IMAGE CLASSIFICATIE_
Onderzoek

Groep 3 Opleiding: HBO-ICT Studiejaar: 3 27 februari 2023 Docenten: Martijn Driessen, Sjir Schütt Versie 2.3	**Studenten:** Sven van Ee 1645999 Marijn Martin 1547235 Sam Wolters 1653733 Sjoerd de Bruin 1650151

HAN_UNIVERSITY OF APPLIED SCIENCES

HAN_UNIVERSITY
OF APPLIED SCIENCES

INHOUDSOPGAVE

1 INLEIDING

Image classificatie is een belangrijke techniek in computer vision en wordt veelvuldig gebruikt bij de analyse en verwerking van beelden en video's. Het doel van image classificatie is om klassementen in beelden te identificeren. Dit is van groot belang voor veel industrieën, waaronder veiligheid, entertainment en retail.

In dit onderzoeksverslag richten we ons op het onderzoeken van verschillende methoden voor image classificatie. Er wordt onderzocht welke technieken het meest nauwkeurig en efficiënt zijn en hoe we deze kunnen toepassen in onze specifieke usecase.

We zullen een uitgebreide analyse uitvoeren van verschillende image classificatie algoritmen, waarbij we zowel de nauwkeurigheid als de snelheid zullen vergelijken. Hierdoor kunnen we bepalen welke technieken het meest geschikt zijn voor onze specifieke toepassingen en hoe we deze het beste kunnen implementeren.

Met dit onderzoek hopen we een diepgaand inzicht te verkrijgen in de mogelijkheden en beperkingen van image classificatie.

2 ONDERZOEKSAANPAK

Het onderzoek is kwalitatief gericht uitgevoerd en is in de vorm van een literatuuronderzoek gedaan. De bron voor dit onderzoek is het internet.

De hoofdvraag voor dit onderzoek luidt als volgt: ‘Welk image classificatie algoritme is het beste voor het NotS project?’. Om deze hoofdvraag te beantwoorden zijn een aantal deelvragen opgesteld:

- Wat is image classificatie?
- Hoe werkt image classificatie?
- Hoe wordt de nauwkeurigheid beoordeeld?

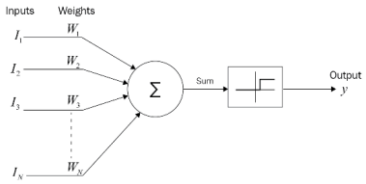
Als eerst wordt besproken wat image classificatie is, hoe het werkt en hoe de nauwkeurigheid wordt beoordeeld. Vervolgens worden criteria gespecificeerd die gebruikt zullen worden bij het beoordelen van de algoritmen. Deze criteria zijn opgebouwd aan de hand van zelf gespecificeerde criteria die ervoor moeten zorgen dat het algoritme relevant is.
Dertien algoritmen worden geselecteerd die beoordeeld zullen gaan worden aan de hand van deze gespecificeerde criteria. Daarna wordt aan de hand van een checklist gecontroleerd welke algoritmen voldoen aan de criteria. Ten slotte wordt een shortlist opgesteld om te kijken welk algoritme het meest naar voren komt. Op deze algoritmen worden een aantal testen uitgevoerd.

Door deze deelvragen te beantwoorden kan uiteindelijk in de conclusie de hoofdvraag worden beantwoord.

3 WAT IS IMAGE CLASSIFICATIE?

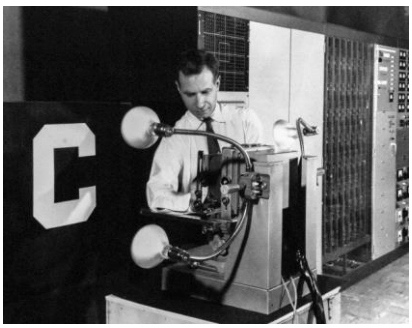
3.1 De geschiedenis van image classificatie

Image classificatie is geen nieuw onderwerp in computer vision. Al in 1943 probeerde mensen al om het menselijke brein op een wiskundige manier te modelleren. Hiervoor hebben neurobiologen *Mculloch* en *Pitts* een wiskundig neuron bedacht dat een signaal wel of niet doorstuurt afhankelijk van de invoer. In 1949 is door *Donald Hebb* bedacht dat deze neuronen een verschillende kracht op elkaar moeten kunnen uitoefenen, de *weight* (History of the Perceptron, sd). Echter was het rond deze tijd nog niet makkelijk om deze ideeën daadwerkelijk om te zetten naar een computer.



Figuur 1. McCulloch Pitts Neuron. (The McCulloch-Pitts neuron, sd)

In 1958 werd de eerste poging gedaan naar een computer die afbeeldingen kon classificeren. Er is toen door *Frank Rosenblatt* een computer bedacht, de *Perceptron*, die automatisch letters kon classificeren. De neuronen die hierbij gebruikt waren iets anders dan de McCulloch Pitts neuronen en konden nu ook getallen doorgeven tussen de 0 en 1, er was dus geen “uit” of “aan” meer (Lefkowitz, 2019).



Figuur 2. Rosenblatt en de Perceptron. (Lefkowitz, 2019)

Dit model bleek voor dit probleem goed te werken en er ontstonden hoge verwachtingen voor de Perceptron. Zo had het Amerikaanse leger de volgende verwachting voor de volgende generaties van de Perceptron: “*The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence*“ (A Sociological History of the Neural, 1993). Maar al snel werden de limieten van het systeem ontdekt. Ze lieten zien dat de Perceptron alleen “linearly separable” functies kon oplossen. Als gevolg hiervan werd er bijna geen nieuw onderzoek gedaan naar image classificatie tot 1980 (History of the Perceptron, sd).

Er waren drie grote problemen die zich voordeden bij het ontwikkelen van image classificatie algoritmen. De hardware was niet krachtig genoeg om grotere modellen efficiënt te kunnen ontwikkelen, er was niet genoeg data om modellen op te trainen en de algoritmes die er bestonden waren niet accuraat genoeg.

In de jaren 1980 en later werden veel neurale netwerken ontwikkeld voor specifieke taken, bij het ontwikkelen hiervan werden veel belangrijke technieken ontdekt die nu nog steeds gebruikt worden

zoals Convolutional layers, back-propagation en gradient descent (Backpropagation Applied to Handwritten Zip Code Recognition, 1989). Met deze technieken konden algoritmes ontwikkeld worden die beter presteerde in accuraatheid.

Het internet heeft geholpen bij het trainen van AI door toegang te bieden tot grote hoeveelheden data. AI-training begint met data en alle machine learning-projecten vereisen hoogwaardige, goed geannoteerde data om succesvol te zijn. (TELUS International, 2021)

Pas na 2000 werden de eerste parallel implementaties op een grafische kaart ontwikkeld, waardoor image classificatie algoritmes sneller en beter getraind konden worden (Oh, 2004). In 2006 kwam Nvidia uit met een programmeer toolkit genaamd ‘CUDA’. Met het gebruik van deze toolkit konden onderzoekers sneller en goedkoper deep learning modellen ontwikkelen op een grafische kaart (Tilley, 2016).

