One-shot learning van gebaren in een convolutioneel neuraal netwerk

Jasper Vaneessen

Promotor: prof. dr. ir. Joni Dambre
Begeleider: Lionel Pigou

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: informatica

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. Rik Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016-2017

UNIVERSITEIT
GENT

Toelating tot bruikleen

Jasper Vaneessen, juni 2017

Voorwoord

Jasper Vaneessen, juni 2017

Overzicht

One-shot learning van gebaren in een convolutioneel neuraal netwerk

Jasper Vaneessen

Promotor: Prof. Dr. Ir. Joni Dambre
Begeleider: Ir. Lionel Pigou

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen:
informatica

Vakgroep Elektronica en Informatiesystemen
Voorzitter: Prof. Dr. Ir. Rik Van de Walle

Faculteit Ingenieurswetenschappen & Architectuur
Academiejaar 2016-2017 Universiteit Gent

Trefwoorden

gebarenherkenning, convolutionele neurale netwerken, machine learning, deep learning, one-shot learning

One-shot learning of gestures using a convolutional neural network

Jasper Vaneessen

Supervisor(s): Lionel Pigou

Abstract—**The deaf and hard of hearing community has a problem communicating with hearing people. They can often understand verbal communication through means of lip reading but they can't express their thoughts directly to someone who does not know sign language. Recently a lot of progress has been made in the field of image processing and for example facial recognition. Convolutional neural networks have proven to be very efficient at various recognition tasks such as facial recognition or recognition of handwritten characters. In this study a convolutional neural network is built and trained to recognize gestures from a set of twenty different ones with a test error of 11.51%.**

Keywords— **One-shot learning, convolutional neural networks, gesture recognition, ChaLearn LAP 2014**

I. Introduction

Sign language is the main form of communication in the deaf and hard of hearing community. It is a highly visual-spatial, linguistically complete and natural language. Instead of using acoustically conveyed sound patterns sign language combines hand shapes, orientation and location as well as movement of arms and body. These signs can also be supported by facial expressions.

Some deaf people can read lips and thus understand some of the verbal communication, provided people articulate well and speak slowly. But when they try to convey a message, they have to rely on interpreters or text writing.

Interpreters can be very expensive and their availability is limited while writing down everything you want to say can be very inconvenient and feels limiting. It is very frustrating when your ability to express yourself is dependent of other peoples capabilities or their readiness.

There is also a communication issue among the non-hearing community. Sign language is not at all universal. Different countries have different sign languages and some even have regional ones. So even when someone knows a sign language, this does not mean they can communicate with all deaf people.

II. Methodology

A. Convolutional neural networks

Instead of accounting for all these various edge cases and spending a lot of time on building an optimal feature set we can use another technique. Humans and some animals can recognize objects from a very young age. It is not something we do explicitly, we just see the difference between for example a bicycle and a motorcycle. We have this ability because we learn from previous experiences. Every time humans see a new object they implicitly store features of it in their brain and use these to identify the same object in the future.

There has been a lot of research on the visual cortex of respectively a cat and a monkey. The cortex consists of simple and complex cells. The simple cells perform the feature extraction while the complex cells combine local features from a small spatial neighborhood. This is called spatial pooling and it is crucial to obtain a translation-invariant result.

Convolutional Neural Networks (CNN) try and mimic this principle. They consist of alternating convolutional and pooling layers. The convolutional layer functions as a feature extractor with several feature maps in each layer. Each feature map detects another feature and the amount of feature maps per layer increases the deeper we go into the network. Pooling layers combine these features to a smaller resolution, thus ensuring a translation invariant detection. Using this kind of architecture

Fig. 1. Sample of the ChaLearn, looking at people 2014 dataset

we can feed our network a dataset and let it learn to recognize objects in it. Here the ChaLearn, looking at people 2014 dataset is used. It consists of ten thousand samples of twenty gestures. These gestures are performed by different people in different environments to ensure a general enough result.

The network would learn from these examples as humans do. It looks at an example, tries to predict the gesture and checks its result with the actual gesture. If it is wrong, the network learns from its mistakes and adjusts itself. So all we have to do is let our network learn from a big set of examples and make this learning process optimal. In figure 2 the architecture used in this

Fig. 2. A schema of the used CNN architecture

study is visualized. The network consists of three convolutional and pooling layers and one fully connected or dense layer. This final layer performs the classification of the samples. It takes input from all the feature maps in the third convolutional layer and outputs it to the twenty output nodes of the output layer. Each node represents a gesture which can be recognized.

So at the start of the network, very global and generic features are detected. As we descend into the network all these features get combined and more specific aspects of the image are detected until finally the dense layers makes a decision and outputs the recognized gesture.

III. Conclusion

References

[1] Bart Lannoo, Didier Colle, Mario Pickavet, Piet Demeester, *Optical Switching Architecture to Implement Moveable Cells in a Multimedia Train Environment*, Proc. of ECOC 2004, 30th European Conf. on Optical Communication, vol. 3, pp. 344-345, Stockholm, Sweden, 5-9 Sep. 2004.

[2] Michael Neufeld, Ashish Jain, Dirk Grunwald, *Nsclick:: bridging network simulation and deployment*, http://systems.cs.colorado.edu/Networking/nsclick/

[3] *The Click Modular Router Project*, http://www.read.cs.ucla.edu/click/
[4] *NS – Network Simulator*, http://nsnam.isi.edu/nsnam/

Inhoudsopgave

Afkortingen

ANN	Artificieel Neuraal Netwerk
CNN	Convolutioneel Neuraal Netwerk
NGT	Nederlandse Gebarentaal
VGT	Vlaamse Gebarentaal
LSFB	la Langue des Signes de Belgique Francophone
CAB	Vlaams Communicatie Assistentie Bureau voor Doven
HMM	Hidden Markov Model
FNN	Fuzzy Neuraal Netwerk
DBM	Deep Boltzman Machine
MHI	Motion History Images

Hoofdstuk 1

Inleiding

1.1 Gebarentaal

Gebarentaal is in de eerste plaats een taal. Taal is een begrip dat moeilijk te definiëren valt en de meeste pogingen hiertoe beperken zich tot gesproken taal. Een definitie die ook voor gebarentaal kan gebruikt worden is:
"Een taal is een natuurlijk ontstaan menselijk communicatiemidddel waarmee je kan communiceren over alles wat je denkt, ziet, voelt en droomt. Een taal bestaat uit bouwstenen. Die bouwstenen worden volgens bepaalde regels samengevoegd tot grotere gehelen. Elke taal heeft eigen bouwstenen en regels" [Buyens, 2003].

Gesproken taal en gebarentaal verschillen in de manier waarop gecommuniceerd wordt: oraal-auditief tegenover gestueel-visueel. Door middel van hand-, hoofd- en armbewegigingen wordt een woord "uitgesproken" en vervolgens visueel waargenomen.

Een gebarentaal onstaat, net zoals een gesproken taal, spontaan en natuurlijk door contact tussen mensen. Net door deze spontane ontwikkeling is er geen universele gebarentaal. Evenals we verschillende gesproken talen en dialecten kennen per land of regio zijn er ook verschillende gebarentalen [Van Herreweghe en Vermeerbergen, 2009]. In Nederland is er bijvoorbeeld de Nederlandse Gebarentaal (NGT) en in België de Vlaamse Gebarentaal (VGT) en de Waalse Gebarentaal (la Langue des Signes de Belgique Francophone, LSFB). VGT verschilt dan weer van provincie tot provincie, met de grootste verschillen tussen West-Vlaanderen en Limburg, de twee verst uiteenliggende regio's.

