hierdoor bekomen we $4/(4+2)$, oftewel $4/6$ voor onze precision.[15]

De tradeoff die tussen de twee voorkomt, is vooral een gevolg van hoe zeker men wil zijn van iets. Stel dat men met een hoge zekerheid een gele auto wil identificeren. Dan kunnen we beslissen dat enkel bij 90% zekerheid een auto als geel wordt aangeduid. Hierdoor zal de precisie zal dan stijgen omdat er bijna geen fouten worden gemaakt. Er zal echter een grote groep gele auto's zijn die niet werd herkend als gele auto en bijgevolg zal de recall omlaag gaan. Dit werkt vice versa wanneer men een lage zekerheid neemt.

Men kan precision en recall ook afbeelden aan de hand van een precision/recall curve.

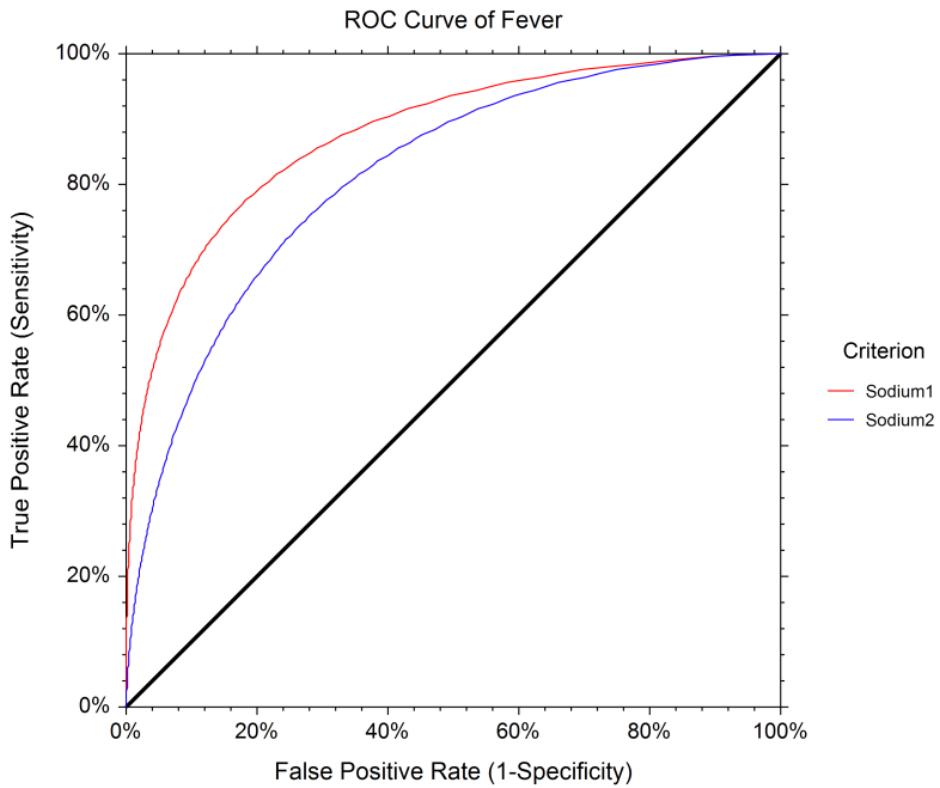


Figuur 2.1: Voorbeeld van een precision/recall curve

2.4.3 ROC curve

Een ROC curve (Receiver Operating Characteristic curve) is gelijkaardig aan een precision/recall curve. Een ROC curve beeldt echter correcte classificaties af op foute classificaties. Net zoals bij precision/recall is er bij ROC ook een inherente tradeoff. Wanneer men namelijk meer correcte classificaties wilt, is het gevolg vaak dat er meer foute classificaties opduiken.

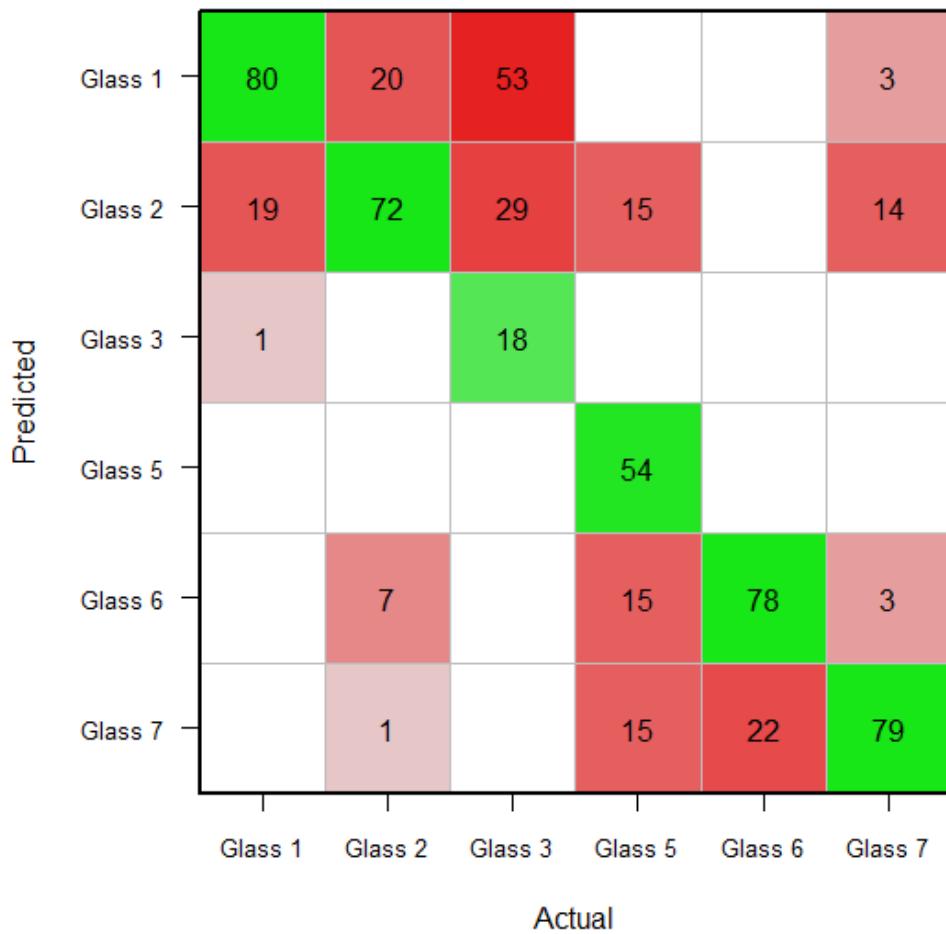
Wanneer men een ROC curve plot, zet men soms een diagonale lijn van linksonder naar rechtsboven. Dit is de ROC curve van een puur willekeurige classifier. Hoe verder van de diagonaal de ROC curve is, hoe beter het model zal presteren. Dit meet men soms met AUC (Area Under the Curve). Een perfecte classifier heeft een AUC van één en een willekeurige 0,5 (dit kan men zien omdat de curve net zoals de diagonaal het vierkant in twee deelt).[4]