3.2 Waar wordt image classificatie nu voor gebruikt?

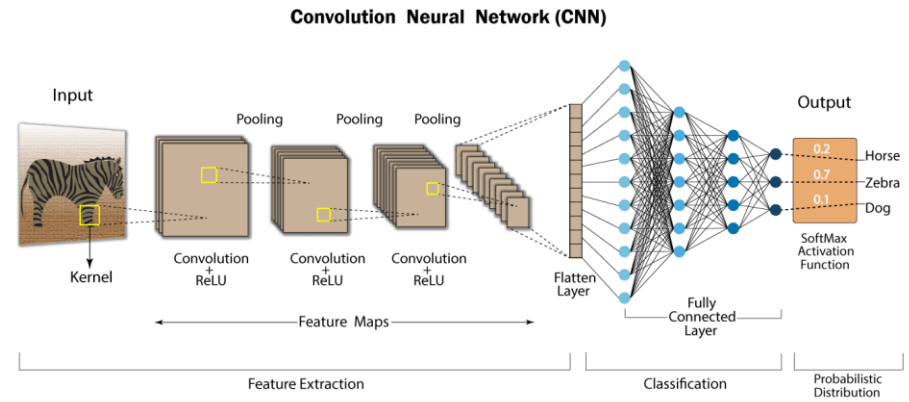
Image classificatie is een belangrijke technologie achter zelfrijdende auto's, waardoor ze een stopbord kunnen herkennen of een voetganger kunnen onderscheiden van een lantaarnpaal. Het is ook nuttig in een verscheidenheid aan toepassingen zoals ziekte-identificatie in bio-imaging, industriële inspectie en robotzicht (MATLAB, sd). Ook kan er met image classificatie gecontroleerd worden of een gebouw beschadigd is door bijvoorbeeld natuurrampen (ArcGIS, sd).

Tijdens de kick-off van het NotS semester gaf een spreker van Alliander een presentatie over het gebruik van image classificatie in het bedrijfsleven. Alliander past deze technologie toe om de onderdelen in een meterkast te herkennen en classificeren aan de hand van foto's die klanten maken bij storingen. Dit bespaart tijd en kosten, waardoor storingen sneller kunnen worden verholpen (NotS Kick-Off Presentatie, sd).

4 HOE WERKT IMAGE CLASSIFICATIE?

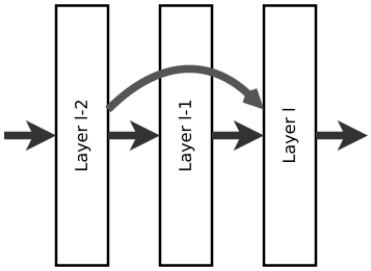
Tegenwoordig worden image classificatie algoritmes getraind met Convolutional Neural Networks (CNNs). Voordat de afbeelding geclassificeerd kan worden moet er eerst een aantal beeldbewerking stappen uitgevoerd worden op de afbeelding. Dit is nodig om de interessante eigenschappen of “features” uit de afbeelding te halen. Dit gebeurt met een aantal convoluties, pooling en activation stappen. Deze worden verder toegelicht in het hoofdstuk ‘Trainen'. Na deze stappen worden de waardes van de features in een netwerk verwerkt.

Dit netwerk bestaat uit een grote groep verbonden neuronen, ook wel de Fully Connected Layer (FCL) genoemd, waarbij één groep van deze neuronen een laag heet. Door veel van deze lagen achter elkaar te plaatsen en elkaar te laten beïnvloeden ontstaat een FCL. De eerste laag bevat de waardes van de pixels in een afbeelding en wordt ook wel de *input layer* genoemd. Uiteindelijk heeft de laatste laag, de *output layer*, een aantal waardes aangenomen. Uit deze waardes kan worden afgeleid wat zich in de afbeelding bevindt.



Figuur 3: CNN (Swapna, 2020)

4.1 ResNet

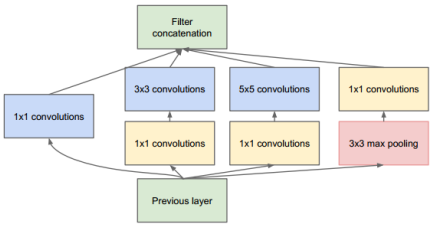


Figuur 4: Uitleg ResNet (Residual neural network, 2023)

Residual Networks (ResNets) waren uitgevonden om het vanishing gradient en degradation gradient probleem op te lossen die CNNs hadden. Het degradation gradient probleem gebeurt wanneer er zo veel layers aan een CNN toegevoegd worden dat er meer training fouten optreden. ResNet lost dit probleem op met het gebruik van 'skip connections'. Als we kijken naar 'Figuur 5' dan zien wij dat een pijl van 'Layer l-2' naar 'Layer l' gaat. Dit is een zo genoemde skip connection. Het netwerk zal de ouput van 'Layer l-2' meegeven aan 'Layer l-1', maar ook verder doorgeven aan 'Layer l'. Vervolgens worden de outputs van 'Layer l-2' en 'Layer l-1' bij elkaar opgeteld en aan 'Layer l' meegegeven. (Residual neural network, 2023)

4.2 InceptionNet

InceptionNet is een uitbreiding op CNN. Het maakt gebruik van 'inception blocks' waarbij elke block verschillende convolutie lagen met verschillende kernel sizes bevat. Dit bleek betere resultaten terug te geven en zorgde er ook voor dat minder krachtige hardware nodig was. (Sarkar, 2020)



Figuur 5: Uitleg InceptionNet block (Sarkar, 2020)

Het gebruik van zo'n netwerk bestaat uit twee stappen, trainen en classificeren. Bij het trainen wordt het netwerk aangeleerd welke classificaties bij bepaalde afbeeldingen horen, bij de classificatie stap kunnen nieuwe afbeelding worden geclassificeerd. Deze stappen worden hieronder verder toegelicht.

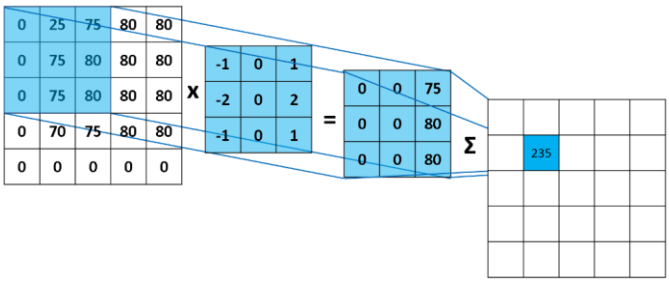
4.3 Trainen

Voordat het netwerk gebruikt kan worden moet het eerst getraind worden, dit gebeurt per afbeelding in vier stappen (Introducing Convolutional Neural Networks, 2022).

4.3.1 Convolutie

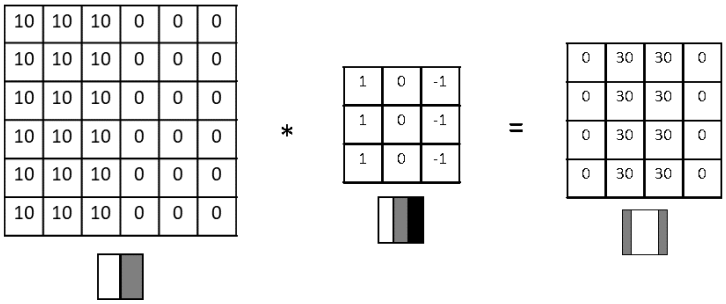
Het netwerk gaat stap voor stap door de afbeelding heen en zet de afbeelding om naar een andere afbeelding met bepaalde eigenschappen die interessant zijn voor het netwerk, deze nieuwe afbeelding heet ook wel de *feature map.* Dit omzetten naar een feature map gebeurt met een aantal filters en elk filter maakt zijn eigen feature map.

