Een vergelijking tussen de nauwkeurigheid van objectherkenning in schetsen door convolutionele neurale netwerken en door mensen

Bachelor Thesis (7.5 ECTS)

Kunstmatige Intelligentie

Universiteit Utrecht

Thomas Zwart

5909104

Begeleider: Dr. M. van Ommen

Tweede beoordelaar: Dr. T.B. Klos

20 augustus, 2019

Utrecht University

Abstract

Eén van de meest fundamentele manieren van communicatie is door middel van plaatjes. Het correct herkennen van deze plaatjes is daarom erg belangrijk. Binnen de kunstmatige intelligentie is het een interessante vraag of een algoritme deze taak beter kan dan mensen. In dit onderzoek wordt een convolutioneel neuraal netwerk (CNN) ontworpen, geïmplementeerd en getest om schetsen uit de Quickdraw dataset van Google te classificeren. Een groep participanten wordt gevraagd te classificeren aan de hand van een enquête. De resultaten van de classificatienauwkeurigheid van dit model worden vergeleken met de nauwkeurigheid van classificatie door mensen. Uit de resultaten blijkt dat op de gebruikte twintig categorieën uit de data het CNN significant beter presteert dan mensen, afgezien van één controleklasse die kijkt of het model in staat is te herkennen wanneer de schetsen niet tot de twintig gebruikte klassen behoren waarop het CNN slechter presteert. Wat geconcludeerd kan worden uit dit onderzoek is dat een convolutioneel neuraal netwerk dat getraind is om een beperkt aantal klassen schetsen te herkennen significant beter is in deze taak dan mensen. Er valt echter niet met zekerheid te zeggen of een convolutioneel neuraal netwerk bij toename in het aantal klassen nog steeds beter is in het herkennen van door mens getekende schetsen dan mensen zelf.

Inhoudsopgave

1. Inleiding

Nog voordat een kind leert schrijven kan hij/zij al tekeningen maken. Van jongs af aan leren mensen over objecten in de wereld te communiceren. Dit gebeurt vaak door middel van tekeningen. Nog voordat het schrift werd uitgevonden werd er gebruik gemaakt van tekeningen in communicatie, denk aan grottekeningen of het gebruik van symbolen. Het eerste schrift, het spijkerschrift, komt voort uit het vereenvoudigen van tekeningen. Tekenen is een primitieve methode van communicatie voor mensen.

Het herkennen van deze primitieve methode van communicatie door computers is binnen de kunstmatige intelligentie een belangrijke ontwikkeling omdat hiervoor de modellen van intelligentie op een fundamenteel niveau de intelligentie van mensen zullen benaderen. Mensen kiezen er namelijk voor om de meest kenmerkende eigenschappen van een object te schetsen als zij hiernaar gevraagd worden. Het is voor de kunstmatige intelligentie interessant om met een model deze schetsen te kunnen herkennen, op basis van de door mensen gekozen eigenschappen van deze objecten. Als dit lukt kunnen vervolgens deze kenmerkende eigenschappen, waaraan objecten herkend worden in schetsen, gebruikt worden om te helpen bij het herkennen van objecten in echte foto's.

Fotoherkenning is in het dagelijks leven steeds relevanter geworden. Postkantoren en banken maken hier bijvoorbeeld al gebruik van om handgeschreven karakters op brieven te herkennen. Deze relevantie is met name te danken aan het succes van convolutionele neurale netwerken door grotere datasets en krachtiger computerhardware zoals GPU's (Li et al., 2010). Convolutionele neurale netwerken (CNN) zijn het standaard algoritme voor fotoherkenning (Stenroos, 2017). Dit zijn neurale netwerken met één of meer convolutionele lagen in het netwerk. Later in dit onderzoek zal duidelijk worden wat convolutionele lagen doen. Uit resultaten blijkt dat dit soort netwerken het best presteren bij fotoherkenning. Het convolutionele neurale netwerk dat voor een lange tijd de hoogste nauwkeurigheid in correct classificeren (accuracy) haalde op de MNIST dataset (LeCun et al., 1998) werd in 2012 gepubliceerd. De MNIST dataset is een dataset met 60 duizend samples die een plaatje van één handgeschreven nummer bevat per sample. Deze data bevatten 10 categorieën/klassen, namelijk de nummers van 0 tot en met 9. In het onderzoek werd een 'multi-column convolutional deep neural network' gebruikt die een error rate van 0.23% haalde (Ciresan, Meier & Schmidhuber, 2012). Dit houdt in dat slechts 0.23% van de foto's in de MNIST dataset fout geclassificeerd werden door dit model.

Daarnaast kwam in 2009 de ImageNet dataset uit met 1.2 miljoen foto's die in 2012 op een CNN met 650 duizend neuronen en 60 miljoen parameters werd getraind en een error rate van 15.3% haalde (Krizhevsky, Sutskever & Hinton, 2012). Voor grote datasets zoals ImageNet worden convolutionele neurale netwerken vaak van tevoren getraind zodat de gewichten in het netwerk niet willekeurig geïnitialiseerd hoeven worden. Dit bespaart tijd tijdens de training van het model en is ook gedaan met het CNN in het onderzoek uit 2012 van Krizhevsky, Sutskever en Hinton. Het model dat zal worden gemaakt in het huidige onderzoek classificeert door mensen getekende schetsen. Als blijkt dat dit model accuraat kan classificeren dan kan dit relevant zijn voor het pre-trainen (van tevoren trainen) van CNN's die objecten in echte foto's moeten herkennen. De schetsen zijn namelijk door mensen getekend, wat betekent dat ze getekend zijn op basis van de kenmerkende eigenschappen van objecten (zoals hier hierboven al aangegeven). Als een netwerk van tevoren getraind wordt op de schetsen en dus op basis van deze eigenschappen van objecten, dan kan dit een positief effect hebben op het model zodra het objecten moet classificeren in echte foto's.

De schetsen die gebruikt worden in dit onderzoek komen uit de Quickdraw dataset. Eind 2016 bracht Google het spel 'Quick, Draw!' uit. Hierin worden mensen online gevraagd om een object of dier (en af en toe een concept) te tekenen. De dataset die is gegenereerd door dit spel zal gebruikt worden in dit onderzoek, de Quickdraw dataset. Dit is een open source dataset die schetsen van de objecten of dieren bevat, deze vormen samen de categorieën/klassen waaruit de data bestaat. Eerdere modellen in

onderzoek naar de Quickdraw dataset zijn onder andere een CNN die een nauwkeurigheid (accuracy) van 62.1% haalde op alle 345 klassen in de dataset (Guo, WoMa & Xu, 2018). Dit houdt in dat 62.1% van de classificaties door dit model op de dataset correct waren. In dit onderzoek werd echter wel slechts 1% per categorie van de originele dataset gebruikt en daarbij werd alle gebruikte data slechts 20 keer aan het netwerk als input gegeven om te trainen. Met meer data en meer training kan dit resultaat waarschijnlijk verbeterd worden. In dit onderzoek zullen minder klassen worden gebruikt, maar meer data per klasse en zal er vaker getraind worden.

Daarnaast is het resultaat uit het onderzoek van Guo, WoMa en Xu niet vergeleken met classificatie door mensen. Om te weten of sommige classificatietaken die vergelijkbaar zijn met het herkennen van schetsen geautomatiseerd kunnen worden, is het relevant om te weten of een CNN beter kan classificeren dan mensen. Ook is het van academisch belang binnen de kunstmatige intelligentie om erachter te komen welke van de twee entiteiten beter is in classificeren. Er zijn veel niveaus waarop cognitieve neurowetenschappers neurale netwerken kunnen gebruiken, bijvoorbeeld modellen bedenken op basis van theoretische literatuur of nieuwe modellen maken om cognitieve theorieën te testen (Storrs & Kriegeskorte, 2019). Als blijkt dat een CNN beter presteert in herkennen van objecten in schetsen dan mensen, dan kan dit model gebruikt worden binnen neuropsychologie om meer inzicht te krijgen in hoe menselijk zicht en patroonherkenning werkt.

In dit onderzoek zal een CNN ontworpen, geïmplementeerd en getest worden met de data van de Quickdraw dataset. Een verwijzing naar de implementatie van het CNN is te vinden in de appendix, sectie 7. Vervolgens zullen de prestaties van deze CNN vergeleken worden met de prestaties van mensen in het classificeren van schetsen uit de dataset. Het doel van het onderzoek is om een goed classificatiemodel te maken en op basis van de resultaten te kijken of dit model de classificatietaak met hoge nauwkeurigheid kan voltooien. Vervolgens om een conclusie te trekken op basis van de vergelijking tussen het model en de mensen, namelijk welke van de twee entiteiten beter presteert in het herkennen van door de mens getekende schetsen.

Onderzoeksvraag

De volgende vraag zal beantwoord worden in dit onderzoek:

> *'Hoe verhoudt de nauwkeurigheid van schetsclassificatie door convolutionele neurale netwerken zich met die door mensen gebruikmakend van Google's quickdraw dataset?'*

2. Data

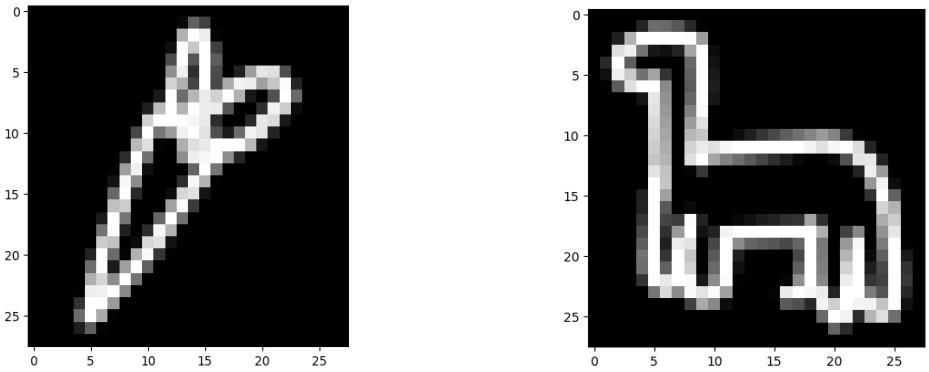
Zoals in de inleiding vermeld, wordt in dit onderzoek gebruik gemaakt van de Quickdraw dataset (Googlecreativelab, 2017). Deze dataset is een verzameling van tekeningen die door mensen over de hele wereld zijn gemaakt op https://quickdraw.withgoogle.com/. In dit spel krijgen mensen twintig seconden de tijd om een object of dier (of af en toe een concept) te tekenen.