Een gebarentaal heeft een eigen grammatica en lexicon. Het lexicon of de gebarenschat is de verzameling van alle woorden of gebaren in de taal. Het lokale gebarenschat moet volledig onafhankelijk van het lokale woordenschat worden beschouwd.
Bepaalde woorden uit de ene taal kunnen niet eenduidig vertaald worden in een andere taal. Het woord "gezelligheid"kent bijvoorbeeld geen Engelse vertaling en voor het Duitse "fingerspitzengefühl"hebben we in de Nederlandse taal ook geen alternatief.
Tussen een gebarentaal en een gesproken taal geldt dezelfde verhouding. Er is niet altijd een een-op-een relatie tussen een woord en een gebaar.

Communicatie tussen doven en horenden is vaak een struikelblok. Sommige doven kunnen liplezen en zo opmaken wat een spreker wil vertellen. Voorwaarde hierbij is dat de spreker goed moet articuleren en natuurlijk niet te snel spreekt.
Er kan ook altijd schriftelijk gecommunciceerd worden maar dit is een erg trage en onpersoonlijke vorm van communicatie. Ook is de bedrevenheid van een dove persoon in het schrijven van een gesproken taal vaak lager dan die van een horende.
Doven kunnen zich ook beroepen op een tolk. Dit kan een vriend zijn die horende is en gebarentaal kent of een beroepstolk. In Vlaanderen kunnen doven terecht bij het Vlaams Communicatie Assistentie Bureau voor Doven (CAB) om een tolk in te huren. [CAB-Vlaanderen, 2017] De Vlaamse overheid betaalt een aantal tolkuren terug. Onder andere achttien tolkuren voor privédoeleinden, achttien voor sollicitaties en een situatie-afhankelijk aantal tolkuren voor arbeid en beroepsopleiding.

1.2 Automatische gebarentaalherkenning

Er is dus een communicatieprobleem tussen doven en horenden omdat ze niet dezelfde taal spreken. Er zijn ook vele verschillende gebarentalen en dialecten waardoor er tussen doven onderling ook niet altijd vlot gecommuniceerd wordt. Door het gebruik van hedendaagse technologie moet het mogelijk zijn hierin te helpen en een automatisch herkenningssysteem uit te werken waarmee gebaren in real-time kunnen vertaald worden.

Het herkennen van objecten of gebaren is iets waar de mens niet bij stilstaat. Een pasgeboren kind begint vanaf het openen van de ogen zijn waarneming en herkenningsvermogen te trainen. Terwijl we leren organiseren we vormen, objecten en categoriën in nuttige taxonomiën en

linken deze dan later naar onze taal [Fei-Fei et al., 2006]. Eenmaal de leeftijd van zes jaar bereikt is kan een kind bijvoorbeeld 104 objectcategoriën onderscheiden zonder hierbij stil te staan.

Als mens kunnen we gebaren makkelijk differentiëren door registratie van armbewegingen, mimiek, houding van de handen en de manier waarop vingers gestrekt of geplooid worden. De neurologische fenomenen die deze vaardigheden kunnen verklaren worden nog steeds onderzocht.

Een machine of computer kan zien via het gebruik van een camera. Een beeld wordt voorgesteld door een matrix met pixelwaarden die de lichtintensiteit op dat bepaalde punt weergeeft. Traditioneel zijn er grijswaarden- en kleurbeelden maar tegenwoordig wordt ook vaak gebruikt gemaakt van 3D-cameratechnologiën, zoals de Microsoft Kinect [Kühn, 2011], zodat er een aanvullend dieptebeeld is. Deze beelden gelden dan als de visuele data voor het systeem, daarna moeten specifieke technologiën worden ingezet om nuttige informatie uit deze data te halen.

Een automatisch herkenningssysteem zal moeten leren omgaan met de grote variabiliteit van de invoer. De gebaren die het moet herkennen zullen uitgevoerd worden door mensen van verschillende grootte en lichaamsbouw. De vlotheid van het gebaren tussen ervaren en beginnende gebarentaligen zal sterk verschillen en de persoon zal niet altijd mooi recht in het midden van het beeld staan of even ver van de camera. Ook links- en rechtshandigheid heeft een invloed op het gebaren evenals de expressiviteit van de spreker.
De aanwezigheid van andere mensen of veel beweging in de achtergrond bemoeilijkt ook het herkennen van gebaren. Daarenboven moet ook nog rekening gehouden worden met de lokale belichting. De spreker kan onderbelicht of overbelicht zijn waardoor bepaalde contouren moeilijker te detecteren vallen.

Een compleet gebarentaalherkenningssysteem zal moeten voorzien in gebarensegmentatie, gebarenherkenning en grammaticale samenstelling van gebaren.

1.2.1 Gebarensegmentatie

Wanneer we een persoon die gebarentaal spreekt registreren met een camera krijgen we een continue stroom aan informatie. In een bepaalde tijdspanne kan een persoon een of meerdere gebaren uitvoeren en het is onbekend waneer een gebaar begint of eindigt. De segmentatie van deze gebaren is dus een eerste uitdaging voor een herkenningssysteem. Er is minder belangstelling naar deze "continue" gebaarherkenningssystemen omdat vaak wordt uitgegaan van vooraf gesegmenteerde beelden [Kuehne et al., 2011].

Tussen elk gebaar zit er een beweging die de overgang vormt tussen twee gebaren: de bewegingsepenthesis. Armen en handen gaan van eindpositie van het eerste gebaar naar beginpositie van het volgende. [Yang et al., 2010] Deze beweging moet gedetecteerd en gefilterd worden willen we een foutloze segmentatie krijgen.

1.2.2 Gebarenherkenning

Eenmaal we weten wanneer een gebaar begint en eindigt kunnen we het gaan identificeren. Uit de verzamelde visuele data wordt nuttige informatie geëxtraheerd waarmee het model kan beslissen over welk gebaar het gaat. Het beeld wordt omgezet in een beeldrepresentatie, bestaande uit een of meerdere featurevectoren. Deze representatie wordt vervolgens gebruikt door een classificatiemethode die het een klasselable geeft.

[Guo en Yang, 2017] stelt een gebaarherkenningssyteem voor die zich focust op de handen. Uit het dieptebeeld van een Kinectcamera wordt de hand gesegmenteerd via thresholding. Drie features worden vervolgens bijgehouden en gebruikt: de verandering van de handvorm, de beweging van de hand in het tweedimensionaal vlak en de beweging van de hand in de diepte (z-as).
Er wordt gebruik gemaakt van twee classificatie methodes: Hidden Markov Modellen (HMM) en Fuzzy Neurale Netwerken (FNN). HMM is een classificatietechniek die rekening houdt met het tijdsaspect. FNN is een combinatie van fuzzy (vage) theorie en artificiële neurale netwerken.

1.2.3 Grammaticale samenstelling

TODO

1.3 One-shot learning

1.3.1 Uitbreidbaarheid herkenningssysteem

Een taal is voortdurend in verandering. Het lexicon van een gebarentaal groeit mee met de tijd. Gloednieuwe termen of zaken die voordien geen beschrijving kenden in een gebarentaal worden toegevoegd. Hogescholen en universiteiten gebruikten lange tijd geen gebarentaal waardoor er weinig wetenschappelijke termen opgenomen zijn in de gebarenschat. Gelukkig komen er vandaag steeds meer wetenschappelijke gebaren bij.

Een automatische herkenningssysteem zal moeten leren omgaan met dit groeiende lexicon. Een strategie kan zijn om na verloop van tijd (vanaf een bepaald aantal nieuwe gebaren) het systeem te hertrainen met voorbeelden van de oude gebaren en de nieuwe gebaren. Hierbij wordt er dus vanaf nul gestart en een nieuw model opgebouwd.