Een filter bestaat uit een of meerdere kernels. Een kernel is een matrix van nummers die over de afbeelding wordt verschoven. Bij elke verschuiving worden de waardes uit een stukje van de afbeelding vermenigvuldigt met de waardes in de kernel. De waardes die hieruit berekend worden vormen de feature map.



Figuur 6. Verschuivende kernel en vorming van feature map. (Convolutional Neural Networks - Basics, 2017)

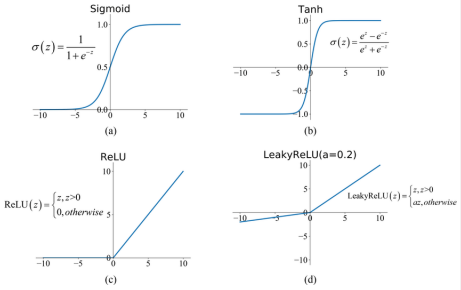
Een voorbeeld van een nuttig filter is bijvoorbeeld een kernel die randen in een afbeelding kan herkennen. Hierbij wordt een kernel met positieve waardes aan een kant, en negatieve waardes aan de andere kant over een afbeelding geschoven. Hierdoor worden in de feature map alleen waardes behouden als er een contrast is in de originele afbeelding.



Figuur 7. Voorbeeld van een edge detection kernel. (CNN More On Edge Detection, 2018)

Met gebruikt van machine learning kunnen er veel grotere en ingewikkelder filters ontwikkeld worden die mensen niet zelf zouden kunnen bedenken. Hierdoor kan een model “leren” hoe het een afbeelding moet analyseren om tot een nuttige feature map te komen. (Ganesh, 2019)

4.3.2 Activation



Figuur 8: Activation functions (Feng, 2019)

Na de convolutie wordt een wiskundige operatie uitgevoerd om ongewenste waardes eruit te filteren. In ‘Figuur 9’ is een voorbeeld van vier verschillende berekeningen te zien.

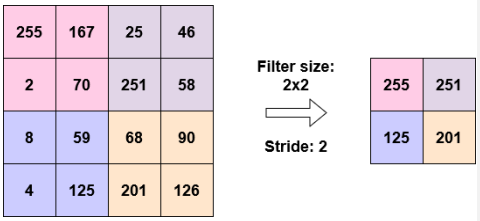
- Sigmoid: Wanneer de invoer waarde een enorm groot negatief getal of enorm groot positief getal is, zal de Sigmoid functie deze waarde omzetten naar een waarde tussen de 0 en de 1 (Saeed, 2021).
- Tanh: De functie neemt elk getal als invoer en geeft waarden uit in het bereik van -1 tot 1. Hoe groter de invoer (meer positief), des te dichter de uitvoerwaarde bij 1.0 zal liggen, terwijl hoe kleiner de invoer (meer negatief), des te dichter de uitvoer bij -1.0 zal liggen (Brownlee, How to Choose an Activation Function for Deep Learning, 2021).
- ReLU: Dit zet alle negatieve waardes in het netwerk om naar een nul, en behoud alle positieve waardes. Doordat de waardes nu minder lineair zijn kan het netwerk complexere relaties leren (Brownlee, A Gentle Introduction to the Rectified Linear Unit, 2019).
- LeakyReLU: Deze functie is gebaseerd op ReLU, maar heeft een kleine helling voor negatieve waarden in plaats van een vlakke lijn. De hellingcoëfficiënt wordt voorafgaand aan de training bepaald, dat wil zeggen dat deze niet tijdens de training wordt geleerd (McCloskey, 2022).

De populaire activatiefuncties sigmoid en tanh worden niet veel meer gebruikt vanwege hun problemen met vanishing gradient en exploding gradient. Deze problemen worden verder toegelicht in het kopje ‘Backpropagation’.

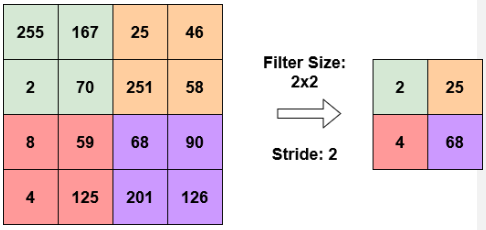
4.3.3 Pooling

Het resultaat van de vorige stappen wordt gecomprimeerd naar een kleinere schaal. Hierbij worden de waardes van een aantal nodes samengevoegd naar een enkele node. De drie methodes die vrijwel altijd gebruikt worden zijn:

- Maximum pooling: De nieuwe waarde is de maximale waarde uit de originele groep waardes.
- Minimum pooling: De nieuwe waarde is de laagste waarde uit de originele groep waardes.
- Average pooling: De nieuwe waarde is de gemiddelde waarde uit de originele groep waardes.

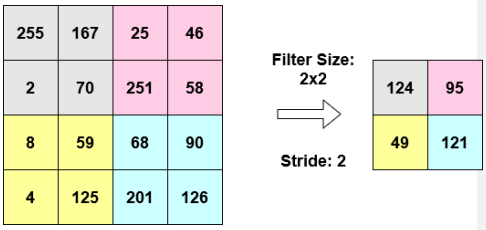


Figuur 9: Maximum pooling (Hasan, sd)



Figuur 10: Minimum pooling (Hasan, sd)

Maximum pooling wordt vaker gebruikt dan gemiddelde en minimale pooling, omdat het een scherper beeld oplevert dat gericht is op de maximale waarden. (Sharma, sd)



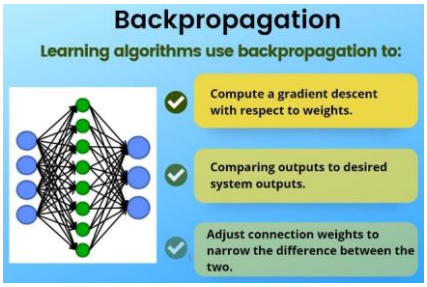
Figuur 11: Average pooling (Hasan, sd)

Pooling heeft drie belangrijke effecten:

- Het bespaart rekenkracht: Doordat pooling het aantal waardes in de afbeelding verminderd hoeft het netwerk ook een minder grotere input aan te nemen. Doordat de input nu minder groot is kan het totale netwerk ook kleiner zijn, wat rekenkracht bespaart. (savyakhosla, 2023)
- Het maakt het netwerk consistenter in voorspellingen: Door de downsampling van pooling wordt informatie verloren van de locatie van objecten in de afbeelding. Hierdoor wordt het netwerk minder gevoelig voor veranderingen in de verplaatsing van deze objecten. Dit heet '*local translation invariance*'. Doordat de objecten in een afbeelding zich nu niet meer op een bepaalde plaats hoeven te bevinden kan het netwerk met veel meer verschillende afbeeldingen omgaan en is het dus veel consistenter (Soni, 2019).
- Het beschermt tegen overfitting: Door het selecteren van een subset aan kenmerken wordt de kans kleiner dat er valse patronen gevonden worden. (Talbi, 2020)