De dataset bevat 345 klassen met ongeveer 120.000 schetsen per klasse. Klassen zijn de verschillende categorieën uit de data. Omdat de klassen ook door mensen geclassificeerd worden tijdens de enquête in dit onderzoek zijn er slechts 20 van de 345 klassen gebruikt. Dit aantal is redelijk om van mensen te verwachten te kunnen classificeren. Voor het algoritme is er een 21ste 'null-klasse' toegevoegd, deze bevat verschillende schetsen van 30 andere (nog niet gebruikte) klassen.

De 'null-klasse' is toegevoegd om het neurale netwerk geen oneerlijk voordeel te geven ten opzichte van de mens. Het neurale netwerk wordt zonder deze klasse namelijk op de twintig klassen getraind waar ook op getest zal worden en 'weet' als het ware uit welke klassen het kan kiezen. Dit kan worden gezien als een multiple-choice optie voor het algoritme. Aangezien de mensen deze multiple-choice optie niet hebben, moet het algoritme deze ook worden ontnomen om een eerlijke vergelijking te kunnen maken. Een manier om dit probleem tegemoet te komen is een 'null-klasse' met onbekende schetsen toe te voegen. Door het algoritme op 21 klassen te trainen waaronder de 'null-klasse' heeft het CNN de kans om schetsen niet te herkennen en te classificeren als 'null-klasse'. Het doel van de 'null-klasse' is niet om te zien hoe goed het algoritme kan ontdekken of de schetsen tot deze klasse behoren, maar om te zien of het algoritme kan ontdekken of bepaalde schetsen niet tot één van de andere twintig (hoofd)klassen behoren. De bedoeling is dat het in de 'null-klasse' voor het neurale netwerk lastig is een patroon te ontdekken, of zelfs helemaal niet te ontdekken, aangezien er zoveel verschillende categorieën schetsen in zitten. Wanneer het neurale netwerk vervolgens wordt getest en een schets tegenkomt die niet tot één van de twintig (hoofd)klassen behoort dan is de hypothese dat het netwerk deze zal classificeren met de 'null-klasse'.

Er zijn verschillende representaties van de schetsen in de data. In dit onderzoek wordt de representatie van 28 bij 28 pixels gebruikt als een array van 784 waarden voor elke schets. Elke schets is gelabeld met een 'one-hot' label. Dit label wordt vaak gebruikt bij neurale netwerken met classificatietaken. Een 'one-hot' label is een array *A* van *n* waarden waarbij *n* gelijk is aan het aantal verschillende soorten klassen in de data waarop het neurale netwerk getraind wordt. Er wordt gebruik gemaakt van een array omdat *n* een vast aantal is, de labels van tevoren worden gemaakt en niet meer veranderen. Elke index in de array *A* correspondeert met een unieke klasse uit de data. Er is een bijectieve functie tussen de indices in de array en de verschillende klassen in de data, die van tevoren wordt bepaald. Hoe deze functie eruitziet maakt niet uit voor het trainen van het netwerk zolang elke unieke index maar bij een andere unieke klasse hoort. Het is slechts een representatie van een label dat handig is voor het neurale netwerk bij het trainen. Op elke index heeft de array als waarde 0 behalve op één index *i*, waar de array de waarde 1 zal hebben. Dit betekent dat het 'one-hot' label bij de klasse hoort die correspondeert met index *i*.

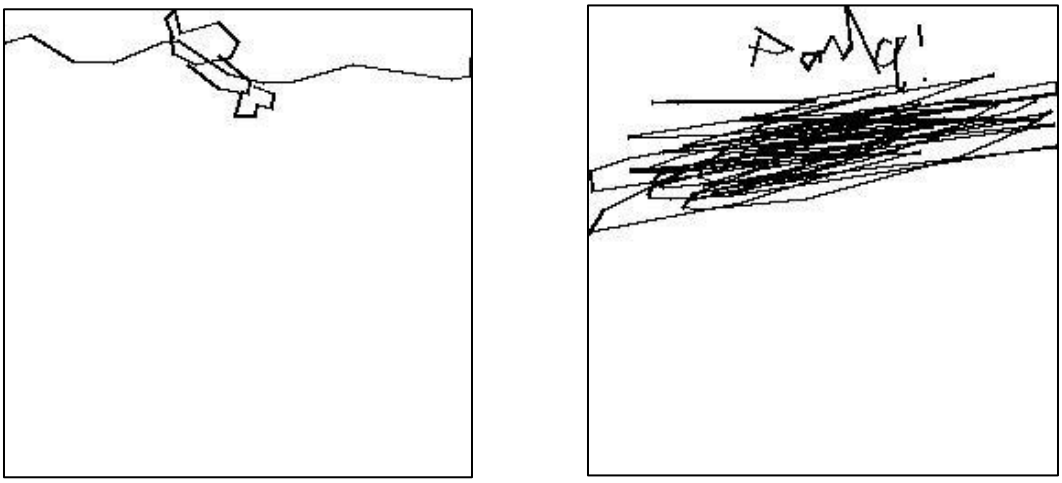
In dit onderzoek bestaat het 'one-hot' label uit 21 waardes, aangezien er 21 klassen worden gebruikt voor het trainen van het neurale netwerk. Bijvoorbeeld alle schetsen die tot de klasse 'mier' behoren hebben de volgende 'one-hot' label met lengte 21: [1, 0, 0, …, 0, 0, 0] en schetsen die tot de klasse 'brood' behoren de volgende 'one-hot' label met lengte 21: [0, 1, 0, …, 0, 0, 0]. Van tevoren wordt de bijectieve relatie tussen indices en klassen bepaald, dus welke index in het label correspondeert met welke klasse.



Figuur 1: De 28 bij 28 pixel representatie van de data zoals het algoritme de data ziet. Een wortel (links) en een giraffe (rechts).

Voor de 20 klassen uit de data plus de ‘null-klasse’ (die uit 30 andere klassen bestaat) zullen per klasse 40.000 schetsen worden gebruikt in de trainingsfase van het netwerk. Dit resulteert in 840.000 schetsen in totaal. Deze worden gemixt en opgesplitst in trainingsdata van grootte 800.000 en validatiedata van grootte 40.000. Gezien de omvang van de data kan ervan uit worden gegaan dat er sprake is van een vrijwel gelijke verdeling van de samples over de klassen. Kleine statistische variaties op deze verdeling zullen geen effect hebben op de validiteit van dit onderzoek. De uiteindelijke testdata zal bestaan uit 1000 nieuwe samples van elke schets en 1000 van de ‘null-klasse’, wat uitkomt op 21.000 samples als testdata.

Doordat de dataset ongecontroleerd is ontstaan uit tekeningen die mensen op een website maken bevat deze ruis. Bijvoorbeeld door tekeningen van mensen die niet geprobeerd hebben het object te tekenen dat werd gevraagd of die hun tekening slechts half afgemaakt hebben. Ook zijn er voorbeelden waar heel klein is getekend, dit zal in een 28 bij 28 pixelrepresentatie resulteren in een tekening van slechts een paar pixels en hiermee zal het CNN niet veel kunnen. Google heeft de dataset schoon proberen te houden maar er zal altijd ruis in blijven zitten. Van deze ‘slechte’ schetsen zal niet van de participanten worden geacht ze te classificeren, maar het CNN moet er wel mee omgaan.



Figuur 2: Voorbeelden van ruis in de data. Een te klein getekende dolfijn (links) en een panda (rechts).

3. Methode

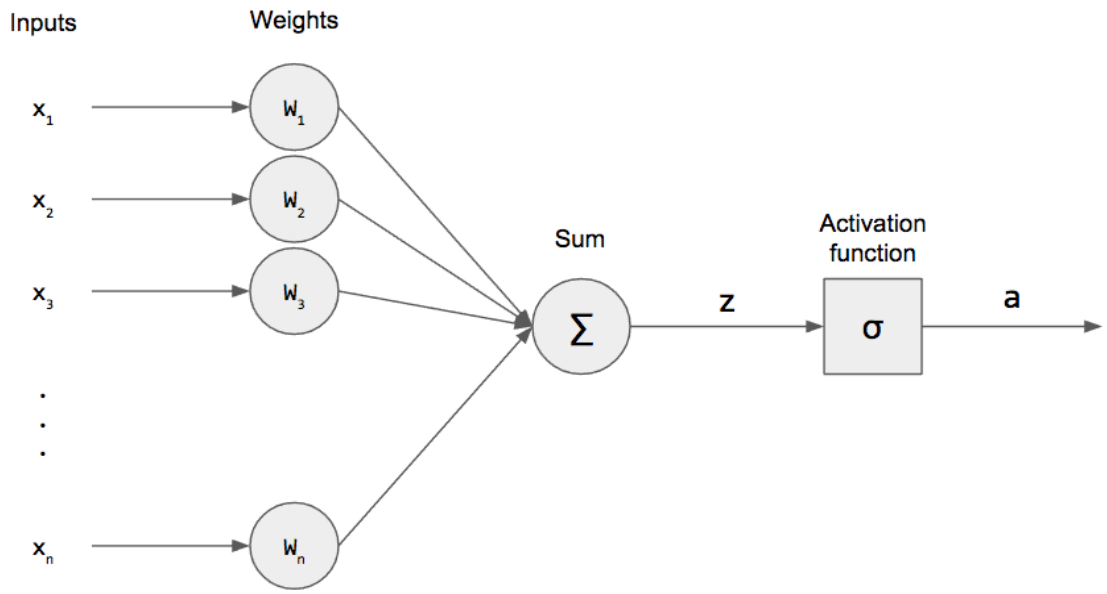
In dit onderzoek wordt een CNN ontworpen, geïmplementeerd en getest. Dit model wordt getraind met de trainingsdata zoals in sectie 2 beschreven. Alle trainingsdata wordt in groepen (batches) aan het netwerk gegeven en per groep worden de gewichten in het netwerk aangepast zodat het model beter presteert. Het trainen gebeurt in epochs, één epoch houdt in dat het CNN éénmaal op alle trainingsdata is getraind. Het aantal epochs blijft toenemen totdat er geen toename in nauwkeurigheid van het model meer is op de validatiedata en dat wordt het trainen gestopt. In sectie 3.1 tot en met 3.5 is beschreven hoe dit werkt. Ook is beschreven hoe het netwerk is ontworpen en welke keuzes zijn gemaakt wat betreft de gebruikte functies, algoritmes en parameters om tot een optimaal ontworpen model te komen. In de appendix, sectie 7, staat additionele informatie over hoe het model is getraind en een link naar de implementatie van het CNN.