Een eerste probleem is het verzamelen van de data. Deep learning methodieken hebben een complexe structuur en erg veel parameters. Om een grote hoeveelheid parameters te optimaliseren voor een taak heb je een grote hoeveelheid data nodig om uit te leren. Als we dus een nieuw gebaar willen bijleren aan een herkenningssysteem hebben we vele voorbeelden nodig van dit ene gebaar, liefst tegen verschillende achtergronden, uitgevoerd door verschillende personen en in verschillende lichtomstandigheden.
Het maken van dergelijke datasets is een erg kostelijke en tijdrovende opdracht.

Het model vanaf nul terug hertrainen vraagt veel tijd en rekenvermogen. Alle vooraf opgedane kennis wordt gewist dus alle tijd en moeite die eerder geïnvesteerd werd is voor niets. Het systeem zal ook minstens evenveel rekentijd nodig hebben als tijdens de opbouw van het vorige model.

Als we zo een aantal keer het herkenningssysteem willen uitbreiden zullen we veel kostbare tijd en energie verspillen.

1.3.2 Bijleren bij mensen

TODO

1.3.3 One-shot learning in de literatuur

[Lake et al., 2011] stelt een generatief model voor voor het herkennen van handgeschreven karakters. Het vertrekt vanuit de notie dat de mens een teken schrijft in verschillende halen of lijnen en ook zo een nieuw teken leert herkennen. Er wordt een dataset opgebouwd van 1600 karakters die door verschillende gebruikers online geregistreerd worden. Elke lijn die een gebruiker plaatst wordt opgeslaan alsook de volgorde van tekenen. Zo bestaat elk teken uit een opeenvolging van lijnen met verschillende vorm en lengte. De verzameling van al deze lijnen wordt gebruikt als voorafgaande kennis om nieuwe tekens bij te leren met een voorbeeld. Het nieuwe teken wordt door het systeem opgedeeld in lijncomponenten die dan afgetoetst worden tegen het model. Zo ontstaat een nieuwe representatie voor het bijgeleerde gebaar die kan gebruikt worden voor herkenning. Er wordt een nauwkeurigheid van 54.9 % behaald tegenover 39.6 % voor een implementatie aan de hand van Deep Boltzman Machines (DBM). Wanneer bij het aanleren van het nieuwe gebaar de lijninformatie van de dataset wordt gebruikt in plaats van die van het systeem zelf wordt een nauwkeurigheid van 63.7 % waargenomen.

[Wu et al., 2012] buigt zich over de ChaLearn One-shot Learning Gesture Challenge 2011 en leert vanuit slechts een voorbeeld een gebaar te herkennen zonder enige voorgaande kennis. Er wordt geëxperimenteerd met een aantal feature descriptors en classificatie methodes waaruit Extended Motion History Images (Extended MHI) en Maximum Correlation Coëfficiënt (MCC) als best presterende worden gevonden. Extended MHI bestaat zelf uit drie representaties: MHI en Inversed recording (INV) focussen zich op bewegingsinformatie respectievelijk in het begin en op het einde van het gebaar terwijl Gait Energy Information (GEI) repetitieve beweging registreert. Het systeem behaalt een Levensteihnafstand van 0.29685 (tussen 0 en 1 waarbij 0 optimaal) op de validatieset en presteert zeer goed op gebaren waarin er veel beweging is. De twee meer statische gebaren uit de dataset worden het minst goed gededecteerd met een nauwkeurigheid lager dan 45 %.

[Caelles et al., 2016] stelt een convolutioneel neuraal netwerk (CNN) voor die uit een voorbeeld de voorgrond van de achtergrond onderscheidt in een video. Het CNN wordt vooraf getraind op de ImageNet dataset. Een dataset van 1,2 miljoen afbeeldingen uit meer dan duizend categoriën. Door deze pre-training op een zeer ruime dataset is het model algemeen en leert het eigenlijk wat 'een object' is. Hierna wordt het model verfijnd voor het volgen van een voorgrondsobject uit een video. Het eerste frame van de video wordt gemaskeerd en hierop stelt het model zich af. Deze architectuur verbetert de state-of-the-art op de Densely Annotated Video Segmentation (DAVIS) dataset met 11.2 % (79.8% vs 68.0%).

1.4 Doelstelling

Hoofdstuk 2

Technische aspecten

2.1 Machine Learning

2.1.1 Inleiding

Leren is een veelzijdig fenomeen dat bestaat uit verschillende processen: het verkrijgen van declaratieve kennis, het ontwikkelen van motorische en cognitieve vaardigheden door instructie en ervaring, het organiseren van nieuwe kennis in algemene representaties en het ontdekken van nieuwe feiten via observatie en experimentatie.

Sinds het begin van het computertijdperk proberen onderzoekers het menselijk leren na te bootsen en deze processen te vertalen naar de informatietheorie. Het machinaal leren is nog steeds een erg uitdagend doel in de kunstmatige intelligentie (KI).

Deze vorm van KI is dus volledig data gedreven tegenover traditionele methoden die zich beroepen op handgemaakte regels. Het computerprogramma wordt niet expliciet geprogrammeerd om een taak uit te voeren maar vertrekt vanuit een algemeen model. Het model leert eerst uit voorbeelden en kan daarna voorspellingen maken op nieuwe invoer.

We kunnen zeggen dat een computerprogramma of machine leert als het zijn performantie op een bepaalde taak verbetert met ervaring [Carbonell et al., 1983]. Het leren gebeurt door de optimalisatie van de parameters van het predictief model door middel van een algoritme uit de machine learning. Het model wordt een aantal voorbeelden gegeven om uit te leren: de trainset. De uitvoer van het model wordt geëvalueerd aan de hand van een prestatiemaat.

Deze prestatiemaat vertelt hoe correct de voorspelling is en bepaalt de mate waarin het model verder gecorrigeerd dient te worden.

Machine learning algoritmes kunnen opgedeeld worden in drie categoriën op basis van leerstijl: gesuperviseerd, ongesuperviseerd en semi-gesuperviseerd leren. De supervisie slaat op het gebruik van de correcte uitgangswaarde tijdens het trainen.

Een ongesuperviseerde leerstijl vertrekt uit een dataset zonder klasselabels of correcte voorspellingswaarde. Er wordt een model opgebouwd die bepaalde structuren deduceert. Dit kan zijn om algemene regels te extraheren, om redundantie te verminderen of om gegevens te groeperen volgens gelijkenis (clustering).

Bij gesuperviseerde methodes wordt een trainset gebruikt die zowel de invoervector als de te voorspellen waarde bevat. Problemen die vaak via gesuperviseerd leren worden aangepakt zijn classificatie en regressie. Classificatie tracht ingevoerde voorbeelden in te delen in discrete categoriën om bijvoorbeeld objecten te herkennen zoals in [Krizhevsky et al., 2012a]. Bij regressie is de uitvoer van het model een continue variabele zoals bijvoorbeeld de prijs van een appartement gegeven zijn oppervlakte.

Semi-gesuperviseerde methodes zijn een mengvorm van de voorgaande twee. De dataset is een mengeling van gelabelde en ongelabelde voorbeelden. Hierbij is er een gewenste indeling, weergegeven door de gelabelde data, maar het model moet zelf de indeling zien te maken.