Figuur 2.2: Voorbeeld van een ROC curve

2.4.4 Confusion matrix

Confusion matices kunnen vooral gebruikt worden voor classificatie problemen. Het principe van een confusion matrix is dat men alle instanties telt dat een klasse X werd geclasseerd als een klasse Y. Deze kunnen dan aan de hand van een matrix afgebeeld worden om bepaalde inzichten op te doen. Hoe hoger de cijfers in de middelste diagonaal, hoe beter de classifier presteert. Typisch gebruikt men ook kleur intensiteiten om snel te kunnen zien hoe de classificaties gebeuren (een hogere intensiteit is typisch een hogere classificatie).[5]



Figuur 2.3: Voorbeeld van een confusion matrix

Hoofdstuk 3

Unsupervised Learning

3.1 Inleiding

Unsupervised Learning is een type Machine Learning dat een fundamenteel verschil heeft ten opzichte van Supervised Learning: de gegeven dataset heeft geen labels ter beschikking om uit te kunnen leren. Ook het doel is typisch anders dan bij Supervised Learning. Typisch gezien bij Unsupervised Learning is er geen weet van hoeveel klassen er zijn. Dit kan namelijk het probleem dat we willen oplossen.

Dit heeft vele interessante toepassingen zoals het groeperen van klanten in een webwinkel voor het gericht adverteren van bepaalde producten die worden gekocht in de webwinkel (kopers zouden bijvoorbeeld dan producten te zien krijgen die andere mensen in hun groep hebben gekocht).

3.2 Clustering

Twee relatief bekende Unsupervised Learning taken zijn *Clustering* en *Dimensionality Reduction*. Clustering is het groeperen van ongelabelde data, terwijl Dimensionality Reduction gebruikt wordt om een compressie uit te

voeren op een grote hoeveelheid data zonder dat de structuur verandert. Er zijn nog andere taken zoals Anomaly Detection, maar om niet uit te lopen zullen we enkel kijken naar clustering.

In een clustering algoritme wordt altijd gezocht naar "gelijkaardigheid" tussen meerdere objecten. Deze gelijkaardigheid is gebaseerd op factoren die afhangen van het type dataset. Zo kan het in een dataset van honden misschien interessant zijn om gelijke rassen te vinden op basis van de hoogte.

Deze gelijkaardigheid meet men met *Distance Measure*. Een Distance Measure bepaalt in hoeverre twee objecten gelijkaardig zijn. Als we bijvoorbeeld katten en honden willen groeperen, hebben een golden retriever en Perzische kat een lagere Distance Measure dan een golden retriever en een Duitse herder. Hoe deze Distance Measure wordt bepaald hangt grotendeels af van welk algoritme men gebruikt en wat voor dataset gebruikt wordt.

De Distance Measure zal echter wel aan eigenschappen moeten voldoen om correct te kunnen werken. Er moet allereerst symmetrie tussen gelijkaardige objecten zijn: Als object A op object B lijkt, dan geldt dit ook in de andere richting. Tevens moet een object altijd op zichzelf lijken. De afstand moet dus gelijk zijn aan nul. Dit heeft als gevolg dat objecten die een Distance Measure van nul hebben, gelijk zijn aan elkaar. Als laatste moet de driehoeksongelijkheid voldaan worden. Als object A lijkt op B en B op C, dan moet A ook op C lijken.

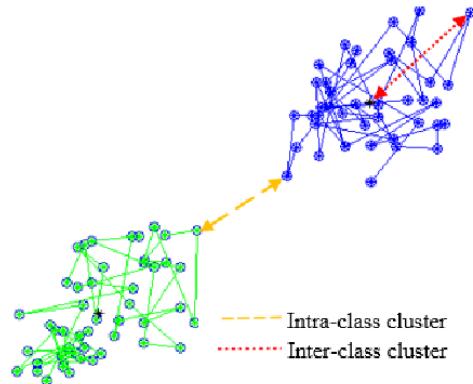
De drie relatief grootste vormen van clustering zijn hiërarchische clustering, partionele clustering en Bayesiaanse clustering. Hiërarchische clustering neemt eerder gevonden clusters en voert hier opnieuw clustering op uit. Deze vorm van clustering kan opgedeeld worden in twee vormen: agglomeratief; ieder element begint als een aparte cluster en gelijke clusters worden samengevoegd in iedere iteratie en verdelen; een grote cluster wordt verdeeld in kleinere en kleinere clusters tot een stop-conditie bereikt wordt.