4.3.4 Backpropagation

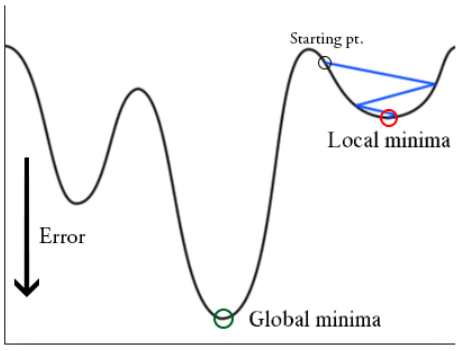
Vervolgens worden de gewichten van output naar input aangepast. Dit wordt gedaan door de gewenste uitvoer te vergelijken met de behaalde uitvoer van het systeem. Zo worden de systemen afgesteld door de verbindingsgewichten aan te passen om het verschil tussen de twee zo veel mogelijk te verkleinen. (Zola, 2022)



Figuur 12: Werking van backpropagation (Backpropagation, 2022)

Cost Function

Het aanpassen van deze gewichten naar een gewenste oplossing gebeurt met een zogenaamde *cost function*. De cost function, ook wel de error of loss function, is een berekening dit uitrekent hoe goed het netwerk is in zijn taak. Het doel bij de backpropagation is om een zo laag mogelijke waarde te bereiken bij de cost function. Het proces van het vinden van een zo laag mogelijke waarde heet *optimizing*. En de beweging van de cost function naar een minima heet de *gradient descent.* Echter is het op deze manier niet mogelijk om het globale minimum van de cost function te vinden, er kan dus niet gegarandeerd worden dat een getraind model de beste oplossing heeft gevonden (Mahmood, 2019).



Figuur 13. Gradient descent naar een lokaal minimum. (Chowdhury, 2020)

Learing Rate & Optimizers

De hoeveelheid waarmee de cost function de gewichten in het netwerk beinvloed heet de *learning rate.* Met een hoge learning rate zullen de gewichten met grote stappen tegelijk aangepast worden. Op deze manier kan er sneller een minimum bereikt worden, maar er is ook het risico dat de stap zo groot is dat het minimum voorbij geschoten wordt. Met een lage learning rate zullen de stappen een stuk kleiner en consistenter zijn, maar je komt dan al snel in een lokaal minimum te zitten. Om deze learning rate op een goede manier aan te passen zijn er Optimizers bedacht, die automatisch de learning rate aanpassen afhankelijk van hoe goed het netwerk presteert.

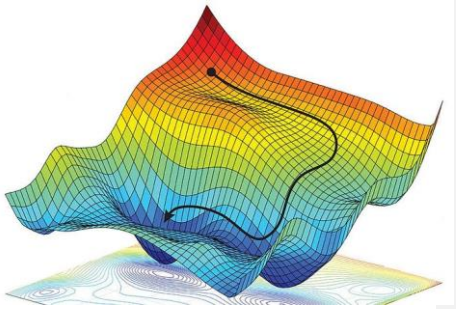
Een optimizer kan bijvoorbeeld de learning rate verhogen wanneer het netwerk geen verbetering meer boekt. Of juist verlagen wanneer de cost function terug omhoog gaat.

Een van de meest gebruikte optimizers is de Adaptive Moment Estimation (Adam) optimizer. Deze optimizer combineert de Adadelta optimizer met de RMSprop optimizer.

Gradient Descent

Er zijn drie veelgebruikte varianten van gradient descent:

- Batch Gradient descent: Hierbij worden alle afbeeldingen uit de trainingsset tegelijk gebruikt om de loss function te berekenen. Er hoeft dus maar een enkele keer een backpropagation stap gedaan te worden, wat rekenkracht bespaart. Maar het grote nadeel is dat er een hele generieke oplossing berekend wordt waardoor de trainingstijd veel langer wordt (Patrikar, 2019).
- Stochastic Gradient descent: Hier wordt elke afbeelding individueel gebruikt om het netwerk te trainen. Er moet dus voor elke afbeelding ook een backpropagation stap gedaan worden. Het netwerk bereikt hierdoor sneller een oplossing, maar deze oplossing zal niet optimaal zijn omdat elke afbeelding een grote individuele invloed heeft op de loss function (Patrikar, 2019).
- Mini-batch Gradient Descent: Mini-batch gradient descent splitst de trainingsdataset op in kleine batches die worden gebruikt om de modelfout te berekenen en de modelcoëfficiënten bij te werken. (Brownlee, A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size, 2017)



Figuur 14. Gradient descent in 3 dimensies (Khandelwal, 2022)

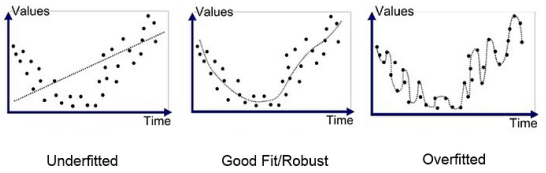
Elke variant verschilt in het aantal data die ze gebruiken (Wagh, 2022). Mini-batch gradient descent wordt aanbevolen omdat het tussen batch gradient descent en stochastische gradient descent valt. Door een subset van de gegevens te nemen, resulteert dit in minder iteraties dan stochastische gradient descent en vermindert het de rekenlast in vergelijking met batch gradient descent (baeldung, 2023).

Vanishing & Exploding Gradient

Er zijn twee grote problemen die kunnen optreden bij gradient descent: *Vanishing gradient* en *Exploding gradient.* Dit kan gebeuren als de gewichten zodanig groot of klein worden dat er bij de gradient descent geen nuttige stappen meer genomen worden, of juist te grootte stappen. Een oplossing die hiervoor gebruikt wordt is *gradient clipping* waarbij de gewichten worden geforceerd een waarde aan te nemen tussen een minimum en maximum (Pykes, 2020).

Fitting

Als een model niet goed genoeg presteert wordt dit *underfitting* genoemd, maar het tegenovergestelde, *overfitting*, kan ook gebeuren. Dit houdt in dat het model te veel de trainingsdata geleerd heeft, en niet de onderliggende patronen in de data. In dit geval zal het model wel goed presteren op de test dataset, maar veel minder goed op de validatie dataset. Het gewenste model ligt dus vaak tussen een underfit en overfit in (Ruizendaal, 2017).



Figuur 15: Voorbeeld overfitting (Talbi, 2020)

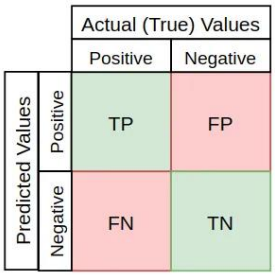
4.4 Classificeren

Nadat het netwerk getraind is en de juiste waardes heeft aangenomen in de neuronen is het classificeren een vrij simpele taak. Door een nieuwe afbeelding en de bijbehorende pixel waardes in de input layer te verwerken gaat het netwerk automatisch berekenen welke bijbehorende waardes er horen in de output layer. De waardes in de output layer zijn de voorspelling van het netwerk over welke classificatie bij deze afbeelding hoort. Door deze voorspelling uit te lezen kan worden ontdekt wat zich er in de afbeelding bevindt (Saha, 2018).

5 HOE WORDT DE NAUWKEURIGHEID BEOORDEELD?

5.1 Confusion matrix

De confusion matrix is een tabel waarin kan worden bijgehouden of een algoritme afbeeldingen juist classificeert. Met behulp van deze tabel kunnen allerlei berekeningen worden gedaan om de nauwkeurigheid te bepalen. (Kanstrén, 2020)



Figuur 16. Confusion matrix (Kanstrén, 2020)

True Positive (TP): Dit is het aantal gevallen waarin het model een positief resultaat voorspelt en het daadwerkelijk positief is. Met andere woorden, het model heeft het correct voorspeld.