2.1.2 Gesuperviseerde classificatie

De machine learning in dit onderzoek valt onder de gesuperviseerde classificatie. Een classificatieprobleem kan algemeen als volgt worden omschreven:

Gegeven een trainset T:

$$T = \{(x^{(n)}, y^{(n)})\}, \quad x^{(n)} \in \mathbb{R}^D, y^{(n)} \in \{0, 1, ..., C\}, n = 1, ..., N \tag{2.1}$$

met $x^{(n)}$ het n-de datavoorbeeld en $y^{(n)}$ zijn klasselabel. C staat voor het aantal discrete categoriën of klasses, D het aantal dimensies van de invoervariabelen en N het aantal trainingsvoorbeelden. Nu kunnen we het classificatieprobleem uitdrukken als de benadering van

een model f met parameters θ:

$$f(x, \theta) = y, \quad \forall (x, y) \in T \tag{2.2}$$

zodat we na deze schatting vanuit f en θ voorspellingen kunnen maken op basis van nieuwe data: $f(x_{nieuw}, \theta) = y', \quad y' \in \{0, 1, ...C\}$

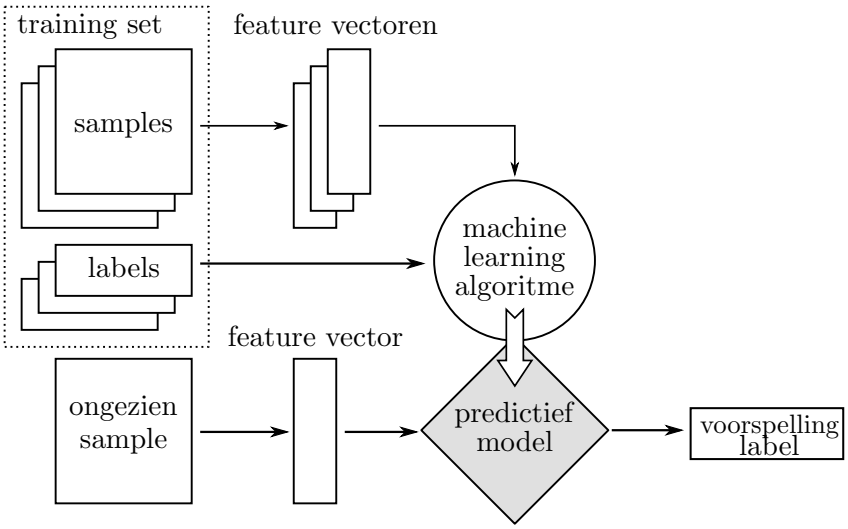
Figuur 2.1 geeft een schematische weergave van de opbouw van een classificatiemodel. Het schatten van f en θ wordt uitgevoerd door een techniek uit de machine learning en geeft ons het uiteindelijke predictief model $f(x, \theta) = y$. De samples worden meestal verwerkt voor ze als input worden doorgegeven aan het model. In het geval van classificatie van video en afbeeldingen worden er technieken uit de beeldverwerking gebruikt om relevante informatie uit het beeld te halen. Welke specifieke techniek er gebruikt wordt is een keuze die erg bewust moet gemaakt worden en een grote invloed heeft op de performantie van het classificatiemodel. Deze informatie wordt gebundeld in een featurevector (x) en aan het ML algoritme gegeven. Vanuit de verkregen featurevectoren en de kennis van het correcte klasselabel wordt het predictief model dan geoptimaliseerd.

Eenmaal het model voldoende geleerd heeft kan het predictief model gebruikt worden in een productieomgeving om nieuwe voorbeelden, al dan niet correct, te classificeren.

Het model dat in dit onderzoek gebruikt wordt is dat van het artificieel neuraal netwerk, een veel gebruikt predictief model voor classificatie. In sectie 2.2 wordt deze techniek dan ook verder besproken.

2.1.3 Overfitting

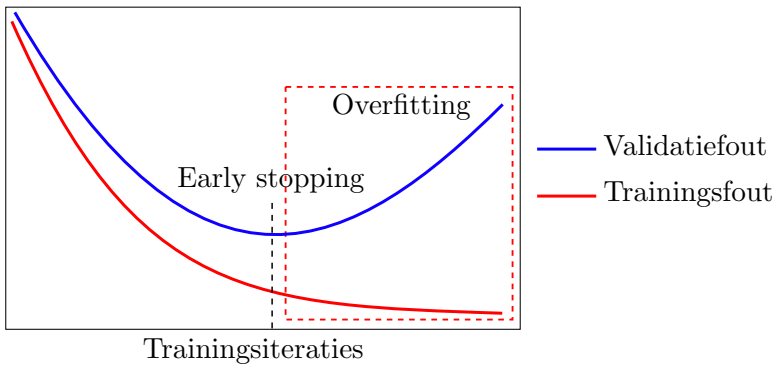
Stel dat een student kunstwetenschappen voor zijn examen het werk van verschillende kunstenaars moet herkennen. Hij krijgt een boek met daarin foto's van verschillende schilderijen samen met de naam van de uitvoerder. Indien de student al deze voorbeelden met hun antwoord vanbuiten leert en op het examen net dezelfde schilderijen als in het boek zou voorgeschoteld krijgen dan zal hij erg hoog scoren. Krijgt hij echter nog nooit eerder geziene werken van dezelfde schilders, dan zal zijn score veel lager liggen. Hij heeft geen onderliggende eigenschappen uit de beelden gehaald die terugkomen in ander werk maar memoriseerde gewoon werk en uitvoerder. Dit is een informeel voorbeeld van overfitting.



Figuur 2.1: Schematische weergave van het opstellen van een predictief model met behulp van machine learning technieken

Bij overfitting beschrijft het predictief model de ruis en fouten in het signaal in plaats van de gewenste onderliggende patronen en kenmerken van de gegeven taak. Het komt voor wanneer het model complex is en vele parameters heeft in vergelijking met het aantal trainingsvoorbeelden. Een model dat "overfit" is heeft een laag voorspellend vermogen op ongeziene data. Het model is niet algemeen of gegeneraliseerd en memoriseert de data.

Het risico op overfitten komt uit het feit dat de prestatiemaat voor de training van het model niet correct de effectiviteit van het model weergeeft. De training van een model gebeurt doorgaans via optimalisatie van zijn prestatie op een training set. De werkelijke performantie



Figuur 2.2: Plot van trainings- en validatiefout ter illustratie van overfitting en de early stopping techniek.

wordt echter gegeven door prestatie op ongeziene data. Net om deze reden wordt de dataset opgesplitst in drie delen: training-, validatie- en testset.

Om een zo goed mogelijk predictief model te maken wordt een evaluatiescore op de validatieset geminimaliseerd of gemaximaliseerd. Deze data is nog onbekend voor het model en geeft dus een betere aanduiding van zijn performantie. Figuur 2.2 geeft de validatie- en trainingsfout weer tijdens het trainen van een predictief model. Wanneer de trainingsfout blijft dalen maar de validatiefout stagneert of stijgt is er sprake van overfitting. Nu kan er gekozen worden om de training vroegtijdig te stoppen teneinde geen generalisatie te verliezen. Deze techniek wordt *early stopping* genoemd.

Om na afloop een finale evaluatie van het model te maken wordt zijn prestatie op de testset gemeten. Het model heeft de voorbeelden uit deze set nog nooit eerder gezien waardoor de testscore representatief is voor het voorspellingsvermogen van het model. De testset mag nooit gebruikt worden om aanpassingen aan het model te maken, hiervoor dient de validatieset.

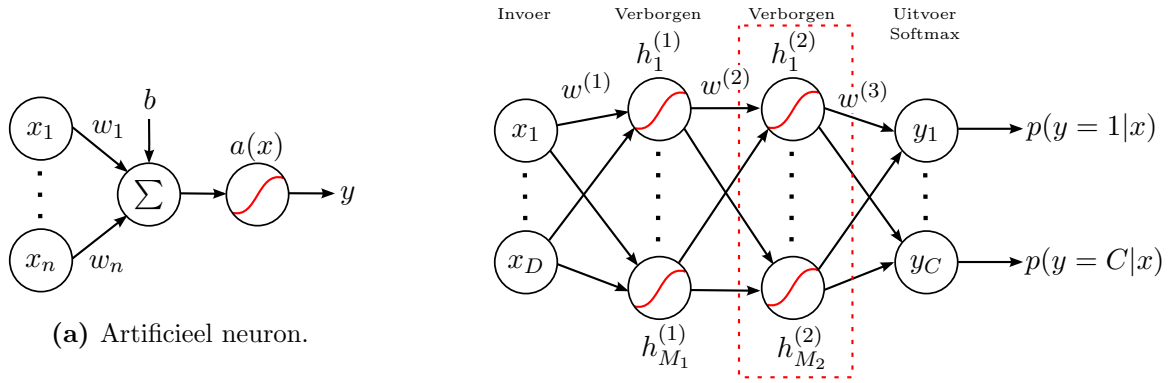
2.2 Artificieel neuraal netwerk

2.2.1 Inleiding

Een lange rekensom uitwerken is iets wat de mens liever overlaat aan een rekenmachine of computer. Alle bits van een rekensom zijn even belangrijk en bepalend voor zijn uitkomst. Op dit vlak is een computer veel efficiënter en betrouwbaarder dan ons brein. Onze hersenen zijn veel bedrevener in andere taken zoals het indelen van objecten volgens gelijkenis of het herkennen van een persoon.