Voor de parameters batch size en dropout rate worden experimenten uitgevoerd om het CNN te optimaliseren met gebruik van de validatiedata. Wat batch size en dropout rate zijn wordt respectievelijk in sectie 3.4 en 3.5 uitgelegd. Door het model tijdens het trainen te testen op de validatiedata is te zien welke waarden goed presteren voor deze twee parameters. In sectie 4.1 en 4.2 zijn de resultaten van deze experimenten beschreven.

Als het model eenmaal geoptimaliseerd is dan wordt het nog eenmaal getraind op de trainingsdata en opgeslagen. Vervolgens wordt de nauwkeurigheid van dit model vergeleken met die van de mensen. Beide entiteiten moeten dezelfde twintig klassen schetsen classificeren. Het model moet daarbij ook nog de ‘null-klasse’ classificeren. De testdata zoals in sectie 2 beschreven wordt nu gebruikt om het model te testen en te zien hoe nauwkeurig het model de schetsen classificeert. Aan de hand van een enquête wordt de nauwkeurigheid van de classificatie door mensen getest. Hoe deze enquête wordt afgenomen staat beschreven in sectie 3.6. De resultaten van het model en de enquête worden vergeleken en op basis hiervan zal een conclusie worden getrokken hoe classificatie van schetsen zich verhoudt tussen deze twee entiteiten.

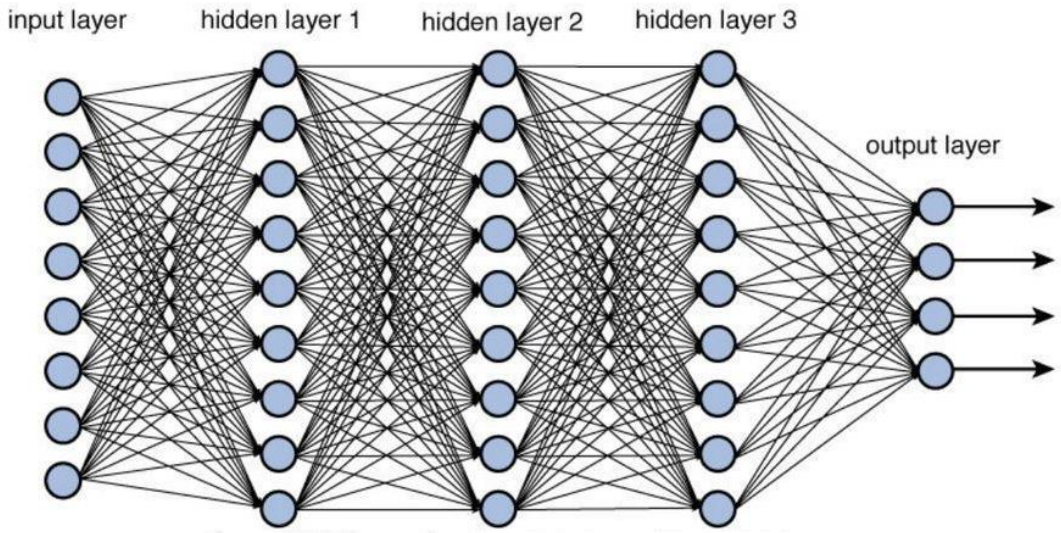
3.1 Neuraal netwerk

In 1943 introduceerde McCulloch & Pitts een model van een kunstmatig neuron, dat nog steeds wordt gebruikt in neurale netwerken. Het modelleert een neuron als de som van de inputs met gewichten en een activatie functie (McCulloch & Pitts, 1943). Dit werd verder uitgewerkt tot een perceptron, een algoritme dat lineair scheidbare data kan classificeren, vervolgens tot een multilayer perceptron die ook niet-lineair scheidbare data kan classificeren. Een multilayer perceptron kan gezien worden als een standaard neuraal netwerk. Daarna werd in 1986 experimenteel aangetoond door Rumelhart, Hinton & Williams dat een backpropagation algoritme een goede manier is om de gewichten te updaten in neurale netwerken (Rumelhart, Hinton & Williams, 1986). Hierdoor werden neurale netwerken rendabel om te gebruiken binnen de machine learning.



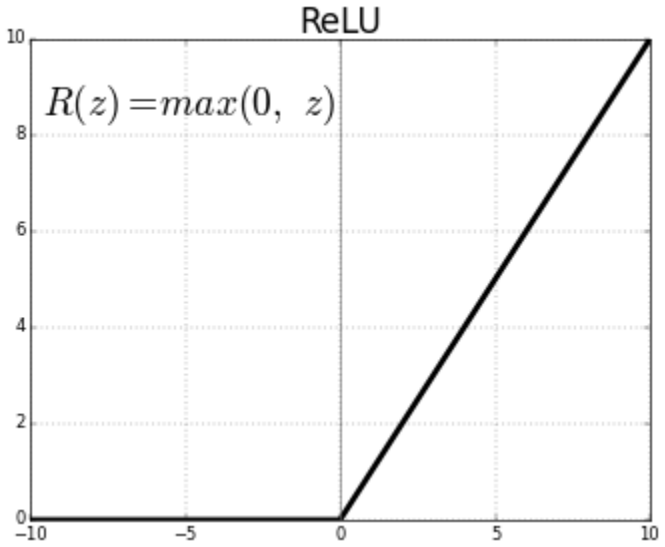
Figuur 3: Schematisch voorbeeld van een perceptron.

Een neuraal netwerk is een collectie van verbonden kunstmatige neuronen (nodes). De nodes zijn vaak geordend in lagen, de input laag, verborgen lagen (hidden layer) en de output laag (zie figuur 4). Er is geen vast aantal verborgen lagen of een vast aantal nodes per laag, dit varieert per model. Nodes kunnen via een connectie, net als synapsen in een biologisch brein, een signaal naar andere nodes sturen. Dit signaal is in standaard implementaties van neurale netwerken vaak een reëel getal. Het signaal dat een node verstuurd wordt vermenigvuldigd met het gewicht dat bij de connectie hoort waarover het signaal wordt gestuurd. Een node kan meerdere signalen ontvangen van de vorige laag. Al deze signalen worden opgeteld en de som ervan is de input van de activatiefunctie die bij de node hoort. De output van de activatiefunctie is het signaal dat de node als output zal geven. Deze output wordt verstuurd naar alle nodes in de volgende laag waarmee de node die het signaal stuurt verbonden is.



Figuur 4: Een conventioneel neuraal netwerk met een input laag (input layer), verborgen lagen (hidden layers) en een output laag (output layer).

De activatiefunctie die gebruikt wordt in onze CNN is de ‘rectified lineair unit (ReLU)’ functie (zie figuur 5). Deze functie is het meest gebruikt voor neurale netwerken en heeft de beste resultaten tot op heden (Nwankpa et al, 2018). Het grote voordeel van deze activatiefunctie is dat het niet alle nodes tegelijkertijd activeert, maar dat sommige nodes geen signaal als output zullen hebben. Dit kan bijvoorbeeld gebeuren als er negatieve gewichten zijn toegekend aan bepaalde connecties waardoor de ReLU functie een negatieve input kan ontvangen. Hierdoor is de ReLU activatiefunctie computationeel efficiënt en convergeert in praktijk tot wel zes keer sneller dan andere activatiefuncties (Udofia, 2018).



Figuur 5: De formule en grafiek van de ReLU functie.

3.2 Convolutionele- en pooling lagen

Kenmerkend aan een convolutioneel neuraal netwerk zijn de convolutionele lagen waarin convoluties worden uitgevoerd op de input. Deze lagen worden gebruikt om patronen te ontdekken in de schetsen, dit heet ook wel 'feature extraction'. Verderop in deze sectie wordt uitgelegd hoe convoluties werken.