Prediction	Actual value	Type	Explanation
1	1	True Positive	Predicted Positive and was Positive
Cat	Cat	True Positive	Cat classifier: Predicted a Cat and it was an actual cat
Cancer	Cancer	True Positive	Cancer classifier: Predicted Cancer and patient really had Cancer

Tabel 1: True positives

False Positive (FP): Dit is het aantal gevallen waarin het model een positief resultaat voorspelt, maar het is eigenlijk negatief. Met andere woorden, het model heeft het foutief voorspeld.

Prediction	Actual value	Type	Explanation
1	0	False Positive	Predicted Positive but was Negative
Cat	No Cat	False Positive	Cat classifier: Predicted a Cat but it was not a cat (maybe it was a dog)
Cancer	No Cancer	False Positive	Cancer classifier: Predicted Cancer but the patient did not have cancer

Tabel 2: False positives

True Negative (TN): Dit is het aantal gevallen waarin het model een negatief resultaat voorspelt en het daadwerkelijk negatief is. Het model heeft het hier correct voorspeld.

Prediction	Actual value	Type	Explanation
0	0	True Negative	Predicted Negative and was Negative
No Cat	No Cat	True Negative	Cat classifier: Predicted No Cat and it was not a cat (maybe it was a dog)
No Cancer	No Cancer	True Negative	Cancer classifier: Predicted No Cancer and patient had no cancer

Tabel 3: True negatives

False Negative (FN): Dit is het aantal gevallen waarin het model een negatief resultaat voorspelt, maar het is eigenlijk positief. Het model heeft het hier foutief voorspeld.

Prediction	Actual value	Type	Explanation
0	1	False Negative	Predicted Negative and was Positive
No Cat	Cat	False Negative	Cat classifier: Predicted No Cat but there actually was a cat
No Cancer	Cancer	False Negative	Cancer classifier: Predicted No Cancer but the patient really had cancer

Tabel 4: False negatives

5.2 Berekeningen nauwkeurigheid

5.2.1 Accuracy

Met de accuracy wordt berekend hoeveel juist voorspellingen er worden gedaan vergeleken met alle voorspellingen. Dit is een van de meest gebruikte scores voor accuraatheid. De berekening hiervoor is als volgt: $\frac{True\ positives+True\ negatives}{True\ positives+True\ negatives+False\ positives+False\ negatives}$

Stel je voor dat een model getraind is op het detecteren van spam e-mails. Deze wordt getest op een dataset van 100 e-mails. 20 hiervan zijn spam en 80 zijn normale e-mails.
Het model detecteert 18 van de 20 spam e-mails correct en detecteert 2 van de 80 normale e-mails incorrect als spam. De andere 78 normale e-mails worden wel correct gedetecteerd en 2 spam e-mails worden incorrect gedetecteerd als normale e-mails.

Met dit voorbeeld zal de bovenstaande berekening er als volgt uit komen te zien
$\frac{18+78}{(18+78+2+2)} = 0.96$
Dit betekent dat het model een accuraatheid heeft van 96%. Dit betekent dat het model heel goed is in het detecteren van beide spam als normale e-mails.

5.2.2 Precision

Bij *Precision* wordt berekend hoeveel van de positieve voorspellingen ook echt correct zijn. De gebruikte formule hiervoor is: $\frac{True\ positives}{True\ positives\ +\ False\ positives}$

Als er bijvoorbeeld 20 positieve voorspellingen zijn, waarvan er 15 true positive zijn is er een precision van $\frac{15}{20} = 75\%$ behaald.

5.2.3 Recall

Recall wordt berekend door de hoeveelheid positieve voorspellingen te vergelijken met de totaal aantal positieve waardes. Om dit te berekenen wordt de berekening $\frac{True\ positive}{True\ positive+ \atop False\ negative}$

Dus als je 100 e-mails zou hebben waarvan 20 spam e-mails zijn en 80 normale e-mails. Als het model 18 van de 20 spam spam e-mails correct detecteert als spam en 2 van de 80 normale e-mails fout detecteert als spam.

Met bovenstaande berekening zou met de data de volgende berekening gemaakt moeten worden.

$\frac{18}{(18+2)} = 0.9\ of\ 90\%$ wat dus laat zien dat dit model goed is in het detecteren van spam mails met lage false negatives.

5.2.4 Specificity

Bij *Specificity* wordt berekend hoeveel van de negatieve voorspellingen correct zijn. De formule hiervoor is: $\frac{True\ negative}{True\ negatives + \atop False\ positives}$

Als er bijvoorbeeld 10 correct negatieve en 5 foute positieve voorspellingen gedaan worden dan is de specificity $\frac{10}{10+5} = 67\%$

5.2.5 F1 score

Bij de F1 score wordt de accuraatheid bepaald door gebruik te maken van de precision en de recall. Het wordt ook wel een '*harmonisch gemiddelde*' genoemd. Het idee is om met een enkele berekening te laten zien hoe het algoritme scoort op de precision en recall.

De gebruikte formule hiervoor is $2 * \frac{Precision*Recall}{Precision+Recall}$

Stel een algoritme bereikt een precision van 90%, en een recall score van 50%, dan is de F1 score $2 * \frac{0.9*0.5}{0.9+0.5} = 64\%$

6 BEOORDELINGSCRITERIA

Met opmerkingen [SdB(1)]: Dit hoofdstuk ff doorlezen en herschrijven mogelijk

Een aantal beoordelingscriteria zijn opgesteld om de algoritme selectie gericht te kunnen houden. Er zijn een aantal criteria gespecificeerd in '**Fout! Verwijzingsbron niet gevonden.**'. Door middel van deze criteria wordt ervoor gezorgd dat de meest geschikte algoritme geselecteerd wordt.

Index	Criteria
C1)	Het is mogelijk om de nauwkeurigheid van het model te beoordelen door middel van verschillende metingen, zoals de precisie, recall en F1-score. Ook kunnen de classificatieresultaten visueel worden geanalyseerd om te zien hoe goed het model presteert.
C2)	De snelheid van het model kan worden gemeten door de tijd te meten die het kost om een enkele afbeelding te classificeren. Ook kunnen prestatietests worden uitgevoerd om te zien hoe het model presteert bij verschillende werkbelastingen.
C3)	Het onderhoud van het model kan worden beoordeeld door te kijken naar de vereisten voor het trainen en updaten van het model. Het moet eenvoudig zijn om het model te onderhouden en te updaten met nieuwe gegevens en algoritmen. Ook kan er gekeken worden naar de beschikbaarheid van de nodige bronnen en tools om het model te onderhouden.