Het is deze gedachte die vele onderzoekers ertoe bracht een artificieel neuraal netwerk (ANN) uit te bouwen. Het menselijke brein is een erg complex neuraal netwerk bestaande uit meer dan 86 miljard neuronen. Dit zijn de bouwstenen van ons zenuwstelsel en ze staan in voor het verwerken en overdragen van informatie. In figuur 2.6a is een weergave te zien van een artificieel neuron, de bouwsteen van het ANN. De functionaliteit van dit neuron bootst deze van het biologisch neuron na en wordt als volgt gedefiniëerd:

$$n = a\left(b + \sum_{i=1}^{D} w_i x_i\right) \tag{2.3}$$



(a) Artificieel neuron.

(b) Artificieel neuraal netwerk

Figuur 2.3: Een ANN met een invoerlaag, twee verborgen lagen en een softmax uitvoerlaag samen met zijn bouwsteen, het artificieel neuron.

met n de uitvoer van het neuron, a de *activatiefunctie*, b de bias, x_i de i-de invoer en w_i de gewichten van zijn respectievelijke inkomende verbindingen.

Deze artificiële neuronen worden gegroepeerd in lagen om een ANN te vormen zoals afgebeeld in figuur 2.6b. Alle lagen worden onderling volledig verbonden, elk neuron krijgt invoer van alle neuronen uit de voorgaande laag en geeft zijn uitvoer door aan alle neuronen van de volgende laag. Deze lagen worden dan ook vaak *dense layers* genoemd vanwege hun verbindingsdichtheid.

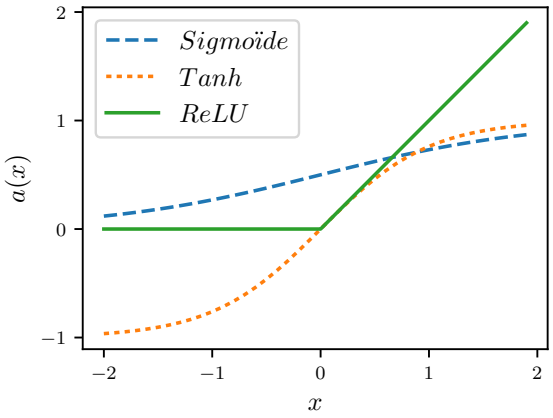
Hier beschouwen we enkel *feedforward neurale netwerken*. Aan het begin van het netwerk hebben we de invoerlaag met evenveel knopen als de dimensie van de featurevectoren. De invoer propageert zich van links naar rechts door het netwerk tot het bij de uitvoerlaag komt. Omdat de uitvoer van de binnenste lagen niet meteen zichtbaar is worden deze de verborgen lagen genoemd.

De uitvoer van de verschillende lagen uit het ANN in figuur 2.6b kunnen we als volgt beschrijven:

$$\begin{aligned} h_j^{(1)} &= a\left(b^{(1)} + \sum_{i=1}^{D} w_{i,j}^{(1)} x_i\right), \quad j = 1, ..., M_1 \\ h_j^{(2)} &= a\left(b^{(2)} + \sum_{i=1}^{M_1} w_{i,j}^{(2)} h_i^{(1)}\right), \quad j = 1, ..., M_2 \\ y_j &= a\left(b^{(3)} + \sum_{i=1}^{M_2} w_{i,j}^{(3)} h_i^{(2)}\right), \quad j = 1, ..., C \end{aligned} \tag{2.4}$$

met D het aantal invoerparameters, M_1 het aantal verborgen knopen in de eerste laag, M_2 het aantal verborgen knopen in de tweede laag en C het aantal te voorspellen klassen. M_1 en M_2 zijn hyperparameters van het model die geoptimaliseerd worden door de evaluatie van het model tegenover de validatieset.

Het is mogelijk en wenselijk om nog meer verborgen lagen toe te voegen. Het omkaderde gedeelte in figuur 2.6b kan een aantal keer herhaald worden om zo het model toe te laten een complexere structuur te modelleren.



Figuur 2.4: Weergave van de drie meest populaire activatiefuncties voor classificatie doeleinden

De activatiefunctie $a(x)$ uit (2.3) is van groot belang in een ANN aangezien ze zorgt voor een niet-lineariteit. Moest deze er niet zijn kan elk neuraal netwerk met een verborgen laag vervangen worden door een model zonder aangezien de samenstelling van lineaire transformaties zelf een lineaire transformatie is. Figuur 2.4 geeft drie van de meeste gebruikte niet lineaire activatiefuncties weer. In dit onderzoek zal gebruik gemaakt worden van de rectifier activatiefunctie:

$$a(x) = max(0, x) \tag{2.5}$$

Een neuron wordt vaak vernoemd naar de activatiefunctie die ze gebruikt. In dit geval spreken we van een Rectified Linear Unit (ReLU). Het gebruik van ReLU's wordt gestaafd door vele onderzoekswerken omtrent gesuperviseerde *deep learning* zoals [Glorot et al., 2011], [Pigou, 2014] en [Wu et al., 2016]. Een ReLU zal alle negatieve waarden afbeelden op 0 en alle positieve op zichzelf.

Aan de uitgang van het ANN worden geen ReLU's geplaatst maar hebben we de softmaxlaag. De uitvoer van de softmaxlaag geeft de voorspelde probabiliteit per klasse weer. Deze softmax wordt als volgt berekend:

$$\begin{aligned} p(y=j|x) &= \frac{exp(\alpha(x)_j)}{\sum_{i=1}^{C} exp(\alpha(x)_i)} \\ softmax(x) &= \{p(y=1|x), p(y=2|x), ..., p(y=C|x)\} \end{aligned} \tag{2.6}$$

α is de pre-activatie (waarde voor activatie) van de laatste verborgen laag, C het aantal klassen en $p(y = j|x)$ de probabiliteit dat een gegeven invoer x tot de klasse met label j behoort.

De uitvoer van de softmaxlaag is een vector wiens som gelijk is aan 1. Om een voorspelling uit deze softmaxlaag te kiezen wordt het maximum genomen uit deze uitvoervector, de klasse met de hoogste probabiliteit.

2.2.2 Training van een neuraal netwerk

Alle parameters θ van het netwerk (gewichten w en biases b) worden pseudo-willekeurig gekozen. Tijdens het trainen worden deze dan iteratief aangepast om de prestatie van het predictief model te verbeteren. Dit gebeurt aan de hand van een *back-propagation* algoritme. De data of het signaal van de invoer gaat eerst voorwaarts door het netwerk waarna we een voorspelling krijgen. Vervolgens wordt deze voorspelling geëvalueerd en zal de fout op de voorspelling vanuit het einde van het netwerk terug naar het begin propageren. Op basis van deze fout worden de gewichten aangepast en zo leert het model uit ervaring.