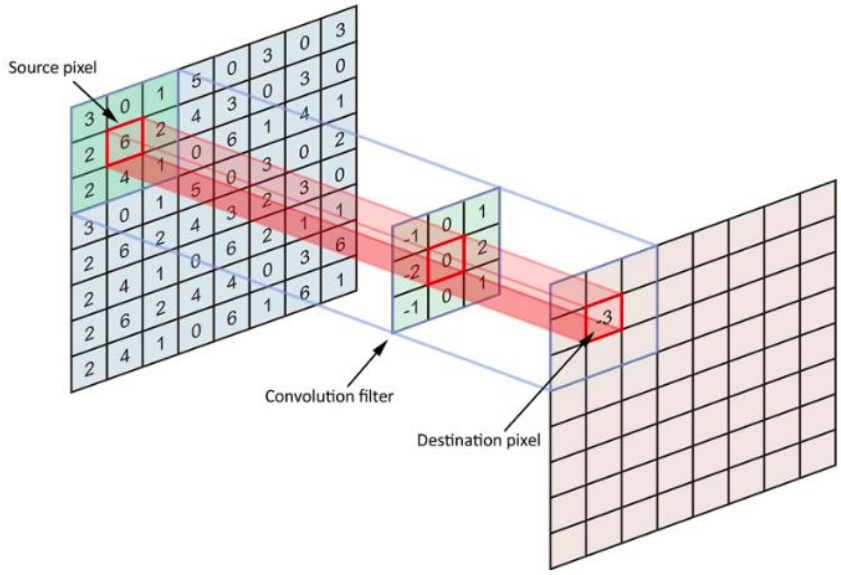
Anders dan de lagen in een standaard neuraal netwerk hebben convolutionele lagen geen nodes. De gewichten in deze laag zijn de waarden in de filters die gebruikt worden om convoluties uit te voeren. Deze filters worden gebruikt om patronen te ontdekken in de schetsen. De input van een convolutionele laag is in dit onderzoek een schets (input-schets), of meerdere schetsen, gerepresenteerd als een array. De output van een convolutionele laag zijn meerdere featuremaps. Featuremaps is een benaming van de schets nadat er convoluties op zijn uitgevoerd, omdat deze operatie de kenmerken (features) uit de schets zichtbaar maakt. Er kunnen meerdere convolutionele lagen achter elkaar worden toegepast om complexere patronen in de input te ontdekken.

Een convolutie is het inwendig product van twee matrices volgens de onderstaande formules. De waarden van deze matrices zijn de waarden van de pixels van een input-schets (waar het filter overheen ligt) en de waarden van de gewichten in één van de filters van de convolutionele laag. De filter is een twee dimensionaal rooster van *a* bij *a* met gewichten dat over de pixels van de schets heen beweegt. Als op elke pixel van de input I een convolutie wordt toegepast met een filter dan is output *O* een featuremap van de input-schets I. Zie figuur 6 voor een visualisatie van een convolutie op één pixel.

$$z = \frac{a}{2} - 1$$

$$O_{x,y} = \sum_{n=-z}^{z} \sum_{m=-z}^{z} F_{n,m} I_{x-n,y-m}$$

a = De breedte van de filter.
I = De input schets als twee dimensionale array.
F = De filter als twee dimensionale array.
O = De output featuremap als twee dimensionale array.



Figuur 6: Een visualisatie van een 3 bij 3 convolutie uitgevoerd op één pixel (Dertat, 2017).

Een convolutionele laag heeft meerdere filters die worden gebruikt voor convolutie op de input. Hierdoor worden de schetsen, of een enkele schets, die als input in de laag gaan omgezet in meerdere featuremaps, één featuremap per filter. Meerdere filters zijn nodig om verschillende patronen in de schetsen te kunnen ontdekken. Combinaties van deze patronen, zoals hoeken of lijnen, kunnen een indicatie zijn dat een schets tot een bepaalde klasse hoort. Onze CNN heeft in de eerste convolutionele laag 32 filters. Er gaat één input naar die laag, de schets, en als output heeft de laag 32 featuremaps waarop de convoluties zijn toegepast. De tweede convolutionele laag in onze CNN krijgt als input de 32 featuremaps van de vorige laag en heeft 64 filters, dus 64 featuremaps als output (zie 3.3).

De gewichten in de filters worden geïnitialiseerd met willekeurige gewichten uit een normaalverdeling in het begin van training. Na elke batch zullen met een optimizer de gewichten van de filters aangepast worden om de verliesfunctie te minimaliseren (zie 3.4). De filters worden hierdoor steeds beter geschikt voor het herkennen van de juiste patronen; het model leert zelf deze patronen ontdekken. Voorbeelden van dit soort patronen zijn lijnen, hoeken en soms combinaties hiervan. Door meerdere convolutionele lagen achter elkaar te zetten is het mogelijk om gehele onderdelen van bepaalde schetsen te herkennen.

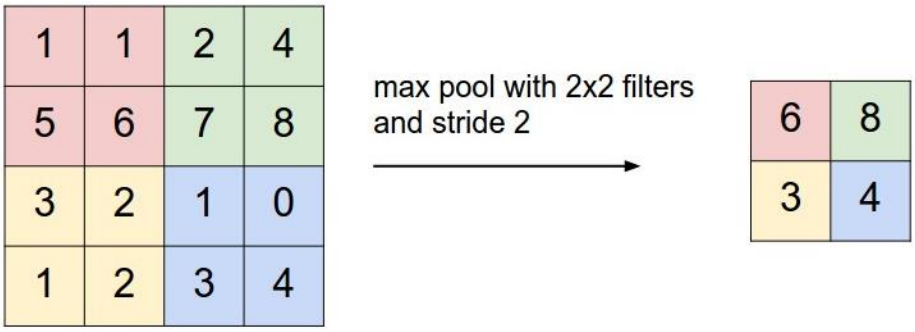
De grootte van de filter wordt van tevoren bepaald. De grootte van het filter dat gebruikt wordt in onze CNN is 5 bij 5. Aangezien alle schetsen in twintig seconden zijn getekend kan de aanname gemaakt worden dat er weinig detail zit in de schetsen. Een filter van 5 bij 5 is groot genoeg om met deze ruwe schetsen om te gaan en klein genoeg zodat er niet veel gegeneraliseerd wordt en er nog wel specifieke patronen ontdekt kunnen worden.

De stapgrootte waarmee de filter over de pixels van de schets heen beweegt is één, hierdoor zal de input van 28 bij 28 pixels na convolutie een featuremap van nog steeds 28 bij 28 pixels worden, aangezien voor elke pixel in de input een convolutie wordt gedaan en deze in de output wordt geplaatst. Een probleem doet zich voor wanneer de filter bij de rand van de schets komt, aangezien het midden van de filter bij de randpixel van de schets zit en gewichten aan één (of twee) kant(en) van de filter geen pixel in de schets hebben om mee te vermenigvuldigen. Er zal dan gebruik gemaakt worden van ‘zero-padding’. Dit houdt in dat de niet gedefinieerde waarden buiten de schets als 0 gezet zullen worden. Als de filter aan de rand van de schets komt dan worden de gewichten aan de kant(en) van de filter die buiten de schets valt vermenigvuldigd met 0.

Na elke convolutionele laag is er een pooling laag waar max pooling plaatsvindt. Max pooling zorgt ervoor dat de featuremaps steeds kleiner worden terwijl de belangrijkste informatie behouden blijft. Dit is nodig voor datareductie zodat de computationele kracht die nodig is om de data te verwerken minder is (Thoma, 2017). Er beweegt bij max pooling een twee dimensionaal rooster over de featuremap. De waarden van de featuremap waarover dit rooster ligt zijn de input voor de max pooling operatie en de output is de hoogste van die waarden. In onze CNN wordt een rooster van 2 bij 2 gebruikt met een stapgrootte van 2. De onderstaande formule beschrijft hoe per pixel een 2 bij 2 max pooling operatie wordt uitgevoerd. Als dit met een stapgrootte van 2 over de hele featuremap wordt gedaan dan is het resultaat een featuremap die gereduceerd is met een factor van 2. Zie figuur 7 voor een visualisatie van max pooling.

$$O_{x,y} = \max\left(F_{2x,2y}, F_{2x+1,2y}, F_{2x,2y+1}, F_{2x+1,2y+1}\right)$$

F = De input featuremap als twee dimensionale array.
O = De output featuremap als twee dimensionale array.

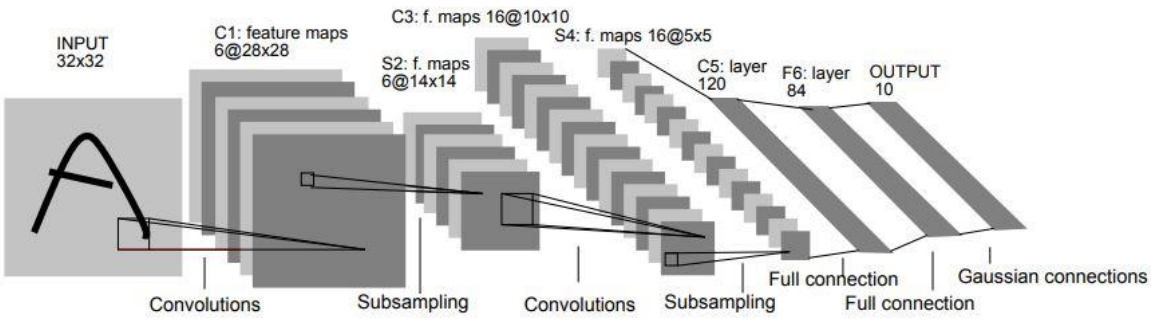


Figuur 7: Een visualisatie van 2 bij 2 max pooling met een stapgrootte (stride) van 2.

De eerste pooling laag ontvangt de featuremaps van de convolutionele laag ervoor, featuremaps van 28 bij 28 pixels. Na de eerste pooling laag zullen deze featuremaps gereduceerd zijn naar 14 bij 14 pixels door max pooling. Bij de tweede convolutionele laag gevolgd door max pooling gebeurt hetzelfde proces en zijn de featuremaps als output 7 bij 7 pixels groot.