Tabel 5: Beoordelingscriteria

7 LONGLIST

Met opmerkingen [SdB(2)]: Tekst herschrijven en meer algoritmes toevoegen

De onderstaande dertien algoritmes zijn geselecteerd die samen de longlist vormen. Deze selectie is in het hoofdstuk: 'Algoritme Selectie' gebruikt om te kijken welk algoritme voldoet aan de criteria die zijn opgesteld in hoofdstuk 'Beoordelingscriteria'. Zie '**Fout! Verwijzingsbron niet gevonden.**2' voor de geselecteerde algoritmes. Deze dertien algoritmes zijn afkomstig van meerdere sites die al een lijst hebben opgesteld. (Yıldız, sd) (Saha, 2018)

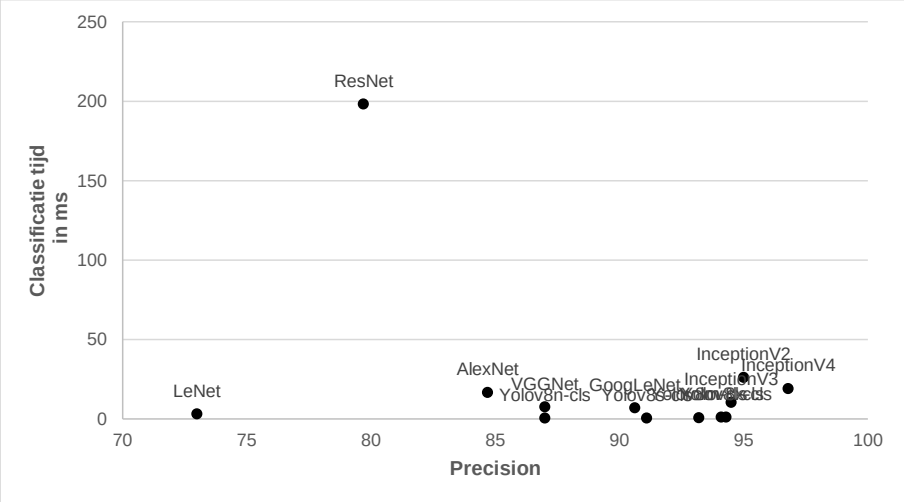
Index	Algoritme	Bron
IC1)	LeNet	(LeNet, sd)
IC2)	AlexNet	(AlexNet, sd)
IC3)	VGGNet	(VGGNet, sd)
IC4)	GoogLeNet	(GoogLeNet, sd)
IC5)	ResNet	(ResNet, 2023)
IC6)	InceptionV2	(The Inception Net, sd)
IC7)	InceptionV3	(The Inception Net, sd)
IC8)	InceptionV4	(The Inception Net, sd)
IC9)	Yolov8n-cls	(Ultralytics github page, sd)
IC10)	Yolov8s-cls	(Ultralytics github page, sd)
IC11)	Yolov8m-cls	(Ultralytics github page, sd)
IC12)	Yolov8l-cls	(Ultralytics github page, sd)
IC13)	Yolov8x-cls	(Ultralytics github page, sd)

Tabel 6: Longlist image classificatie mogelijkheden

8 ALGORITME SELECTIE

Met opmerkingen [SdB(3]: Tekst herschrijven

De algoritme selectie is uitgevoerd aan de hand van een checklist met de criteria uit hoofdstuk 4: ‘Beoordelingscriteria’. De indices uit ‘**Fout! Verwijzingsbron niet gevonden.**’ zijn opgenomen in ‘**Fout! Verwijzingsbron niet gevonden.Fout! Verwijzingsbron niet gevonden.**’. Verder is in deze grafiek de longlist uit ‘**Fout! Verwijzingsbron niet gevonden.**’ opgenomen. In de checklist is per algoritme onderzoek gedaan of deze aan de bijhorende criteria voldoet. Zie ‘**Fout! Verwijzingsbron niet gevonden.**’ voor de volledige uitwerking van de checklist.



Figuur 17: Algoritme selectie

9 SHORTLIST

Met opmerkingen [SdB(4]: Herschrijven

Met behulp van de beoordelingscriteria is bepaald welke algoritmes als beste naar voren komen. Hierna worden de algoritmes die het beste presteren in accuraatheid en snelheid verder grondig getest.

Er worden drie algoritmes getest voor image classificatie. Een algoritme dat het beste presteert in snelheid, een die het beste presteert in precisie en ten slotte een algoritme dat een goede optie biedt tussen precisie en snelheid. De gekozen algoritmes voor image classificatie zijn LeNet, InceptionV4 en YOLOv8x-cls. Elk algoritme wordt getest aan de hand van hetzelfde testplan zodat er een eerlijke vergelijking wordt gemaakt.

9.1 Testplan

Met behulp van Google Colab wordt er een online omgeving opgezet. Hierbij wordt ervoor gezorgd dat de soft- en hardware bij elke test gelijk zijn zodat deze geen invloed kunnen hebben op het resultaat van de test. Ook wordt er bij elke test gebruik gemaakt van de CIFAR-100 dataset (Toronto Edu, sd). Deze dataset is gekozen omdat deze veel variatie bevat, en ook gebruikt wordt om de nauwkeurigheid van een algoritme te bepalen in wetenschappelijke onderzoeken.

Het testen verloopt volgens de volgende stappen:

1. Er wordt een implementatie van het algoritme gebruikt die werkt in Google Colab. Deze implementatie wordt waar nodig aangepast zodat deze gebruik maakt van de eerder benoemde dataset.
2. Er wordt gemeten hoe lang het algoritme nodig heeft om een enkele afbeelding te classificeren.
3. Er wordt gemeten wat de confidence score van het algoritme is op aantal afbeeldingen. De gebruikte afbeeldingen zijn die van een walvis, computer, paddenstoel en een auto. Deze categorieën zijn gekozen omdat ze allemaal voorkomen in de veel gebruikte datasets waarop modellen getraind zijn.
4. Er wordt onderzocht of het model geëxporteerd kan worden naar een ML.NET-compatibel bestand.




9.2 LeNet

LeNet is een Image Classificatie algoritme dat al sinds 1998 bestaat. Het is toen bedacht om kleine afbeelding van getallen en letters te classificeren. Sinds die tijd zijn er al veel verbeteringen bedacht en toegepast, en ondertussen is LeNet-5 de modernste versie.

Voor de implementatie is bestaande code gebruikt (Paudel, 2020) die is omgezet naar een Google Colab notebook, zie bijlage 'Bijlage B'. Met deze implementatie moest LeNet eerst nog getraind worden, er is gekozen voor de CIFAR-100 dataset met een batch size van 20. Dit trainen verliep erg langzaam en na 10 epochs had LeNet een accuraatheid behaald van 5.3% procent op de test dataset. De behaalde winst in accuraatheid werd elke epoch al snel kleiner.

De gebruikte implementatie maakt gebruik van Apache MXNet. Het is mogelijk om deze modellen om te zetten naar een ONNX-model dat gebruikt kan worden in ML.NET.

Voor het classificeren had LeNet gemiddeld een tijd nodig van 8 milliseconden. Deze implementatie van LeNet gaf geen confidence scores, maar deed de volgende voorspellingen:

Walvis: "aquatic mammals"
Paddenstoel: "large man-made outdoor things"
Auto: "household electrical devices"
Computer: "household electrical devices"

LeNet lijkt dus geen geschikte keuze voor onze usecase omdat er dan veel zelf getraind moet worden en de nauwkeurigheid laag blijft.

9.3 Inceptionv4

Inceptionv4 is de vierde variant van het Inception algoritme.

De gebruikte implementatie maakt gebruik van een pretrained model dat is getraind op de ImageNet database (Inception v4). Het zelf trainen van dit model zou extreem veel rekenkracht en tijd kosten tot een redelijke accuraatheid behaald kan worden. Hiervoor is een uitwerking van Google Colab gebruikt, zie bijlage 'Inceptionv4 code voorbeeld'.