Om de prestatie van het model te maximaliseren wordt een kostfunctie geminimaliseerd. Deze geeft een indicatie van de gemaakte voorspellingsfout door aan slechte voorspellingen een hogere kost toe te kennen. Welke kostenfunctie gebruikt wordt hangt af van de uit te voeren taak. Voor multinomiale classificatie met behulp van een softmax is *categorical cross-entropy* het meest geschikt:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \log(p_{i,j}) \tag{2.7}$$

met N het aantal voorbeelden en C het aantal klassen. De probabiliteit uit de softmax wordt hier afgekort tot p. Het ware label y is weergegeven in een one-hot codering: een vector

met de grootte van het aantal klassen waarvoor alle waarden 0 zijn behalve op de index van het correcte klasselabel, daar is ze gelijk aan 1. Bijgevolg verdwijnt de logterm bij een foute voorspelling en wordt in de kostenfunctie enkel rekening gehouden met de fout op het ware label. Door het gebruik van een negatieve log krijgen waarden dicht bij 0 een erg hoge fout en is de fout bij $p_{i,j} = 1$ gelijk aan 0.

2.2.3 Gradient descent

Nu moeten de parameters θ gevonden worden die de kostenfunctie minimaliseren. Het minimum van deze kostenfunctie komt immers overeen met het optimale predictief model. Om dit minimum te vinden wordt gradient descent toegepast. In Figuur 2.5 wordt deze techniek weergegeven in een tweedimensionale parameterruimte. Het algoritme zal de parameters iteratief bijstellen door het volgen van de stijlste helling, gegeven door de gradiënt van de kostfunctie:

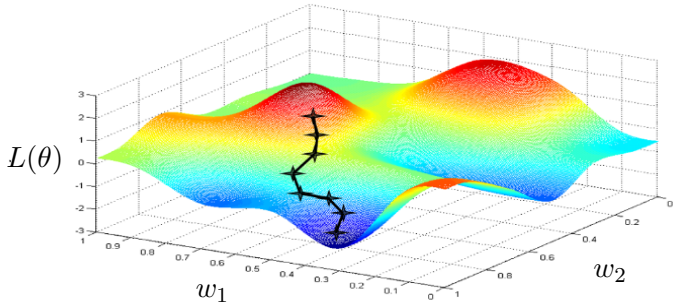
$$\theta = \theta - \alpha \nabla_\theta L(\theta) \tag{2.8}$$

met α de leersnelheid of *learning rate*. Deze learning rate is een hyperparameter van het model en bepaalt hoeveel de parameters bijgesteld worden per iteratie. Een te kleine leersnelheid zal zorgen voor een trage convergentie en bijgevolg lange trainingsfase. Een te grote leersnelheid loopt het risico voorbij lokale minima te lopen en niet te convergeren.

Het gradient descent algoritme zoals weergegeven in (2.8) moet de foutfunctie voor alle voorbeelden berekenen om de parameters correct aan te passen. Het trainen van neurale netwerken vraagt om grote datasets en daarom kan deze fout- en gradientsberekening erg traag uitvallen. Om dit probleem aan te pakken wordt de *mini-batch gradient descent* variant geïmplementeerd:

$$\theta = \theta - \alpha \nabla_\theta L(\theta; x_{i:i+B}; y_{i:i+B}) \tag{2.9}$$

met $L(\theta; x_{i:i+B}; y_{i:i+B})$ de kostfunctie van een klein gedeelte of *mini-batch* van de voorbeelden en B de batch-grootte. Zo bekomt men een schatting van de globale kostfunctie en kunnen er sneller stappen worden gezet. De keuze van de batch-grootte B is net als de learning rate α een hyperparameter van het netwerk.



Figuur 2.5: Visualisatie van gradient descent algoritme in een tweedimensionale parameterruimte. De zwarte lijn geeft de verschillende iteratieve stappen van het algoritme weer. Door het volgen van de steilste helling wordt een lokaal minimum van de kostfunctie gevonden.

2.2.4 Momentum-optimalisatie voor gradient descent

Als we de parameterruimte die het gradient descent algoritme afzoekt zouden vergelijken met een helling komt het in de problemen als er veel kleine putten zijn tijdens de afdaling. Het algoritme zal oscilleren tussen deze lichte hellingen, optimalisatiepad zal zigzaggen en de convergentiesnelheid van het algoritme zal sterk dalen. Ook is het meer waarschijnlijk dat het algoritme zal blijven steken in een van deze putten en niet verder afdaalt naar het minimum.

[Botev et al., 2016] beschrijft het gebruik van momentum als optimalisatie voor het gradient descent algoritme. De parameters worden geüpdate in dezelfde richting als de vorige update. Dit kan als volgt beschreven worden:

$$\begin{aligned} v_{t+1} &= \mu v_t - \alpha \nabla_{\theta_t} L(\theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{2.10}$$

met t de iteratiestap en $\mu \in [0, 1]$ de momentumcoëfficiënt. Bewegingen in dezelfde richting zullen snelheid accumuleren waardoor het optimalisatiepad over de kleine putten rolt. Er is echter nog een verbetering aan te brengen aan deze optimalisatie genaamd Nesterov's accelerated gradient (NAG):

$$\begin{aligned} v_{t+1} &= \mu v_t - \alpha \nabla_{\theta_t} L(\theta_t + \mu_t v_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{2.11}$$

NAG kijkt eigenlijk een stap vooruit door een afschatting van de volgende beweging te maken voor de gradientsberekening. Hiervoor wordt de huidige momentumwaarde gebruikt: $\theta_t + \mu_t v_t$.

In praktijk zien we dat NAG doorgaans dichter tot het lokale minimum komt dan klassiek momentum. Het verschil is doorgaans heel klein maar door het iteratieve proces cumuleren deze kleine verschillen tot snellere convergentie bij gebruik van NAG.

(a) ANN **(b)** ANN na toepassing dropout

Figuur 2.6: Links is een standaard ANN te zien met twee verborgen lagen, rechts is hetzelfde netwerk te zien na toepassing van dropout. De gekruiste units zijn de gedropte units.

2.2.5 Dropout

Tijdens het trainen kan het gebeuren dat neuronen volledig afhankelijk worden van de output van een enkel neuron uit de vorige laag (of een kleine groep neuronen). Deze units worden eigenlijk lui en dragen weinig tot niets bij aan het herkenningsvermogen. Dit fenomeen heet co-adaptatie. Om de units te verplichten zelf bij te dragen tot de herkenning van bepaalde patronen wordt dropout toegepast.

Tijdens elke iteratiestap van het trainingsalgoritme worden een aantal units uitgeschakeld (vandaar de benaming dropout). Of een unit al dan niet wegvalt wordt bepaald door een probabiliteit *Bernoulli(p)*. Deze p is een hyperparameter van het netwerk en wordt doorgaans ingesteld op 0.5 waardoor getraind wordt met de helft van de units. Elke iteratiestap wordt er dus getraind met een andere deelverzameling neuronen, in essentie een andere netwerk. Het uitschakelen van units gebeurt enkel tijdens training, daarom worden de gewichten herschaald tijdens het trainen met een factor $1/(1-p)$. De niet geschaalde gewichten zijn bruikbaar in de testfase wanneer alle neuronen worden benut.

Dropout is een regularisatiemethode die ondanks zijn simpliciteit erg efficiënt blijkt. [Srivastava et al., 2014] geeft een uitvoerige analyse en vergelijkende studie van het gebruik van dropout bij het uitvoeren van taken zoals gensplitsing, spraak-, tekst- en beeldherkenning. In elke van deze

experimenen zorgt dropout voor een hogere regularisatie. Zo wordt aangetoond dat dropout overfitting kan tegengaan in om het even welk toepassingsgebied.