Partionele clustering zoekt alle clusters in één keer op in tegenstelling tot hiërarchische clustering. Deze vorm van clustering kan ook gebruikt worden per iteratie in hiërarchische verdelende clustering. Tot slot gebruikt Bayesiaanse clustering posteriori distributies over de collectie van alle partities van de data om aan clustering te kunnen doen.[26]

3.3 K-means clustering

K-means clustering is een partitionele clustering methode voorgesteld in 1967 door MacQueen. Deze clustering methode heeft een relatief specifieke werking. De werking kan ingedeeld worden in twee fasen: het kiezen van de centra van de clusters en het berekenen tot welke clusters de input data behoort.

De eerste fase doet men op basis van de parameter K . K staat namelijk voor het aantal clusters dat men wil vormen uit de input data (vandaar de naam K-means). De eerste keer dat deze fase wordt uitgevoerd gebeurt door K willekeurige datapunten te kiezen die de centra representeren van de clusters. Deze centra worden in volgende iteraties aangepast om de intra-klassen cluster afstand zo klein mogelijk te maken, terwijl de inter-klassen cluster afstand zo hoog mogelijk wordt gemaakt.



Figuur 3.1: Visuele weergave van de twee afstanden

De tweede fase bestaat uit het toekennen van de clusters aan de overige datapunten. Dit wordt gedaan door de afstand te berekenen van ieder datapunt tot het centrum van iedere cluster. De cluster van het centrum met de kleinste afstand tot het datapunt is dan degene tot welke het datapunt dan behoort.

Zoals eerder aangegeven worden in de eerste iteratie de centra willekeurig

gekozen uit de datapunten. Dit zal er voor zorgen dat de eerste clusters veel te gevarieerd zullen zijn. Daarom worden in volgende iteraties in de eerste fase de nieuwe centra berekend door het gemiddelde te nemen over alle datapunten binnen de cluster.

$$c_i = \frac{1}{S_i} \sum_{x_i \in S_i} x_i$$

Het K-means algoritme brengt echter twee groten vragen met zich mee: wat moet de waarde van K zijn voor een bepaalde dataset en wat moeten de initiële centra zijn om de clusters zo puur mogelijk te maken? De centra moeten namelijk niet verplicht willekeurig gekozen worden. Men kan de centra ook zelf kiezen.

Echter voor beide vragen is tot heden geen eenduidig antwoord. De enige manier om zowel K als de beste cluster centra te vinden is door het algoritme meermaals uit te laten voeren op de dataset en te kijken welke waarden de beste resultaten leveren.[30]

3.4 Evaluatie

Hoewel het moeilijker is om bij unsupervised learning aan evaluatie te kunnen doen zijn er toch een aantal mogelijkheden om na te kijken of de gevormde de clusters zo dicht mogelijk liggen bij de gewenste oplossing. Hieronder wordt purity en de Rand Index besproken maar er zijn er nog meer: F-measure, Normalized Mutual Information, Jaccard Index...

3.4.1 Purity

Purity is mogelijk wanneer men gelabelde data heeft om mee te werken. Purity wordt berekend door iedere cluster de klasse toe te kennen die het meeste voorkomt binnen dezelfde cluster. Hierna kan dan de klasse binnen de cluster van een datapunt vergeleken met de klasse van het datapunt zelf. De formule is dus als volgt:

$$\frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d|$$

met M de verzameling van clusters en D de verzameling van klassen. Het probleem van deze maat is dat er geen rekening gehouden wordt met het aantal clusters. Een perfecte score kan dus behaald worden door simpelweg ieder datapunt zijn eigen cluster te geven.[22]

3.4.2 Rand Index

De Rand Index berekent hoe gelijkaardig clusters zijn vergeleken met de standaard classificatie ervan. Anders gezegd is de Rand index een maat voor het percentage van juiste beslissingen die gemaakt zijn door het algoritme en kan berekend worden op volgende wijze:

$$RI = \frac{TP + TN}{TF + TP + FN + TN}$$

waarbij TN het aantal true positives, TN true negatives, FP false positives en FN false negatives is.

De Rand Index weegt echter false positives en false negatives als even zwaar wat bij sommige applicaties ongewenst is (bijvoorbeeld een videosite voor kinderen waar een geschikte video die als ongeschikt wordt bestempeld beter is dan een ongeschikte video die als geschikt wordt bestempeld).[22]

Hoofdstuk 4

Artificiële Neurale Netwerken

4.1 Principe

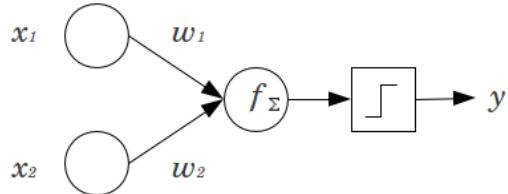
Artificiële neurale netwerken zijn ontstaan uit de neurofysiologie in 1943 en modelleren tot op een zeker niveau hoe een neuraal netwerk binnen het brein van de mens werkt. Voor een lange tijd werden deze netwerken verwaarloosd aangezien men er niet de grootse dingen mee kon doen die men dacht te kunnen doen. De laatste jaren is er echter weer een heropleving van de studie hiervan. Dit kan wellicht aan verschillende dingen te wijten zijn zoals grotere beschikbare datasets op het internet, grotere rekenkracht dankzij GPU's...