Voor het classificeren had Inceptionv4 gemiddeld 513 milliseconden nodig, wat veel meer is dan de andere geteste algoritmes. Het behaalde echter wel zeer goede resultaten met het classificeren van afbeeldingen:

Walvis:
Er is goed herkend dat het een foto van een walvis is met een zekerheid van 94%. De andere voorspellingen zijn ook allemaal van vissen of dieren die in het water leven.

- grey whale 0.9491520524024963
- killer whale 0.01201635506004095
- gar 0.003791891038417816
- dugong 0.0011697809677571058
- barracouta 0.00114177237264812

Paddenstoel:
Ook de paddenstoel is goed herkend. Een *agaric* is een type paddenstoel met een rode kap.

- agaric 0.7462348341941833
- mushroom 0.18073613941669464
- bolete 0.0026492690667510033
- earthstar 0.0024849004112184048
- stinkhorn 0.001441263360902667

Computer:
InceptionV4 heeft de computer goed geclassificeerd. Ook is er herkend dat er een monitor, muis en toetsenbord in de afbeelding zit.

- desktop computer 0.9359255433082581
- screen 0.010701705701649189
- mouse 0.003928416408598423
- computer keyboard 0.003608789062127471
- monitor 0.00330243818461895

Auto

Als een van de weinige is Inceptionv4 er zeer zeker van dat er zich een sportauto bevindt in de foto. Ook ‘racer’ en ‘car wheel’ zijn accepteerbare classificaties.

- sports car 0.7739995121955872
- racer 0.07688093930482864
- car wheel 0.0447685569524765
- grille 0.015540487132966518
- convertible 0.008422225713729858

Conclusie

InceptionV4 is een zeer accuraat algoritme, ook vooral omdat het is model is getraind op de ImageNet dataset. Echter is het uitbreiden van klassementen die zich niet in deze dataset zitten lastig omdat het veel computerkracht en tijd vereist om te trainen. Ook is de tijdsduur voor het classificeren een stuk langer vergeleken met andere algoritmes. InceptionV4 is een zeer geschikte oplossing als alleen de accuraatheid een belangrijke facto is, maar voor onze usecase is deze minder geschikt.

9.4 YOLOv8x-cls

10 CONCLUSIE

Uit het onderzoek naar verschillende image classificatie algoritmes is gebleken dat YOLOv8x-cls een goede keuze is voor classificatie taken. Hoewel alle onderzochte algoritmes in staat waren om accurate resultaten te produceren, bleek YOLOv8x-cls de meest veelzijdige optie te zijn. De combinatie van snelheid en nauwkeurigheid maakt YOLOv8x-cls een ideale keuze voor toepassingen waarbij real-time beeldherkenning vereist is. Kortom, het gebruik van YOLOv8x-cls als classificatie algoritme wordt aanbevolen vanwege de goede prestaties.

BIJLAGE

BIJLAGE A. YOLOV8X-CLS CODE VOORBEELD

pytorch_vision_goo
glenet.ipynb

BIJLAGE B. LENET CODE VOORBEELD

LeNet.ipynb

BIJLAGE C. INCEPTIONV4 CODE VOORBEELD

InceptionNetv4.ipy
nb

LITERATUURLIJST

A Sociological History of the Neural. (1993). Opgehaald van https://gwern.net/doc/ai/nn/1993-olazaran.pdf

AlexNet. (sd). Opgehaald van MathWorks: https://www.mathworks.com/help/deeplearning/ref/alexnet.html

ArcGIS. (sd). *Object classification*. Opgehaald van ArcGIS: https://pro.arcgis.com/en/pro-app/latest/tool-reference/image-analyst/object-classification.htm

Backpropagation. (2022, maart 3). Opgehaald van Techopedia: https://www.techopedia.com/definition/17833/backpropagation

Backpropagation Applied to Handwritten Zip Code Recognition. (1989, juli 7). Opgehaald van http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf

baeldung. (2023, maart 16). *Differences Between Gradient, Stochastic and Mini Batch Gradient Descent*. Opgehaald van baeldung: https://www.baeldung.com/cs/gradient-stochastic-and-mini-batch

Bochkovskiy, A. (2020, april 23). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. Opgehaald van arXiv: https://arxiv.org/abs/2004.10934

Bochkovskiy, A. (23, april 2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection.* Retrieved from arXiv: https://arxiv.org/pdf/2004.10934.pdf

Brownlee, J. (2017, juli 21). *A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size*. Opgehaald van Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

Brownlee, J. (2019, januari 9). *A Gentle Introduction to the Rectified Linear Unit*. Opgehaald van Machine Learning Mastery: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brownlee, J. (2021, januari 18). *How to Choose an Activation Function for Deep Learning*. Opgehaald van Machine Learning Mastery: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Chowdhury, H. A. (2020, januari). *Gradient Sescent Stuck at Local Minima*. Retrieved from ResearchGate: https://www.researchgate.net/figure/Gradient-Descent-Stuck-at-Local-Minima-18_fig4_338621083

CNN More On Edge Detection. (2018, januari 11). Retrieved from Data Hacker: https://datahacker.rs/edge-detection-extended/

COCO - Homepage. (sd). Opgehaald van COCO: https://cocodataset.org/

Convolutional Neural Networks - Basics. (2017, april 7). Retrieved from MLNotebook: https://mlnotebook.github.io/post/CNN1/

Feng, J. (2019, september). *Reconstruction of porous media from extremely limited information using conditional generative adversarial networks*. Retrieved from Researchgate:

https://www.researchgate.net/publication/335845675_Reconstruction_of_porous_media_from_extremely_limited_information_using_conditional_generative_adversarial_networks

Gad, A. F. (2021). *Faster R-CNN Explained for Object Detection Tasks*. Retrieved from PaperspaceBlog: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

Ganesh, P. (2019, oktober). Retrieved from https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37

Girshick, R. (2015, september 27). *Fast R-CNN*. Retrieved from arXiv: https://arxiv.org/abs/1504.08083

GoogLeNet. (sd). Opgehaald van MathWorks: https://www.mathworks.com/help/deeplearning/ref/googlenet.html

Hasan, F. (sd). *What are some deep details about pooling layers in CNN?* Opgehaald van Educative: https://www.educative.io/answers/what-are-some-deep-details-about-pooling-layers-in-cnn

He, K. (2018, januari 24). *Mask R-CNN*. Retrieved from arXiv: https://arxiv.org/abs/1703.06870

History of the Perceptron. (sd). Opgehaald van California State University: https://home.csulb.edu/~cwallis/artificialn/History.htm

Hui, J. (2018, maart 27). *Understanding Feature Pyramid Networks for object detection (FPN)*. Retrieved from Medium: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Inception v4. (n.d.). Retrieved from Hugging Face: https://huggingface.co/docs/timm/models/inception-v4

Introducing Convolutional Neural Networks. (2022, juli 18). Opgehaald van Google: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Kanstrén, T. (2020, september 11). *A Look at Precision, Recall, and F1-Score*. Opgehaald van Towards Data Science: https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Khandelwal, A. (2022, september 1). *The 8-Step Startup Ideation Algorithm.* Retrieved from Medium: https://medium.com/vivid-dev/the-8-step-startup-ideation-algorithm-9fe1a89b427f

Lefkowitz, M. (2019, september 25). *Professor's perceptron paved the way for AI – 60 years too soon*. Retrieved from Cornell University: https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