3.3 Convolutionele netwerkarchitectuur

In dit onderzoek wordt een netwerkarchitectuur gebruikt die te vergelijken is met LeNet-5. Dit is een CNN ontworpen door Lecun, Bottou, Bengio en Haffner in 1998 om handgetekende nummers te classificeren uit de MNIST dataset (Lecun et al, 1998). Het netwerk bestaat uit een twee convolutionele lagen met respectievelijk 6 en 16 filters gevolgd door 2 bij 2 pooling, daarna twee volledig verbonden lagen van respectievelijk 120 en 84 nodes. Zie figuur 8 voor een visualisatie van de netwerkarchitectuur van LeNet-5. Het netwerk haalde een error rate van 0.8% op de MNIST dataset (Thoma, 2017). Dit houdt in dat slechts 0.8% van de tekeningen uit de data fout geclassificeerd werden. De MNIST en Quickdraw dataset zijn in veel aspecten vergelijkbaar, beide zijn ze gerepresenteerd met greyscale waarden (kleurloos) en door mensen getekend. De data hebben vergelijkbare dimensies namelijk 32 bij 32 pixels in MNIST en de schetsen in dit onderzoek 28 bij 28 pixels. Er is slechts één van de klassen getekend in de plaatjes op een witte achtergrond, bij de MNIST dataset betreft dit één nummer en bij de Quickdraw dataset één object of dier. Er is daardoor geen ruis op de achtergrond dat classificatie lastiger maakt. In verband met deze gelijkenis tussen de data en het succes van LeNet-5 is er gekozen voor een vergelijkbare netwerkarchitectuur voor het CNN dat is ontworpen in dit onderzoek.



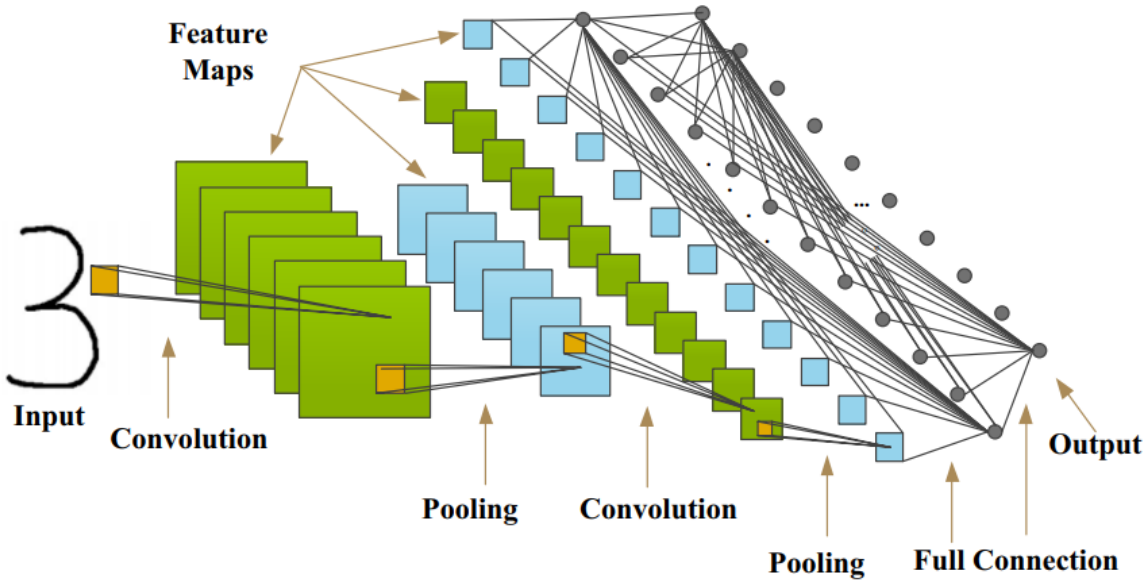
Figuur 8: Een visualisatie van de netwerkarchitectuur van LeNet-5 (Lecun et al, 1998).

Onze CNN begint met twee convolutionele lagen die beide zijn gevolgd door een pooling laag. De eerste convolutionele laag ontvangt één input, namelijk de schets van 28 bij 28 pixels, en doet hier aan de hand van 32 filters convoluties op. Als output heeft deze laag 32 featuremaps die na max pooling 14 bij 14 pixels groot zijn. De tweede convolutionele laag ontvangt deze 32 featuremaps en doet hier convoluties op met 64 filters. Als output heeft deze laag 64 featuremaps die na max pooling 7 bij 7 pixels groot zijn. Met deze gereduceerde featuremaps heeft verdere convolutie geen effect meer (Simard, Steinkraus & Platt, 2003). In LeNet-5 zijn de outputs van de twee convolutionele lagen respectievelijk 6 en 16 featuremaps en in ons netwerk respectievelijk 32 en 64 featuremaps. Dit is omdat meer filters nodig zijn voor het model in dit onderzoek aangezien er niet alleen meer klassen zijn, namelijk 21 tegenover de 10 uit MNIST, maar ook aangezien schetsen van objecten of dieren meer variëren dan handgetekende nummers. Er zijn dus meer filters nodig om met deze variatie om te gaan.

Na de convolutionele lagen met max pooling komt een enkele volledig verbonden laag. Als een convolutionele laag gevolgd wordt door een volledig verbonden laag is het nodig de featuremaps te vectorizeren (Thoma, 2017), zodat de waarden van de featuremaps naar nodes in de volledig verbonden laag gestuurd kunnen worden. Dit houdt in dat alle 64 featuremaps van 7 bij 7 pixels die als output uit de tweede convolutionele laag met max pooling komen naar de onderstaande, eendimensionale vector moeten worden omgezet. Hierin zijn *x* de waarden van de pixels in de featuremaps.

$$[x_1, x_2, \ldots, x_{64*7*7}]$$

De waarden uit deze vector worden elk naar alle nodes van de volledig verbonden laag met 1024 nodes gestuurd. ‘Volledig verbonden’ omdat elke waarde uit de vector een connectie met een gewicht met elke node heeft. Als laatste is deze laag van 1024 nodes verbonden met de output laag van 21 nodes. Deze 21 nodes representeren de 21 verschillende klassen in de data waarop het netwerk wordt getraind. Bij het testen van het netwerk op een schets zal één van deze nodes de hoogste waarde krijgen en dan zal die schets geclassificeerd worden met de klasse die correspondeert met die node.



Figuur 9: Een schematisch voorbeeld van de gebruikte CNN architectuur (Li et al., 2016).

3.4 Adam optimizer, batch size & verliesfunctie

Het doel in de trainingsfase is om de uitkomst van de verliesfunctie, het verlies, te minimaliseren. Bij een lager verlies liggen de voorspellingen van het model en de correcte labels dichter bij elkaar volgens de verliesfunctie. Er wordt gebruik gemaakt van een optimalisatiealgoritme dat, voor elke batch, de gewichten van de convolutie filters en de gewichten tussen de lagen van het netwerk update. Het algoritme zal de gewichten updaten aan de hand van de gradiënt van de verliesfunctie, zodat de uitkomst van de verliesfunctie wordt geminimaliseerd.

Softmax cross-entropy is de verliesfunctie die in dit onderzoek is gebruikt (Bendersky, 2016). Eerst wordt op de voorspellingen van het CNN de softmax functie toegepast. Vervolgens wordt op het resultaat van de softmax functie en de correcte labels de cross-entropy functie gebruikt om het verlies te berekenen. Softmax cross-entropy is één van de standaard verliesfuncties voor classificatieproblemen (Lu & Tran, 2017).

Voor optimalisatie van de gewichten in het filter en tussen de lagen van het netwerk wordt de Adam optimizer gebruikt (Kingma & Ba, 2015). Adam is een optimalisatiealgoritme dat twee voordelen van andere optimalisatie methodes combineert (Kingma & Ba, 2015). Het vermogen van het Adagrad optimalisatiealgoritme om met kleine gradiënten om te gaan, als de gradiënt van de verliesfunctie erg klein is heeft het optimalisatiealgoritme meer moeite met het correct updaten van de gewichten omdat het minder duidelijk is in welke richting de gradiënt beweegt. En het vermogen van het RMSProp optimalisatiealgoritme om met bewegende minima om te gaan, dit kan gebeuren doordat verschillende batches een variërend minima kunnen hebben en er per batch wordt geoptimaliseerd. Ook kan er ruis in de data zijn waardoor de minima verschuift. Adam optimizer kwam als beste uit de bus in een experiment gedaan door David Mack, die verschillende optimizers heeft vergeleken aan de hand van een CNN (Mack, 2015). De standaard aanbevolen parameters van Adam zijn gebruikt in dit onderzoek (Kingma & Ba, 2015).

Parameter	Waarde
Learning Rate (α)	0.001
β_1	0.9
β_2	0.999

Tabel 1: Parameters gebruikt voor de Adam optimizer.

Tijdens de training wordt alle data meerdere malen door het netwerk gestuurd. Elke keer als éénmaal op alle data is getraind is dit één epoch. Gezien de grote van de dataset kan alle trainingsdata niet in één keer door het netwerk gestuurd worden. Daarom wordt de data door het netwerk gestuurd gesepareerd in aparte groepen (batches). De batch size is de hoeveelheid datapunten die per iteratie door het neurale netwerk heengaan en waar de optimizer per keer het verlies op probeert te minimaliseren. Het aantal iteraties is gelijk aan de grootte van de trainingsdataset gedeeld door de batch size. Een lagere batch size is preciezer en haalt over het algemeen een hogere accuracy (tenzij deze te laag is, zie 4.1). Accuracy is de nauwkeurigheid waarin het CNN de schetsen classificeert. Een lagere batch size duurt echter langer om te trainen, aangezien er meer iteraties nodig zijn voordat de hele trainingsdataset door het netwerk is gepasseerd en de hardware (GPU) niet optimaal wordt gebruikt (Thoma, 2017). Het is essentieel een juiste batch size te vinden gegeven de trainingsdata. Deze moet zo hoog mogelijk zijn dat het niet tot verlies van accuracy leidt maar wel snel is. In sectie 4.1 van de resultaten is met verschillende batch sizes geëxperimenteerd.