Een artificieel neuraal netwerk bestaat uit neuronen die met elkaar verbonden zijn. Elk neuron heeft één of meer inputs en één of meer outputs. De architectuur van het neurale netwerk bepaalt hoe de neuronen onderling verbonden zijn. De relatief eenvoudigste architectuur is de perceptron.

4.1.1 Perceptron

De perceptron heeft inputs waarbij iedere input afzonderlijk een gewicht met zich meedraagt. Deze gewichten worden met hun input vermenigvuldigd en met elkaar gesommeerd. Het resultaat is een gewogen som van de inputs.

Om de output te bepalen wordt deze gewogen som als input gebruikt van een activatie functie. Een activatie functie vertaalt de gewogen som naar een waarde die de output kan gebruiken. Een relatief veelgebruikte activatie functie voor perceptrons is de "step function".



Figuur 4.1: Visuele weergave van een perceptron

Deze enkele perceptron kan gebruikt worden voor binaire lineaire classificatie. Zo kan deze gebruikt om honden en katten te identificeren op basis van hun hoogte, gewicht... De perceptron zal echter niet meteen dit juist kunnen doen. Net zoals bij standaard machine learning algoritmes moet deze eerst getraind worden. Het trainen wordt gedaan door de gewichten van de inputs aan te passen.[20]

4.1.2 Training

Het correct aanpassen van de gewichten doet men door eerst het verlies te berekenen dat men heeft opgelopen vergeleken met de ideale uitkomst en daarna dit verlies te minimaliseren. Het verlies bereken hangt net zoals het minimaliseren af van het probleem dat men wenst op te lossen. Als men bijvoorbeeld probeert foto's te reconstrueren kan men de gereconstrueerde foto te vergelijken met de echtelijke foto en kijken hoe ver men hiervan af zit.

Voor het minimaliseren van het verlies gebruikt men typisch een optimizer die ieder gewicht van het netwerk afstelt om het verlies zo klein mogelijk te maken. Dit wordt intern gedaan aan de hand van gradiënt voor ieder gewicht waarmee men dan kan gaan zien of men dichter naar het minimum aan het gaan is of er verder van af.[20]

4.2 Convolutionele Neurale Netwerken

Convolutionele neurale netwerken zijn een vorm van neurale netwerken die vooral gebruikt worden in image recognition. Ze bestaan al sinds 1980 en hebben sindsdien hun plaats gevonden in zelfbesturende auto's, image search systemen... Ze kunnen zelfs gebruikt worden voor voice recognition en natural language processing.

Net zoals normale neurale netwerken vinden convolutionele neurale netwerken (CNN) hun oorsprong in de biologie. Men ontdekte dat in het menselijk brein neuronen enkel reageren op een bepaald gebied in hetgeen gezien wordt. Daarnaast kunnen de neuronen zelf ook ingedeeld worden in hoge niveau neuronen en laag niveau neuronen.

De lage niveau neuronen reageren op verschillende lage patronen zoals horizontale of verticale lijnen, terwijl de hoge niveau neuronen reageren op combinaties van de lage niveau neuronen om complexe patronen te begrijpen.[12]

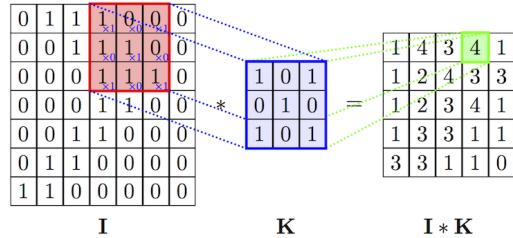
4.2.1 Convolutionele laag

Een convolutionele laag is een laag van neuronen die verbonden zijn met een specifiek gebied van bijvoorbeeld een foto (in plaats van de hele foto). Een tweede convolutionele laag doet dan hetzelfde maar dan op de eerste convolutionele laag, enz. Het nut hiervan is dat de eerste laag lagere features leert, terwijl de hogere lagen de complexere figuren hieruit leren.

Een neuron op rij i en kolom j is verbonden met output neuronen van de vorige laag op rijen i tot $i + f_h - 1$ en kolommen j tot $j + f_w - 1$, waarbij f_h en f_w respectievelijk verwijzen naar de hoogte en breedte van het veld dat een neuron moet onderzoeken (ook wel een receptive field genoemd). Om ervoor te zorgen dat een bovenste laag even groot is als de onderste, voegt men soms extra nullen rond de input. Dit noemt men zero padding.

Een kleine bovenlaag kan ook verbonden worden met een grotere onderlaag. Dit doet men door wat plaats te laten tussen de receptive fields. De

afstand tussen deze fields noemt men dan de stride. Deze stride kan voor hoogte en breedte verschillen.[12]



Figuur 4.2: Visuele weergave van een convolutionele laag

Filters

De gewichten binnen een CNN zien er iets anders uit dan bij een standaard artificieel neuraal netwerk. Deze kunnen namelijk afgebeeld worden als foto's en noemt men ook wel *filters* of *convolutionele kernels*. Dit kan bijvoorbeeld een zwarte foto zijn met één horizontale witte lijn door het midden. Als alle neuronen in het netwerk het vorige voorbeeld als gewicht nemen en we voeden het een bepaalde foto, dan zullen vooral witte horizontale lijnen in de foto duidelijker te zien zijn.[12]

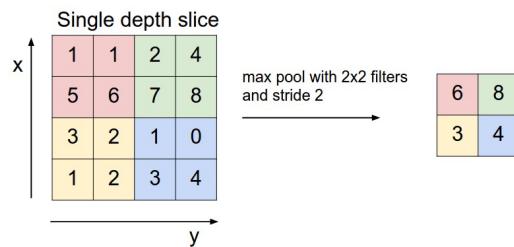
4.2.2 Pooling laag

Het doel van een pooling laag is subsampling van de input image om er voor te zorgen dat de computationele belasting op de convolutionele laag zo min mogelijk is. De architectuur van een pooling laag is gelijkaardig aan die van een convolutionele laag: hogere lagen zijn verbonden met lagere lagen om complexere features te extraheren.