2.3 Convolutioneel neuraal netwerk

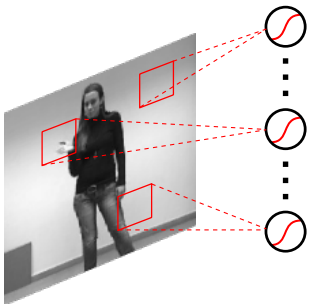
De machine learning techniek die sinds 2012 in de belangstelling staat is die van het convolutioneel neuraal netwerk. Mede dankzij het recordbrekende resultaat van [Krizhevsky et al., 2012b] op de "ImageNet Large Scale Visual Recognition Challenge" wordt deze techniek vaak onderzocht en toegepast op beeldherkenningsproblemen zoals in [Ji et al., 2013] en [Karpathy et al., 2014].

Traditionele beeldherkenningstechnieken zoals in [Perronnin et al., 2010] en [Jhuang et al., 2007] leiden het beeld af naar een manueel ontworpen beeldrepresentatie. Deze kenmerken zijn niet meteen zichtbaar maar worden via verschillende bewerkingen blootgelegd. Het is erg moeilijk om te beslissen welk soort kenmerk het best geschikt is voor de gegeven taak. Vaak moet er dan ook veelvuldig geëxperimenteerd worden met verschillende representaties.

Convolutionele neurale netwerken hebben als grote troef dat ze deze feature-extractie automatiseren. CNN's zijn biologisch geïnspireerde model. Het onderzoek van [Hubel en Wiesel, 1968] gaf een verklaring voor de manier waarop zoogdieren, specifiek de kat en de aap, de wereld rondom zich visueel waarnemen. De neuronen in de visuele cortex hebben een complexe laagsgewijze architectuur. Elke cel reageert op prikkels in een deelgebied van het gezichtsveld. Deze receptieve velden overlappen gedeeltelijk en bedekken zo het volledige zicht.

Het herkenningsvermogen van de mens is nog altijd ontegensprekelijk veel hoger dan om het even welk artificieel herkenningssysteem. Vandaar dat er geprobeerd wordt de functionaliteit van de visuele cortex na te bootsen met een CNN.

CNN's zijn een speciaal soort meerlagig neuraal netwerk. Net zoals een ANN worden ze getraind met behulp van een back-propagation algoritme zoals beschreven in Sectie **??**. Waar ze verschillen is hun architectuur. Een ANN is volledig verboden met zijn invoer. Als we een afbeelding van 64x64 pixels rechtstreeks als invoer in een ANN willen gebruiken moeten er per knoop 4096 gewichten worden getraind. De artificiële neuronen in een CNN werken net zoals de receptieve velden in de visuele cortex.



Figuur 2.7: Lokale connectiviteit van receptieve velden. Voor elk overlappend deelgebied van de invoer is er een neuron verbonden. Alle gewichten van het lokale filter worden gedeeld.

Zoals te zien in Figuur 2.7 wordt elk mogelijk (overlappend) deelgebied verbonden met een neuron. De parameters van deze verbindingen worden gedeeld tussen alle units. Zo leert het netwerk een bepaalde feature detecteren om het even waar in het beeld en zijn er slechts een beperkt aantal te trainen gewichten.

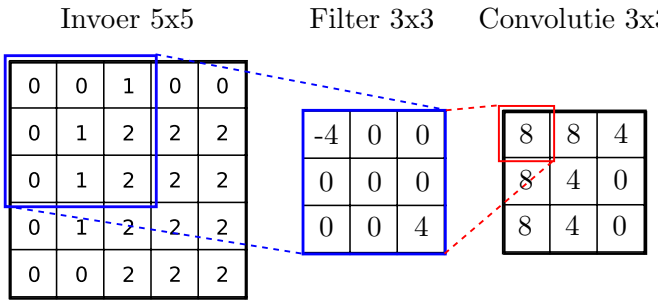
In een CNN worden convolutielagen afgewisseld met pooling lagen. Zo worden er meerdere lagen na elkaar geplaatst om een diepe architectuur te verkrijgen. In de eerste lagen zullen kenmerken van laag niveau gedetecteerd worden en naarmate we dieper in het netwerk gaan worden kenmerken van hoger niveau geëxtraheerd. Zo zullen de eerste lagen features zoals randen detecteren die dan dieper in het netwerk worden samengesteld tot bijvoorbeeld een bepaalde handvorm.

2.3.1 Tweedimensionale convolutie

De receptieve velden delen hun parameters en zoeken dus naar hetzelfde kenmerk in een beeld. Ze filteren het beeld met behulp van een tweedimensionale discrete convolutie die als volgt kan worden beschreven:

$$(w * x)_{h,b} = \sum_{b'=-\infty}^{+\infty} \sum_{h'=-\infty}^{+\infty} f_{b',h'} x_{b-b',h-h'} \tag{2.12}$$

met w de filterwaarden en x de invoer. De grootte van het filter en van de invoer is uiteraard begrensd. Met onze filtergrootte (H_f, B_f) en de dimensie van onze invoer (H_x, B_x) kunnen we (2.12) vereenvoudigen tot:



Figuur 2.8: Tweedimensionale convolutie met een 3x3 filter.

$$\sum_{b'=0}^{B_f} \sum_{h'=0}^{H_f} f_{b',h'} x_{b-b',h-h'} \tag{2.13}$$

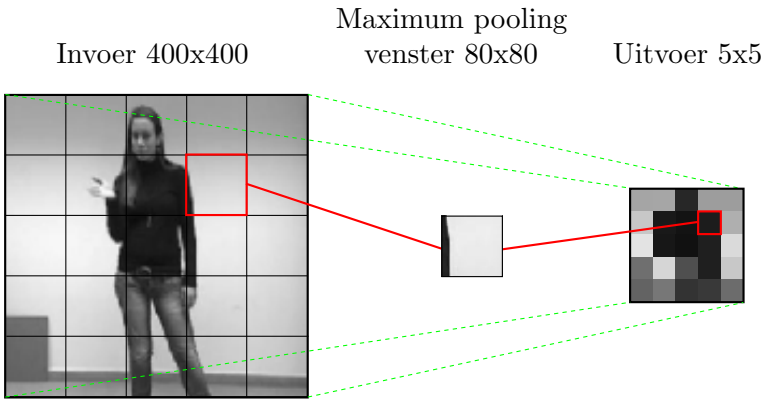
Figuur 2.8 geeft een visuele weergave van deze convolutie operatie terug. Op het resultaat van deze operatie wordt een niet-lineaire activatiefunctie toegepast en zo bekomen we een feature-map. Let er op dat we op de randen een probleem hebben: we krijgen te grote of te kleine indices voor de input. Om dit probleem te omzeilen zijn er twee opties: de invoer uitbreiden of enkel de convolutie berekenen wanneer alle invoerwaardes gedefinieerd zijn. Deze tweede vorm heet de *echte convolutie* en wordt hier toegepast. Hierdoor is de dimensie van de uitvoer (B_{uit}, H_{uit}) kleiner dan die van de invoer:

$$\begin{aligned} B_{uit} &= Bin - B_f + 1 \\ H_{uit} &= Hin - H_f + 1 \end{aligned} \tag{2.14}$$

Om meerdere features te detecteren per laag worden er meerdere feature-maps gevormd om een filterbank te bekomen. Hoeveel feature-maps er per laag worden berekend is een hyperparameter van het CNN. Hier is de tweedimensionale convolutie beschreven omdat in dit onderzoek enkel met deze convolutie gewerkt wordt. Er zijn ook driedimensionale convoluties die rekening met het tijdsaspect en zo spatio-temporele features extraheren.

2.3.2 Maximum pooling

Na de convolutielaag volgt er een pooling laag die de feature-maps opsplitst in kleine partities met een zekere pool grootte en de maximum waarde hieruit overhoudt. In Figuur 2.9 wordt een max-pooling uitgevoerd met een venster van 80x80 pixels op een invoer van 400x400 pixels. Zo daalt de dimensionaliteit maar blijven de belangrijkste features behouden.