3.5 Regularization

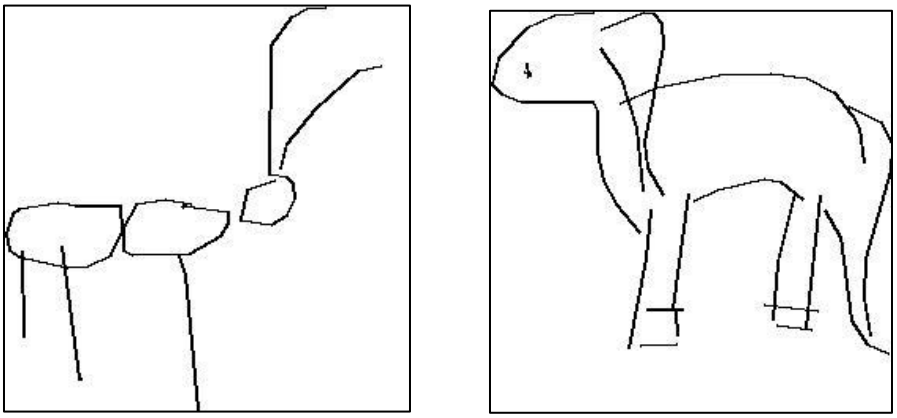
Neurale netwerken hebben veel variabelen in de vorm van gewichten die geoptimaliseerd worden. Dit maakt ze erg capabel om lastige relaties te leren uit de input die ze krijgen. Maar het gevaar is dat er relaties worden gevonden als resultaat van ruis in de trainingsdata of dat er een toevallige relatie in de trainingsdata zit. Het neurale netwerk leert dan een relatie uit de trainingsdata die niet zal bestaan in de testdata, dit heet overfitting. Dit gebeurt zelfs als de trainingsdata en testdata uit dezelfde distributie zijn gehaald (Srivastava et al. 2014).

De term 'regularization' refereert naar de methoden die gebruikt worden om overfitting te reduceren. Overfitting is te voorkomen door de hoeveelheid data te vergroten. Het is dan minder aannemelijk dat er patronen in de ruis gaan zitten of dat er toevallig een unieke relatie in de trainingsdata zit. Daarnaast is het belangrijk om het trainen van het model op tijd te stoppen, zodra er lange tijd geen vooruitgang meer in accuracy is. Langer doortrainen zal alleen maar overfitten zijn en derhalve niet tot beter resultaat leiden op de testdata. Een klein neuraal netwerk is minder goed in staat om complexere relaties te ontdekken doordat er minder gewichten in zitten. De relaties die een klein model ontdekt generaliseren beter en zullen minder snel relaties zijn die in de ruis van de data zitten (Abu-Mostafa et al., 2012). Dit is de reden dat kleine netwerken beter bestand zijn tegen overfitting, hierbij moet wel opgepast worden dat er geen underfitting plaatsvindt. Underfitting is als de patronen in de data die wel ontdekt moeten worden niet gevonden worden door het netwerk. Hoe groot het CNN moet zijn hangt af van hoe complex de data zijn en hoeveel klassen er geclassificeerd moeten worden. De data in dit onderzoek zijn relatief eenvoudig vergeleken met de data uit een dataset als ImageNet, onze data zijn gerepresenteerd met greyscale waarden (kleurloos) en slechts 28 bij 28 pixels. Gegeven de data in dit onderzoek is een klein model adequaat, vandaar dat is gekozen voor de netwerkarchitectuur beschreven in sectie 3.3. Voor een CNN zijn twee convolutionele lagen met respectievelijk 32 en 64 featuremaps als output, gevolgd door één volledig verbonden laag van 1024 nodes, relatief klein. Zeker in vergelijking met het CNN van Krizhevsky, Sutskever & Hinton (2012) op de ImageNet dataset met 650 duizend neuronen.

Desalniettemin kan overfitting nog steeds plaatsvinden en daarom wordt in dit onderzoek geëxperimenteerd met dropout. Dit is de meest gebruikte 'regularization' techniek voor neurale netwerken en werd voor neurale netwerken geïntroduceerd door Srivastava et al. (2014).
Dropout vermindert het aantal actieve neuronen in een laag tijdens training door de output van een neuron op 0 te zetten met een kans p die van tevoren wordt gekozen. Bij een convolutionele laag wordt een featuremap weggelaten, voor elke featuremap in de output, met een kans p. Hierdoor heeft de volgende laag minder data en dat zorgt ervoor dat er minder kans is op overfitting. De motivatie voor dropout is dat elke node in een neuraal netwerk moet leren omgaan met een willekeurig gekozen sample van andere nodes. Dit zal elke node meer robuust maken en hem richten op het ontwikkelen van goede gewichten, zonder op andere nodes te steunen om zijn fouten te corrigeren (Srivastava et al., 2014).

3.6 Enquête

Het menselijk onderzoek is gedaan aan de hand van een online enquête. In die enquête werden participanten gevraagd om de originele schetsen te herkennen.



Figuur 10: Originele onaangepaste schetsen zoals participanten ze te zien kregen.
Mier (links) en een paard (rechts).

De participanten werden gevraagd van alle 20 gebruikte klassen één schets te herkennen. De volgende vraag werd bij elke schets gesteld: “Wat ziet u hier?”. Hierop mocht in een open tekstvak geantwoord worden. Er zat geen tijdslimiet op de enquête, maar uit resultaat blijkt dat iedere participant er minder dan twee minuten over deed. Het aantal participanten dat heeft meegedaan aan dit onderzoek is 92. De leeftijd van deze participanten varieert van 16 tot 80 jaar oud. Geen van de participanten heeft een visuele of mentale beperking, waardoor zij niet in staat zouden zijn tot het herkennen van objecten of dieren in de schetsen.

Door de openheid van de vraag en het tekstueel antwoorden variëren de antwoorden op de vragen. Het kan zo zijn dat iemand de ‘mier’ die hierboven staat herkent als een ‘insect’. Dit is een vorm van een generalisatie die wel correct is, een mier is namelijk een insect. In de resultaten zal onderscheid gemaakt worden in de mate van correctheid, waarbij gradaties van striktheid gehanteerd worden. Er zal een aparte analyse zijn van verschillende eisen aan de classificaties, één waarin generalisaties ook goed gerekend worden, één waarin alleen de exact juiste classificaties goedgekeurd worden, één waarin ook preciezere classificaties goed zijn en combinaties hiervan. Een voorbeeld van een preciezere classificatie is het ‘paard’ hierboven classificeren als een ‘ijslands paard’. Synoniemen en verkleinwoorden worden goed gerekend.

4. Resultaten

Om de onderzoeksvraag goed te kunnen beantwoorden, namelijk *'Hoe verhoudt de nauwkeurigheid van schetsclassificatie door convolutionele neurale netwerken zich met die door mensen gebruikmakend van Google's quickdraw dataset?'*, is het van belang dat het CNN zo nauwkeurig mogelijk is. De nauwkeurigheid is namelijk hetgeen dat gemeten wordt. Met een slecht afgesteld CNN zou de optimale nauwkeurigheid niet gemeten worden en daarmee zou er geen zinnige conclusie getrokken kunnen worden op basis van de vergelijking met de mens. In de methode hiervoor is beargumenteerd waarom bepaalde parameters voor het CNN worden gebruikt voor een optimale nauwkeurigheid. Deze parameters zijn goed onderzocht en conventioneel voor CNN's of kunnen van tevoren worden beredeneerd. Voor verschillende batch sizes en dropout rates is dit niet het geval. Daarom zijn er in dit onderzoek experimenten uitgevoerd om erachter te komen welke batch size en/of dropout rate het beste presteren, zodat de parameters van het CNN optimaal kunnen worden afgesteld. De resultaten van de experimenten met deze twee parameters zijn in sectie 4.1 en 4.2 hieronder beschreven.

Er is ook kort geëxperimenteerd met het toevoegen van meer verbonden lagen en de hoeveelheid nodes. Hieruit bleek dat na het toevoegen van meer verbonden lagen de accuracy afnam, een verklaring hiervoor is dat het netwerk gaat overfitten. Met extra lagen of meer nodes zijn er namelijk meer gewichten en dan is het netwerk in staat om complexere relaties te ontdekken op basis van ruis in de data. Eén volledig verbonden laag van 1024 nodes presteerde het beste. Daarnaast is ook ter controle even kort geëxperimenteerd met andere optimalisatiealgoritmen naast de Adam optimizer, het Adagrad optimalisatiealgoritme en stochastic gradient descent. Hieruit bleek al snel met overtuiging dat de Adam optimizer het beste presteerde zoals verwacht naar aanleiding van het onderzoek van David Mack (2015).