Een groot verschil is echter dat pooling lagen geen gewichten hebben. In plaats daarvan wordt er gebruik gemaakt van een aggregatie functie. Deze

aggregatie functie kan gekozen worden afhankelijk van het probleem. Zo is er bijvoorbeeld het gemiddelde nemen of het maximum.

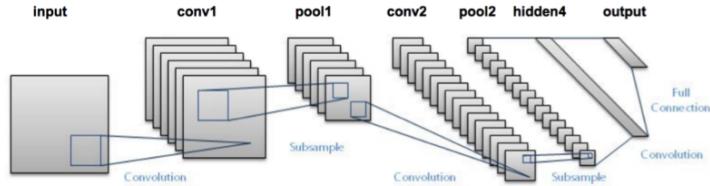
Pooling lagen zijn echter zeer destructief. Met een 2×2 kernel en een stride van 2 zullen de input waarden al verlagen met 75%. Eenmaal dat de pooling laag is toegepast op de input is het vrij moeilijk om nog de oude input te generen uit de output.[12]



Figuur 4.3: Voorbeeld van een pooling laag met een max functie, 2×2 filter en stride 2

4.2.3 Architecturen

Voor CNN's zijn er verschillende architecturen mogelijk die in de meeste gevallen combinaties zijn van convolutionele lagen gevolgd door convolutionele lagen. Het gebruik van aantal lagen, aggregatie functies, strides, kernels... is wat bepaalde architecturen beter doen presteren dan andere. Voorbeelden van architecturen zijn LeNet-5, AlexNet, GoogLeNet...

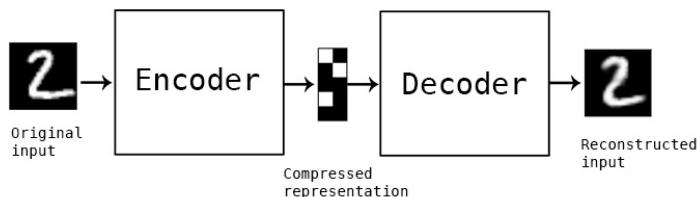


Figuur 4.4: Voorbeeld van een CNN architectuur

4.3 Autoencoders

Autoencoders zijn een soort van neuraal netwerk dat voor verschillende doelen gebruikt kan worden. Al deze doelen hebben echter één ding gemeen: de input leren omzetten naar een efficiëntere representatie, ook wel *codings* genoemd. Deze representatie wordt aangeleerd zonder supervision.

Een mogelijkheid om ze te gebruiken ligt in het reduceren van de dimensies van een gegeven input (dit vanwege voorgaande representatie). Deze kan dan alweer gebruikt worden om pre-training op een neuraal netwerk uit te voeren om resultaten te verbeteren. Ook kunnen ze voor generatieve netwerken gebruikt worden om bijvoorbeeld gezichten te genereren.



Figuur 4.5: Voorbeeld van een autoencoder

In hun absolute basis lijken autoencoders niets meer te doen dan de input te kopiëren naar de output. Dit is echter niet helemaal correct. De limiteringen die men plaatst op het netwerk van de autoencoder zal grotendeels het

gedrag ervan bepalen. Zo kan de autoencoder getraind worden om foto's te herstellen door de mapping te leren tussen geruïneerde foto's en hun originele versie.

Een autoencoder bestaat dus uit twee delen: een encoder en een decoder. De encoder zet de inputs om naar de interne representatie die men wenst en de decoder zet deze representatie dan op zijn beurt weer om naar outputs.[10]

Hoofdstuk 5

Unsupervised Tracking: Eigen contributie

5.1 Principe

Mijn voorstel om te kunnen doen aan unsupervised tracking ondergaat een paar stappen. Allereerst moet men een dataset beschikken die alle objecten bevat die men van plan is te tracken. In dit onderzoek heb ik gebruik gemaakt van CIFAR-10 ofwel CIFAR-10 aangevuld met een extra categorie van foto's.

CIFAR-10 is een dataset van 60000 foto's van 32 hoogte, 32 breedte en de RGB kleuren. De dataset bestaat uit 50000 foto's voor training en 10000 foto's om het model achteraf te kunnen testen. CIFAR-10 bestaat daarnaast ook uit 10 categorieën zoals honden, auto's...

De eerste stap is het pre-trainen van een autoencoder als voorbereiding voor de clustering. Dit pre-trainen is nodig om de belangrijkste features te halen uit de gegeven dataset, aangezien deze features niet gekend zijn.

De autoencoder zelf bestaat uit twee convolution-pooling laag paren en één extra convolutional layer voor de encoder én twee deconvolution-unpooling laag paren en één extra deconvolutional laag.

Hierna kan de dataset omgezet worden naar een efficiëntere representatie, zodat deze gemakkelijker in clusters kunnen gestoken worden. Mijn implementatie gebruikt k-means om een nieuwe manier uit te kunnen proberen van training die in de volgende sectie beschreven wordt.

In mijn implementatie wordt daarna uit het eerste frame een gebied geselecteerd dat getractt moet worden. Dit gebied wordt door de autoencoder omgezet om de cluster te kunnen vinden die best bij de cluster past. Het vinden van de cluster kan gedaan worden door de korste afstand te vinden van het gekozen gebied tot de centra van alle clusters. De cluster van het centrum met de kortste afstand is dan de cluster voor het gebied.