LeNet. (sd). Opgehaald van d2l.ai: https://d2l.ai/chapter_convolutional-neural-networks/lenet.html

Lin, T.-Y. (2016, december 9). *Feature Pyramid Networks for Object Detection*. Opgehaald van arxiv: https://arxiv.org/abs/1612.03144

Liu, W. (2016, november 30). *SSD: Single Shot MultiBox Detector*. Opgehaald van arXiv: https://arxiv.org/abs/1512.02325

Mahmood, H. (2019, januari 2). *Gradient Descent*. Retrieved from Towards Data Science: https://towardsdatascience.com/gradient-descent-3a7db7520711

MATLAB. (sd). *Object Recognition*. Opgehaald van MATLAB & Simulink: https://www.mathworks.com/solutions/image-video-processing/object-recognition.html

McCloskey, B. (2022, november 15). *Leaky ReLU vs. ReLU Activation Functions: Which is Better?* Retrieved from Towards Data Science: https://towardsdatascience.com/leaky-relu-vs-relu-activation-functions-which-is-better-1a1533d0a89f

Nawani, A. (2023, maart 7). *McCulloch Pitt's Model of Neuron*. Retrieved from Code Studio: https://www.codingninjas.com/codestudio/library/mcculloch-pitt-s-model-of-neuron

NotS Kick-Off Presentatie. (sd). Opgehaald van Google Docs: https://docs.google.com/presentation/d/1Aaj659_SFSV0yASYUojlq7QBbvRsHKCvrBEtypHEHyY/

Oh, K.-S. (2004, januari 13). *ScienceDirect*. Opgehaald van GPU implementation of neural networks: https://www.sciencedirect.com/science/article/abs/pii/S0031320304000524

Patrikar, S. (2019, oktober 1). *Batch, Mini Batch & Stochastic Gradient Descent*. Retrieved from Towards Data Science: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Paudel, R. (2020, juni 17). *LeNet for Image Classification using GluonCV*. Retrieved from Towards Data Science: https://towardsdatascience.com/lenet-for-image-classification-using-gluoncv-829ae7ec4715

Pykes, K. (2020, mei 27). *The Vanishing/Exploding Gradient Problem in Deep Neural Networks*. Retrieved from Towards Data Science: https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11

Redmon, J. (2016, december 25). *YOLO9000: Better, Faster, Stronger.* Retrieved from arXiv: https://arxiv.org/pdf/1612.08242v1.pdf

Redmon, J. (2018, april 8). *YOLOv3: An Incremental Improvement.* Retrieved from arXiv: https://arxiv.org/pdf/1804.02767.pdf

Redmon, J. (sd). *YOLO: Real-Time Object Detection*. Opgehaald van https://pjreddie.com/darknet/yolo/

Redmon, J. (sd). *You Only Look Once: Unified, Real-Time Object Detection.* Opgehaald van arXiv: https://arxiv.org/pdf/1506.02640v5.pdf

Residual neural network. (2023, februari 28). Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Residual_neural_network

ResNet. (2023, januari 10). Opgehaald van geeksforgeeks: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

Ruizendaal, R. (2017, mei 12). *Deep Learning #3: More on CNNs & Handling Overfitting*. Retrieved from Towards Data Science: https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d

Saeed, M. (2021, augustus 25). *A Gentle Introduction To Sigmoid Function*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-

function/#:~:text=The%20sigmoid%20function%20is%20also,between%20%2D%E2%88%9E%20and%20%2B%E2%88%9E.

Saha, S. (2018, december 15). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Opgehaald van Medium: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sarkar, A. (2020, oktober 23). *Understanding Inception: Simplifying the Network Architecture*. Opgehaald van Medium: https://medium.com/swlh/understanding-inception-simplifying-the-network-architecture-54cd31d38949

savyakhosla. (2023, januari 11). *CNN | Introduction to Pooling Layer*. Retrieved from Geeks for geeks: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Sharma, P. (sd). *MaxPool vs AvgPool*. Opgehaald van Opengenus: https://iq.opengenus.org/maxpool-vs-avgpool/

Soni, D. (2019, november 13). *Translation Invariance in Convolutional Neural Networks.* Retrieved from Medium: https://divsoni2012.medium.com/translation-invariance-in-convolutional-neural-networks-61d9b6fa03df

Swapna. (2020, augustus 21). *Convolutional Neural Network | Deep Learning*. Opgehaald van Developers Breach: https://developersbreach.com/convolution-neural-network-deep-learning/

Talbi, I. (2020, november 21). *7 ways to avoid overfitting*. Retrieved from Medium: https://medium.com/analytics-vidhya/7-ways-to-avoid-overfitting-9ff0e03554d3

TELUS International. (2021, mei 19). *How do you train artificial intelligence?* Retrieved from TELUS International: https://www.telusinternational.com/insights/ai-data/article/how-to-train-ai

The Inception Net. (sd). Opgehaald van Notes on AI: https://notesonai.com/The+Inception+Net

The McCulloch-Pitts neuron. (sd). Opgehaald van O'Reilly: https://www.oreilly.com/library/view/artificial-intelligence-by/9781788990547/97eeab76-9e0e-4f41-87dc-03a65c3efec3.xhtml

The State-of-the-Art YOLO Model. (sd). Opgehaald van Ultralytics: https://ultralytics.com/yolov8

Tilley, A. (2016, november 30). *How Nvidia Went From Powering Video Games To Revolutionizing Artificial Intelligence*. Opgehaald van Forbes: https://www.forbes.com/sites/aarontilley/2016/11/30/nvidia-deep-learning-ai-intel/?sh=4f92c88e7ff1

Toronto Edu. (sd). *Cifar-10 & Cifar-100 dataset*. Opgehaald van Toronto Edu: https://www.cs.toronto.edu/~kriz/cifar.html

Tsung-Yi. (2018, februari 7). *Focal Loss for Dense Object Detection*. Opgehaald van arXiv: https://arxiv.org/abs/1708.02002

Ultralytics github page. (sd). Opgehaald van Github: https://github.com/ultralytics/ultralytics

VGGNet. (sd). Opgehaald van Viso: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

Wagh, A. (2022, juli 26). *Gradient Descent and its Types*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Wang, C.-Y. (2021, mei 2010). *You Only Learn One Representation: Unified Network for Multiple Tasks*. Opgehaald van arXiv: https://arxiv.org/abs/2105.04206

Yani, M. (2019, mei). *ResearchGate*. Retrieved from ResearchGate: https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451

Yavartanoo, M. (2019, januari 24). *SPNet: Deep 3D Object Classification and Retrieval using Stereographic Projection*. Retrieved from arXiv: https://arxiv.org/abs/1811.01571

Yıldız, A. F. (sd). *Object Recognition Applications in 2022*. Opgehaald van Cameralyze: https://www.cameralyze.co/blog/what-is-object-recognition-and-where-to-use

YOLO v2 – Object Detection. (022, december 6). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/yolo-v2-object-detection/

ZFNet. (2020, september 21). Opgehaald van Towards Datascience: https://towardsdatascience.com/zfnet-an-explanation-of-paper-with-code-f1bd6752121d

Zola, A. (2022, augustus). *Backpropagation algorithm*. Opgehaald van techtarget: https://www.techtarget.com/searchenterpriseai/definition/backpropagation-algorithm

OPEN UP
NEW HAN_UNIVERSITY OF APPLIED SCIENCES
HORIZONS.