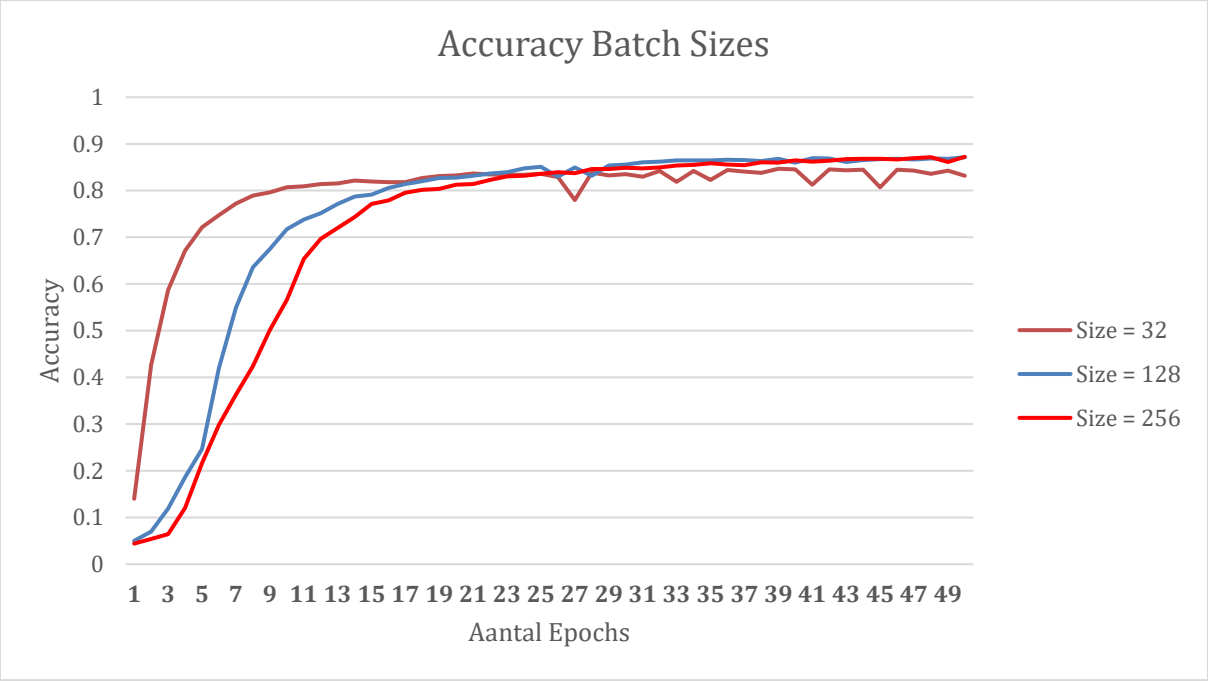
4.1 Batch size

De resultaten van het variëren met batch size zijn hieronder weergeven in tabel 2. De accuracy in tabel 2 is verkregen op basis van de testdata en hierbij was de dropout rate 0% op alle lagen. In figuur 11 is te zien hoe snel bepaalde batch sizes convergeren per epoch naar hun maximale accuracy. Een epoch betekent dat alle trainingsdata, gesepareerd in batches, éénmaal door het netwerk is gegaan.

Batch size	Test accuracy
32	0.847
128	0.880
256	0.877

Tabel 2: De test accuracy resultaten van variëren met batch sizes.

Uit deze resultaten blijkt dat een batch size van 32 sneller convergeert dan een batch size van 128 of 256 zoals te zien is in figuur 11. Daarentegen behaalt een batch size van 32 een aanzienlijk lagere accuracy dan een batch size van 128 of 256. Dus voor een op nauwkeurigheid optimaal presterend CNN zal deze batch size niet geschikt zijn. Tussen de batch sizes van 128 en 256 zit geen significant verschil in de accuracy. Uit figuur 11 blijkt dat een batch size van 128 en 256 ongeveer even snel convergeert. Tegelijkertijd duurt het bij een batch size van 128 wel langer per epoch, omdat er meer iteraties nodig zijn om alle trainingsdata door het netwerk te krijgen. Dus qua tijd is een batch size van 256 sneller dan één van 128. De batch size die in ons model wordt gebruikt is 256, op basis van deze resultaten.



Figuur 11: De trainingsfase van verschillende modellen met variërende batch sizes. Hierbij is per epoch te zien wat de accuracy was op de testdata.

4.2 Dropout

Tijdens het trainen zijn er verschillende modellen getest en daarop is gevarieerd met dropout op verschillende lagen. De resultaten hiervan zijn te zien in tabel 3. Elk van deze modellen had een batch size van 256.

Dropout rate	Test accuracy
0%	0.877
1% op fc laag	0.879
5% op fc laag	0.052
5% op 1ste conv laag	0.879

Tabel 3: De resultaten van variëren met dropout op verschillende lagen. Conv laag is de eerste convolutionele laag en de fc laag is de volledig verbonden laag (fully connected).

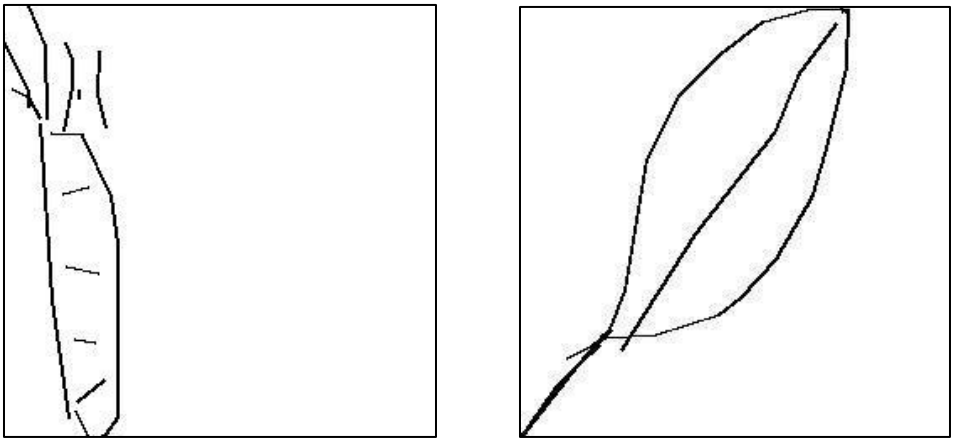
Het is gebruikelijk om dropout in een CNN toe te voegen op de volledig verbonden laag na de convolutionele lagen. Bij het CNN gebruikt in dit onderzoek bleek dat een kleine hoeveelheid dropout van 1% op deze laag geen significant verschil had op de accuracy. Zodra de dropout verhoogd werd naar slechts 5% had het model geen vermogen om te leren. Dit komt doordat er slechts één volledig verbonden laag in ons model zit. De nodes in die ene laag zijn hierdoor belangrijk en is het gevaarlijk veel weg te laten. Daarentegen had 5% dropout op de eerste convolutionele laag niet zo'n drastisch negatief effect op de accuracy, maar ook geen significante verbetering.

Dropout is belangrijk als methode tegen overfitting. Uit de experimenten blijkt dat onze CNN niet veel last heeft van overfitting en dat dropout niet nodig is. Dit is te verklaren aan de hand van de grootte

van het model, zoals verklaart in sectie 3.5 is ons CNN een relatief klein model. De relaties die een klein model ontdekt generaliseren beter (Abu-Mostafa et al., 2012) en dit blijkt ook uit onze resultaten. Op basis van deze bevindingen zal de dropout rate op 0% worden gezet.

4.3 Resultaat van de vergelijking

Als alleen de correcte classificaties goed worden gekeurd, behalen de participanten over alle schetsen een accuracy van 63.75%. Als ook generalisaties en te specifieke antwoorden worden goedgekeurd stijgt de accuracy van de participanten naar 68.97%. Het best presteren de participanten op het classificeren van een 'wortel' met een accuracy van 100%. Daarentegen presteren ze het slechtst op het classificeren van een 'veer' of een 'ui' met een accuracy van slechts 4.35%. De foute classificaties zijn intuïtief te verklaren doordat de schetsen op een ander object lijken en mensen daardoor niet lang stilstaan bij alle opties die het object kunnen zijn anders dan hun eerste (verkeerde) ingeving. De 'veer' bijvoorbeeld werd door bijna alle participanten herkend als 'blad' of 'blaadje'.



Figuur 12: Een wortel (links), de best geclassificeerde schets door mensen.
Een veer (rechts), de slechtst geclassificeerde schets door mensen.

In de tabel hieronder zijn een viertal interessante klassen en de resultaten per klasse weergegeven. Zoals in de tabel te zien is, is bij de klasse 'panda' een grote sprong in accuracy, zodra generalisaties ook goedgekeurd worden. Een pandabeer kan herkend worden als generalisatie 'beer' en dit wordt met deze striktheid goedgekeurd en maakt een significant verschil. Tevens is dit de enige klasse waar generalisaties (of te specifiek) goedkeuren zo'n groot verschil maakt.

Klasse	Accuracy C	Accuracy CG	Accuracy CS	Accuracy CGS	Accuracy CNN
Alle	63.75%	66.85%	65.82%	68.97%	88.00%
Mier	67.39%	71.74%	69.57%	73.91%	87.00%
Wortel	100%	100%	100%	100%	92.65%
Pizza	42.39%	50.00%	43.48%	51.09%	84.66%
Panda	56.52%	84.78%	56.52%	84.78%	88.77%
Null-klasse	-	-	-	-	47.36%

Tabel 4: Accuracy in het herkennen van de schetsen door participanten van de vier meest interessante klassen van de twintig met verschillende striktheid in goedkeuring afgezet tegen de accuracy van het CNN. CGS betekent bijvoorbeeld dat correcte, generalisaties en te specifieke antwoorden allemaal goed zijn gekeurd. C = Correct, G = Generalisatie, S = Te specifiek.

We kunnen zien dat de accuracy van het CNN op alle klassen 88% is. Deze is aanzienlijk hoger dan die van de participanten, die op de minst strenge striktheid een accuracy van 68.97% behalen. Enkel op de klasse ‘wortel’ hadden de participanten een hogere accuracy dan het CNN. Daarnaast is een accuracy van 88% ook hoger dan het onderzoek van Guo et al. (2018) die een accuracy van 62.1% op 345 klassen haalde. De verklaring hiervoor is dat in ons onderzoek minder klassen met meer data per klasse zijn gebruikt.

Het CNN behaald op de ‘null-klasse’ een accuracy van 47.36%. Deze klasse is toegevoegd zodat het CNN ook de optie heeft om klassen niet te herkennen. Deze accuracy van 47.36% betekent dat het CNN schetsen uit de ‘null-klasse’ eerder classificeert als één van de twintig (hoofd)klassen dan als de ‘null-klasse’. Hieruit blijkt dat het model moeite heeft met herkennen of een schets niet tot de twintig (hoofd)klassen behoort. De hypothese (onderaan in de alinea over de ‘null-klasse’ in sectie 2) dat het model een schets zal classificeren met de ‘null-klasse’ als hij niet tot één van de twintig (hoofd)klassen behoort is dus niet geheel correct. Alhoewel een accuracy van 47.36% wel aangeeft dat het CNN de schetsen die niet tot de twintig (hoofd)klassen behoren met enige zekerheid correct kan classificeren als de ‘null-klasse’, gezien er op 21 klassen wordt geclassificeerd.