Als men de overeenkomstige cluster heeft, moet men tenslotte enkel nog aan de hand van een sliding window, die iedere frame afgaat, kijken of het gebied onder de sliding window behoort tot de gekozen cluster of niet. Het vinden van de cluster voor het gebied is dezelfde als beschreven in de vorige paragraaf.

Als het gebied deze bevat, dan kunnen de dimensies van de sliding window gebruikt worden om het gebied te markeren waar de cluster voldaan is (indien meerdere sliding windows de clusters bevatten, dan kan men de minima en maxima van de dimensies gebruiken van de windows om een grotere window te vinden die hen bevat).

Het zou in principe ook mogelijk moeten zijn om aan multi-object tracking te kunnen doen door meerdere gebieden te selecteren en de overeenkomstige clusters te vinden. Om het echter simpel te houden heb ik me gefocust op één object.

5.2 Training

De training van het netwerk is grotendeels gebaseerd op de voorgestelde methode in Clustering with Deep Learning[1]. Deze methode maakt gebruik van twee fasen om het netwerk te trainen. De eerste bestaat uit het pre-trainen van een autoencoder, terwijl de tweede fase bestaat uit het jointly trainen van de autoencoder en het kmeans algoritme. De pre-training is

vanzelfsprekend en gebruikt in de gebruikte implementatie squared mean error als loss en RMSProp als optimizer.

De joint training blijkt echter iets complexer. Allereerst is de reden dat de autoencoder en kmeans jointly worden getraind, omdat op die manier de autoencoder opnieuw kan getraind worden bovenop wat hij al heeft geleerd, maar met een nadruk op de foto's omzetten naar een clustervriendelijke ruimte, opdat clusters compacter en duidelijker zijn.

Deze joint training wordt gedaan aan de hand van een gewogen som van de autoencoder loss en de kmeans loss. De gewichten van beide losses is een parameter die aangepast kan worden of één die kan geoptimaliseerd worden door het netwerk. In mijn experiment heb ik de waarden waardes overgenomen die door de implementatie van de paper gegeven waren.

De autoencoder loss is dezelfde als die bij de pre-training. De kmeans loss echter gebruikt een andere soort loss. Deze gebruikt namelijk cluster assignment hardening. Men bekomt deze metriek door de soft assignments van de datapunten tot de clusters te bepalen. Dit doet men aan de hand van een kernel zoals de Student t-distributie. De metriek is niet meer dan een soort van percentage om aan te geven hoeveel kans er is dat een datapunt tot een cluster behoort.

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\nu)^{-\frac{\nu+1}{2}}}$$

Na het berekenen van de soft assignments kan men percentages hard maken, opdat de waarden dichter bij 0 of 1 liggen (dit zorgt voor de dichtere clusters). Men doet dit door de initiële distributie een andere distributie te laten naderen.

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'}(q_{ij'}^2 / \sum_i q_{ij'})}$$

Tenslotte kan de divergentie tussen de twee distributies berekend worden om de loss te bekomen. De optimizer is dezelfde als voorheen.

$$L = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Nadat de autoencoder in staat is om clustervriendelijke, laag dimensionale datapunten te creeren, kan men het kmeans algoritme trainen op deze datapunten. Het gevolg hiervan zijn clusters die zeer specifiek zijn.

Deze specifieke clusters zijn dan weer degene die gebruikt worden om voor iedere sliding window in iedere frame van een video te bepalen of deze het gebied bevat dat voorheen gekozen werd.

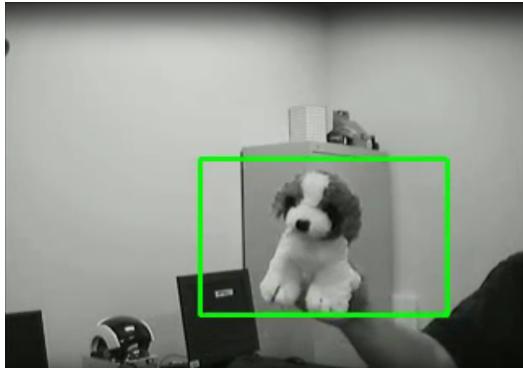
5.3 Experimentele resultaten

De resultaten waren veel belovend. Ik heb het algoritme voornamelijk op twee soorten video's geprobeerd: een speelgoedhond die heen en weer, vooruit en achteruit wordt bewogen en karakters in animatiefilms.

Er waren initieel problemen om de hond in de bounding box te krijgen, waarbij de bepalende metriek het aantal clusters bleek te zijn. Het aantal clusters in het begin was 100 en het resultaat was dat de hond maar in één seconde van de video in de bounding box zat. Bij het verlagen van het aantal clusters naar 10 (het aantal objecten in de dataset), zat de hond in de meeste frames binnen de bounding box. Er was echter een trade-off: de tracker begon de persoon die de hond vasthield ook in sommige frames te betrekken.

Om te kijken of de tracker werkte op zelfgemaakte datasets, heb ik beroep gedaan op een kort fragment van een animatiefilm. In de animatiefilm was het de bedoeling om een blond karakter te tracken. Om dit te kunnen doen werden 1000 extra foto's van blonde karakters toegevoegd. Het eerste resultaat dat ik hiermee bekwam was niet fraai. Zelfs het zwart harige karakter in het fragment was op sommige moment in de bounding box te vinden.

Hierna werden de 1000 foto's vervangen door andere foto's van een andere dataset. Deze dataset werd gemaakt door gezichten uit foto's nemen van de populaire imageboard website Danbooru. Deze site is voornamelijk om fan-



Figuur 5.1: Bounding box die werd gevonden door de tracker voor de hond

art op het internet te zetten van animatie karakters. Elke foto heeft echter zeer specifieke tags die door gebruikers kunnen gezet worden, waardoor het relatief gemakkelijk is om foto's te vinden die aan bepaalde voorwaarden voldoen.

Bij het gebruiken van de nieuwe dataset ging de kwaliteit van de tracking al vele malen beter. Enkel als het karakter te ver was, had de tracker moeite met deze terug te vinden. Niet alleen het aantal clusters is dus van belang, maar ook de kwaliteit van de dataset om een goed resultaat te bekomen.

Dit tweede was ook te merken wanneer ik de dataset had vergroot met twee soorten nieuwe haarkleuren: blauw en paars (een vreemde kleur op zich, maar vrij normaal binnen een sub-set van animatiefilms). Zowel voor blauw als paars werden 1000 foto's van de zojuist genoemde dataset toegevoegd. Ook het aantal clusters werd met twee verhoogd. Een ander fragment werd gebruikt om deze kleur te kunnen testen

Het resultaat was iets minder kwalitatief als de andere, voornamelijk omdat zich vooral één probleem voordeed: paars en blauw werden vaak als dezelfde cluster aanzien. Dit probleem zal hoogstwaarschijnlijk liggen aan het feit dat bij de clustering de kleur paars en blauw te dicht bij elkaar aanleunen.

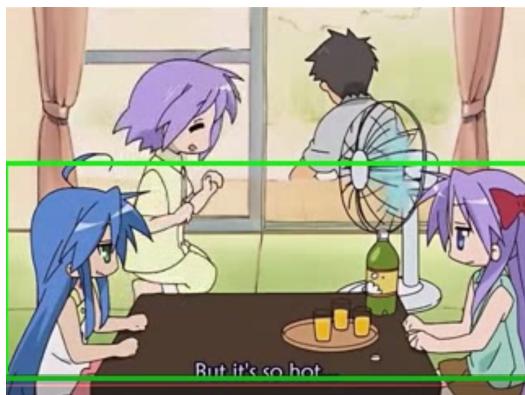
Tenslotte was een laatste test met enkel de toevoeging van de blauw harige



Figuur 5.2: Bounding box die werd gevonden door de tracker voor het blond harige karakter

dataset. Het nieuwe fragment had een aantal frames waarin het karakter op een hoge snelheid over het scherm bewoog. Desalniettemin was het de tracker toch gelukt om het karakter te identificeren over dit klein aantal frames.¹

¹De code voor deze paper kan gevonden worden op https://github.com/IlliasIB/unsupervised_video_tracking



Figuur 5.3: Verwarring tussen blauw en paars haar

Hoofdstuk 6

Conclusie

De tracker lijkt reële resultaten te vertonen onder bepaalde situaties. Parameters die echter aangepast moeten worden afhankelijk van de situatie zijn het aantal clusters en de dataset.

Sub-sets van data die te veel op elkaar gelijkt zal moeite hebben om in verschillende clusters terecht te komen. Daarnaast kan een hoog aantal clusters er toe leiden dat de tracker te precies gaat werken, waardoor het object enkel in zeer specifieke situaties gevonden zal worden. Een laag aantal zorgt er echter voor dat bepaalde objecten gemakkelijker als één en dezelfde zullen aanshouwd worden.

Men kan de implementatie verbeteren door de mogelijkheid aan te bieden meerdere objecten te kiezen in eender welke frame. Dit kan gerealiseerd worden door meerdere clusters te selecteren op basis van de keuzes van de objecten.

De implementatie van de tracker werkt ook momenteel in vrij lage dimensies. De foto's zijn 32 bij 32 en de video moet 360 bij 240 zijn. Het verhogen van de dimensies kan mogelijk als gevolg hebben dat het resultaat van de tracker gunstiger uit komt.

De tracker heeft echter als probleem dat deze momenteel relatief veel tijd nodig heeft (soms tot twee uur) om het autoencoder model in te laden en

ieder frame te analyseren. Het verhogen van het aantal dimensies kan dus ook er voor zorgen dat het tracken in ieder frame nog langer duurt (alhoewel de lange tijd misschien kan liggen aan de implementatie).

Als de dimensies verhoogd worden zal de architectuur van de autoencoder echter aangepast moeten worden om te kunnen compenseren voor het hogere aantal features dat met de input er door komt.

Bibliografie

- [1] Elie Aljalbout e.a. “Clustering with Deep Learning: Taxonomy and New Methods”. In: *CoRR* abs/1801.07648 (2018). arXiv: 1801.07648. URL: <http://arxiv.org/abs/1801.07648>.
- [2] Amit Bhola en Arvind Kumar Tiwari. “Machine Learning Based Approaches for Cancer Classification Using Gene Expression Data”. In: *Machine Learning and Applications: An International Journal* 2.3 (2015).
- [3] Ronan Collobert e.a. “Natural Language Processing (Almost) from Scratch”. In: *J. Mach. Learn. Res.* 12 (nov 2011), p. 2493–2537. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=1953048.2078186>.
- [4] Jesse Davis en Mark Goadrich. “The Relationship Between Precision-Recall and ROC Curves”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, p. 233–240. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143874. URL: <http://doi.acm.org/10.1145/1143844.1143874>.
- [5] Tom Fawcett. “An Introduction to ROC Analysis”. In: *Pattern Recogn. Lett.* 27.8 (jun 2006), p. 861–874. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. URL: <http://dx.doi.org/10.1016/j.patrec.2005.10.010>.
- [6] Zoubin Ghahramani. “Unsupervised Learning”. In: *Advanced Lectures on Machine Learning*. Red. door Olivier Bousquet, Ulrike von Luxburg en Gunnar Rätsch. Deel 3176. Lecture Notes in Artificial Intelligence. Springer, 2003, p. 72–112.