5. Conclusie & Discussie

In dit onderzoek is een CNN opgesteld die schetsen kan classificeren. Daarna is de nauwkeurigheid van dit model vergeleken met de nauwkeurigheid van schetsherkenning bij participanten. Uit de resultaten blijkt dat het CNN significant beter presteert in het herkennen van de schetsen dan de participanten.

Je kan het classificatieprobleem van schetsen bij mensen zien als het hebben van veel representaties van objecten en dieren in onze hersenen. Zodra de mens een schets ziet, wordt dat geassocieerd met één of meerdere van die representaties. Op dezelfde manier heeft het CNN ook deze representaties, maar een stuk minder dan de mens, namelijk 20 klassen plus de 30 uit de 'null-klasse'. Nu representeert de 'null-klasse' in enige mate de onzekerheid (al het andere dat een schets kan zijn dan wat je denkt dat het is) bij herkenning door mensen, maar dit zal nooit geheel representatief zijn voor de grote hoeveelheid dingen die een schets kan zijn in het menselijk brein. Hier heeft het CNN dus een voordeel. Echter uit resultaten blijkt wel dat het CNN moeite heeft met het herkennen van schetsen die tot deze 'null-klasse' behoren en daarmee moeite heeft met herkennen wanneer een schets niet tot één van de twintig klassen behoort.

Daarnaast moet het model zich wel wapenen tegen ruis waar de participanten geen last van hadden en is het zo dat mensen van nature objecten, dieren of concepten kunnen herkennen aan de hand van meerdere representaties. Mensen kunnen een giraffe op verschillende manieren tekenen en nog steeds herkennen dat het een giraffe is. Het CNN heeft meer moeite met inzien dat het nog steeds om dezelfde klasse gaat als de schetsen hierin variëren.

Wat geconcludeerd kan worden uit dit onderzoek is dat de nauwkeurigheid van een convolutioneel neuraal netwerk, dat getraind is om een beperkt aantal klassen schetsen te herkennen, significant hoger is dan die van mensen. Er valt echter niet met zekerheid te zeggen of een convolutioneel neuraal netwerk altijd beter is in het herkennen van door mens getekende schetsen dan mensen zelf. Bij een toename van het aantal klassen waarop classificeert wordt kunnen de mensen wellicht beter presteren dan een CNN. Een mogelijkheid om beter te testen of een CNN over het algemeen beter presteert dan mensen is door een CNN te trainen op een dataset met schetsen zonder labels en het model zelf klassen te laten vinden. Aangezien schetsen veel verschillende representaties hebben lijkt dit een erg lastige taak. Maar dit is een interessant idee voor een vervolgonderzoek.

De data gebruikt in dit onderzoek zijn speciaal gemaakt voor classificatie van schetsen. Elk datapunt is een witte achtergrond met een schets. In realiteit zal bij het classificeren van objecten of dieren haast altijd ruis in de achtergrond te vinden zijn, geen witte achtergrond. Ook zal in de echte wereld verzamelde data meer datapunten bevatten van slechte kwaliteit of missende data dan in de gebruikte dataset.

Met de beschikbare computerkracht is de maximale grootte trainingsdata genomen en die is vrij groot gegeven de aard van de gebruikte data. Echter het model zal beter presteren bij gebruik van meer data, welke beschikbaar was. Waarschijnlijk zou bij gebruik van meer data en meer computerkracht het model ook complexer gemaakt kunnen worden met meer convolutionele lagen en volledig verbonden lagen waardoor hij beter zal presteren. In een vervolgonderzoek zou geëxperimenteerd kunnen worden met meer data en een complexer model voor nog betere resultaten in het classificeren.

6. Literatuurlijst

Li, X., Zhang, G., Huang, H., Wang, Z., & Zheng W. (2016). Performance Analysis of GPU-Based Convolutional Neural Networks. *2016 45th International Conference on Parallel Processing (ICPP).* Philadelphia, PA, USA, 16-19 Augustus. doi: 10.1109/ICPP.2016.15

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278 - 2324. doi: 10.1109/5.726791

Krizhevsky, A., Sutskever I., & Hinton G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems, 25* (2). doi: 10.1145/3065386.

Ciresan D., Meier U., & Schmidhuber J. (2012). Multi-column Deep Neural Networks for Image Classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition.* Providence, RI, USA, 16-21 Juni. doi: 10.1109/CVPR.2012.6248110

Guo K., WoMa J. & Xu E. (2018). Quick, Draw! Doodle Recognition.

McCulloch, W.S., & Pitts, W.H. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *The bulletin of mathematical biophysics, 5*(4), 115-133. doi: 10.1007/BF02478259

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. 318-362. MA, USA: MIT Cambridge press

Simard, P.Y., Steinkraus, D., & Platt, J.C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Edinburgh, UK, 6 Augustus. doi: 10.1109/ICDAR.2003.1227801

Thoma, M. (2017). Analysis and Optimization of Convolutional Neural Network Architectures.

Kingma, D.P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations*. San Diego, CA, USA, 7-9 May.

Mack, D. (2015, April 9). How to pick the best learning rate for your machine learning project. Retrieved from https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

Stenroos, O. (2017). Object detection from images using convolutional neural networks.

Lu, W., & Tran, E. (2017). Free-hand Sketch Recognition Classification.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15*(1), 1929-1958.

Googlecreativelab (2017). Quickdraw-data. GitHub repository. Retrieved from https://github.com/googlecreativelab/quickdraw-dataset

Owens, J.D., Houston, M., Luabke, D., Green, S., Stone, J.E., & Phillips, J.C. (2008). GPU Comping. *Proceedings of the IEEE, 96*(5), 879-899. doi: 10.1109/JPROC.2008.917757

Storrs, K.R., & Kriegeskorte, N. (2019). Deep Learning for Cognitieve Neuroscience. *The Cognitive Neurosciences, 6th edition.*

Bendersky, E. (2016). The Softmax function and its derivative. Retrieved from https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

Udofia, U. (2018). Basic Overview of Convolutional Neural Network (CNN). Retrieved from https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

Dertat, A. (2017). Applied Deep Learning – Part 4: Convolutional Neural Networks. Retrieved from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Nwankpa, C.E., Ijomah, W., Gachagan, A., & Marschall, S. (2018). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *Computer Vision and Pattern Recognition.*

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H. (2012). Learning from data. 126-134. AMLBook.

7. Appendix

De code van het model dat is gebruikt in dit onderzoek is te vinden en te downloaden in de volgende GitHub repository: https://github.com/ThomasZwart/ConvNet-Bachelor-Thesis

Het neurale netwerk is getraind in Python en maakt gebruik van de Tensorflow-library die is uitgebracht door Google in 2015. Deze library is speciaal ontwikkeld voor machine learning en uitstekend voor het maken en trainen van CNN's. Daarnaast wordt er gebruik gemaakt van de CUDA toolkit en cuDNN van Nvidia uitgebracht in 2007 (Li et al., 2016). Deze toolkit stelt ons in staat om het model op de GPU te trainen wat voor neurale netwerken ideaal is, aangezien alle gewichten parallel geüpdatet kunnen worden met dezelfde code (Owens et al., 2008).

Om deze code te gebruiken is Python versie 3.6 of hoger nodig. Daarbij moeten de Python modules Tensorflow en Numpy gedownload zijn. Dit kan gedaan worden door 'pip install numpy' en 'pip install tensorflow' in de console als commando's uit te voeren.

Het bestand 'fillData.py' dat te vinden is in de repository wordt gebruikt om de data voor te bereiden zodat het neurale netwerk hierop getraind en getest kan worden. De data van de Quickdraw dataset is beschikbaar per categorie zonder labels, deze code maakt van alle categorieën één bestand met gelabelde data.

Voor het gebruik van deze code is het nodig om een folder 'data' aan te maken in dezelfde map als 'fillData.py' en daarin voor elke categorie de data als .npy bestanden van de volgende URL: https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap?pli=1 te downloaden. Verdere informatie over het gebruik van de code is te vinden in de 'README' van de repository.

Met de volgende URL is een .npy bestand te downloaden dat door de auteur van tevoren met deze code is gemaakt. Het zijn 1000 samples van alle 21 klassen uit de testdata. https://mega.nz/#!tXZAwCIb!iJiXpThqrIe_egiJJuAlBjmdQj1M9_e4C_m4B3arZU0

Het bestand 'cnn.py' dat te vinden is in de repository bevat het geïmplementeerde convolutionele neurale netwerk zoals beschreven in dit onderzoek. Deze code is verantwoordelijk voor het ontwerpen, trainen en testen van het model. Als eerste wordt het bestand dat met 'fillData.py' is gemaakt opgehaald en verdeeld in trainings- en validatiedata. Hiermee wordt een neuraal netwerk getraind en opgeslagen.

De code zoals hij in de repository staat is niet afgesteld op trainen, maar op het testen van het .npy bestand die hierboven te downloaden is. Bij het testen wordt het model dat door de auteur van tevoren is getraind gebruikt, deze is tevens te vinden in de repository binnen de folder 'model'.
Om het model te testen, download 'cnn.py', de .npy file met de 'mega' link hierboven en de 'model' folder uit de repository. Zet deze drie in dezelfde folder en run 'cnn.py'. Na het inladen van de Tensorflow library wat enkele seconde duurt verschijnt de accuracy van het model, dit zou '0.8793' moeten zijn. Dit model en de testdata zijn niet hetzelfde als degene gebruikt voor de resultaten, vandaar de kleine afwijking in de accuracy.

Informatie over hoe men zelf een model kan trainen is te vinden in de 'README' van de repository.

Voor meer informatie of mocht één van de bovenstaand URL's niet meer werken, neem dan contact op met de auteur via thomas.black@live.nl.