- [7] *Google Picture this: A fresh approach to Photos.* <https://blog.google/products/photos/picture-this-fresh-approach-to-photos/>. Accessed: 2018-02-19.
- [8] Samuel Greengard. “Gaming Machine Learning”. In: *Commun. ACM* 60.12 (nov 2017), p. 14–16. ISSN: 0001-0782. DOI: 10.1145/3148817. URL: <http://doi.acm.org/10.1145/3148817>.
- [9] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn and TensorFlow”. In: O’Reilly Media, 2017. Hfdstk. Why use machine learning?, p. 29–30.
- [10] Geoffrey E Hinton en Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), p. 504–507.
- [11] Ron Kohavi en Foster Provost. “Glossary of terms”. In: *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process* 30.2 (1998), p. 271–274.
- [12] Alex Krizhevsky, Ilya Sutskever en Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, p. 1097–1105.
- [13] Quoc Le e.a. “Building High-level Features Using Large Scale Unsupervised Learning”. In: *ICML 2012: 29th International Conference on Machine Learning* (2012).
- [14] *Machine Learning Mastery Discover Feature Engineering, How to Engineer Features and How to Get Good at It.* <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>. Accessed: 2018-02-19.
- [15] *Medium Precision and Recall.* <https://medium.com/ml-ai-study-group/precision-and-recall-b47202b29bc7>. Accessed: 2018-02-26.
- [16] Donald Michie, David Spiegelhalter en Charles Taylor. “Machine Learning, Neural and Statistical Classification”. In: Overseas Press, 2009. Hfdstk. Classification.
- [17] Richard Oentaryo e.a. “Detecting Click Fraud in Online Advertising: A Data Mining Approach”. In: *J. Mach. Learn. Res.* 15.1 (jan 2014), p. 99–140. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=2627435.2627438>.

- [18] Carl Edward Rasmussen en Christopher K. I. Williams. “Gaussian Processes for Machine Learning”. In: The MIT Press, 2006. Hfdstk. Regression.
- [19] Jinchang Ren e.a. “Tracking the Soccer Ball Using Multiple Fixed Cameras”. In: *Comput. Vis. Image Underst.* 113.5 (mei 2009), p. 633–642. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2008.01.007. URL: <http://dx.doi.org/10.1016/j.cviu.2008.01.007>.
- [20] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.
- [21] Stuart Russell en Peter Norvig. “Artificial Intelligence: A Modern Approach”. In: 3de ed. Prentice Hall, 2010. Hfdstk. Learning From Examples, p. 29–30.
- [22] Hinrich Schütze, Christopher D. Manning en Prabhakar Raghavan. *Introduction to information retrieval*. Deel 39. Cambridge University Press, 2008.
- [23] Soharab Hossain Shaikh, Khalid Saeed en Nabendu Chaki. “Moving Object Detection Using Background Subtraction”. In: deel 3176. Springer, 2014. Hfdstk. Moving Object Detection Approaches, Challenges and Object Tracking, p. 5–14.
- [24] *Sigmoidal Machine Learning For Trading – How To Predict Stock Prices Using Regression?* <https://www.quantinsti.com/blog/machine-learning-trading-predict-stock-prices-regression/>. Accessed: 2018-02-25.
- [25] Yunjia Sun, Edward Lank en Michael Terry. “Label-and-Learn: Visualizing the Likelihood of Machine Learning Classifier’s Success During Data Labeling”. In: *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*. IUI ’17. Limassol, Cyprus: ACM, 2017, p. 523–534. ISBN: 978-1-4503-4348-0. DOI: 10.1145/3025171.3025208. URL: <http://doi.acm.org/10.1145/3025171.3025208>.
- [26] P.N. Tan, Michael Steinbach en Vipin Kumar. “Cluster Analysis: Basic Concepts and Algorithms”. In: (jan 2005), p. 487–568.
- [27] *Towards Data Science Cross-Validation in Machine Learning*. <https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>. Accessed: 2018-02-26.

- [28] Moshe Y. Vardi. “The Moral Imperative of Artificial Intelligence”. In: *Commun. ACM* 59.5 (apr 2016), p. 5–5. ISSN: 0001-0782. DOI: 10 . 1145/2903530. URL: <http://doi.acm.org/10.1145/2903530>.
- [29] Sheng Wei en Ren Jianxin. “Real-time Tracking of Non-rigid Objects”. In: *Proceedings of the 2016 International Conference on Communication and Information Systems*. ICCIS ’16. Bangkok, Thailand: ACM, 2016, p. 11–15. ISBN: 978-1-4503-4791-4. DOI: 10 . 1145 / 3023924 . 3023944. URL: <http://doi.acm.org/10.1145/3023924.3023944>.
- [30] Jyoti Yadav en Monika Sharma. “A Review of K-mean Algorithm”. In: *International Journal of Engineering Trends and Technology* 4.7 (2013), p. 2972–2976.
- [31] Alper Yilmaz, Xin Li en Mubarak Shah. “Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.11 (nov 2004), p. 1531–1536. ISSN: 0162-8828. DOI: 10 . 1109/TPAMI . 2004 . 96. URL: <http://dx.doi.org/10.1109/TPAMI.2004.96>.
- [32] Gao Zhu, Fatih Porikli en Hongdong Li. “Robust Visual Tracking With Deep Convolutional Neural Network Based Object Proposals on PETS”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2016.