Figuur 2.9: Maximum pooling: uit elk vak in de invoer afbeelding wordt het maximum bepaald en overgenomen in de bijhorende pixel in het uitvoerbeeld.

De pooling operatie zorgt voor translatie-invariantie van het herkenningssysteem. Of een hand nu een paar pixels naar links, rechts, boven of onder verschuift maakt niet uit. De pooling zal al deze activaties samenbrengen tot dezelfde positie voor de volgende featuremap.

2.4 Data-augmentatie

Hoe meer data we hebben om uit te leren, hoe beter we het model kunnen generaliseren. Deze data is niet altijd beschikbaar maar kan wel artificieel gecreëerd worden door data-augmentatie. Door bewerkingen uit te voeren op ware voorbeelden kunnen we nieuwe beelden maken die licht variëren.

Wanneer de gebruikte voorbeelden afbeeldingen zijn kunnen er beeldtransformaties worden uitgevoerd zoals translaties, rotaties, cropping en zooming. Er kan ook ruis toegevoegd worden en het beeld kan vervormd worden. De keuze van gebruikte augmentatie technieken en hun parameters zijn afhankelijk van de uit te voeren taak.

Hoofdstuk 3

Uitvoering & resultaten

3.1 Gebruikte technologiën

Alle programmatie wordt uitgevoerd in Python, een dynamische *high-level* programmeertaal. Python is ontwikkeld met het oog op leesbare en kernachtige code. Het is een van de meest gebruikte talen voor het uitvoeren van wetenschappelijke experimenten, mede dankzij de vele krachtige bibliotheken die beschikbaar zijn. In dit onderzoek worden de volgende Python-bibliotheken gebruikt:

- NumPy: voegt ondersteuning toe voor grote, multi-dimensionale arrays en matrices samen met een groot assortiment aan wiskundige functies om deze arrays efficiënt te manipuleren. Deze bibliotheek is een onderdeel van de SciPy-stack (Scientific Python).
- Theano [Bergstra et al., 2010]: een bibliotheek die toelaat wiskundige expressies te definiëren, optimaliseren en evalueren. De twee belangrijkste troeven van Theano zijn de dynamische generatie van c-code en het transparante gebruik van GPU-acceleratie. Expressies worden symbolisch omgezet en gecompileerd voor een snelle en efficiënte uitvoering. Ook worden ze geoptimaliseerd voor gebruik van de GPU. Hiernaast biedt de bibliotheek ook heel wat functies voor machine learning.
- Lasagne [Dieleman et al., 2015]: een lightweight bibliotheek voor het bouwen en trainen van neurale netwerken. Het biedt verschillende veelgebruikte kosten-, regularisatie-, activatie- en leerfuncties aan alsook vele soorten *layers*. Zo gebruikt dit werk de *DenseLayer* voor volledig verbonden lagen en de *Conv2DLayer* voor de tweedimensionale

convolutie. Deze bibliotheek bouwt verder op functionaliteit van Theano waardoor neurale netwerken gebouwd met Lasagne ook gebruik kunnen maken van GPU-acceleratie.

- Scikit-learn: deze bibliotheek biedt heel wat machine learning functionaliteit aan maar hiervan wordt in dit onderzoek geen gebruik gemaakt. Het pakket binnen scikit-learn dat wel gebruikt wordt, is *metrics*. Deze wordt gebruikt voor de evaluatie van het model. Vanuit de voorspelde en ware labels kunnen we via diverse functies de kwaliteit van de classificatie beschouwen. Dit pakket wordt voornamelijk gebruikt voor het berekenen van de precision en recall, en het opstellen van de confusion matrix. Deze termen worden verklaard in Sectie 3.4.4.

- Scikit-image: is een verzameling beeldverwerkingsalgoritmes. Deze bibliotheek wordt gebruikt voor de data-augmentatie.

3.2 Dataset

De gebruikte dataset is die van de "ChaLearn Looking at People Challenge 2014" zoals beschreven in [Escalera et al., 2014]. Uit deze set wordt de derde track gebruikt: "gesture recognition". Deze bevat meer dan 14000 gebaren uit een vocabularium van 20 Italiaanse gebaren. Merk op dat het hier niet gaat om een gebarentaal maar om afzonderlijke gebaren uit de Italiaanse cultuur. De verschillende gebaren en hun benamingen zijn te zien in 3.1.

De gebaren worden uitgevoerd door 27 gebruikers tegen verschillende achtergronden. Er is variatie op vlak van kleding, lichaamsbouw, belichting, gebaaruitvoering. De beelden zijn opgenomen met behulp van een Kinect camera waardoor er vier datastromen zijn: RGB-beeld, dieptebeeld, gebruikersindex en skeletinformatie (Figuur 3.2).

Deze dataset werd ook gebruikt in [Pigou, 2014] waar deze een aantal voorverwerkingsstappen onderging. Deze voorverwerkte dataset is ter beschikking gesteld van dit onderzoek. De voorverwerking probeert de dataset te optimaliseren voor gebarenherkenning. Eerst en vooral worden er vier nieuwe datastromen gecreëerd door het uitsnijden van de linker en rechterhand uit zowel diepte- als RGB-beeld. Ook wordt het bovenlichaam geïsoleerd om een deel van de achtergrond weg te werken. In totaal zijn er zo zes datastromen beschikbaar. De dataset


[0] Vattene


[1] Viene qui


[2] Perfetto


[3] E un furbo


[4] Che due palle


[5] Che vuoi


[6] Vanno d'accordo


[7] Sei pazzo


[8] Cos hai combinato


[9] Nonme me me friega niente


[10] OK


[11] Cosa ti farei


[12] Basta


[13] Le vuoi prendere


[14] Non ce ne piu


[15] Ho fame


[16] Tanto tempo fa


[17] Buonissimo


[18] Si sono d'accordo


[19] Sono stufo

Figuur 3.1: De Italiaanse gebaren gebruikt in de "ChaLearn Looking At People Challenge 2014, Track 3: gesture spotting"

RGB-beeld Dieptebeeld Gebruikersindex Skeletinformatie

Figuur 3.2: De vier datastromen beschikbaar in de "ChaLearn Looking At People Challenge 2014, Track 3: gesture spotting" dataset

bestaat uit 10000 geïsoleerde gebaren samen met hun correcte klasselabel. Elk beeld bestaat uit 32 frames en heeft een resolutie van 64x64 pixels.

In dit onderzoek wordt de dataset opgedeeld in 60% training-, 20% validatie- en 20% testset.

3.3 Onderzoeksopzet

De experimenten worden uitgevoerd op computers van het Reservoir Lab aan van de vakgroep Elektronica- en Informatiesystemen (ELIS) van de Universiteit Gent. De gebruikte computers hebben een hexacore processor (Intel Core i7-3930K) met kloksnelheid van 3.2 GHz en een NVIDIA Tesla K40c grafische kaart.

In een eerste fase moet er een convolutioneel neuraal netwerk worden opgezet die een aanvaardbare nauwkeurigheid behaald op de gebruikte dataset. Dit model wordt besproken in sectie 3.4. De nauwkeurigheden van dit predictief model getraind op alle klassen met alle voorbeelden wordt vervolgens gebruikt als vergelijkingspunt of *baseline* voor de one-shot learning experimenten.

In de volgende stap zal een model, qua architectuur en hyperparameters gelijkend op het basismodel, getraind worden op een deelverzameling van de gebaren. Na deze pre-training kan het model een gebaar bijgeleerd worden en wordt er geëxperimenteerd met het aantal verschillende samples van het nieuwe gebaar, het aantal lagen dat hertraind worden en met het gebruik van data-augmentatie. De bespreking van deze experimenten gebeurt in sectie 3.5.

3.4 Basismodel

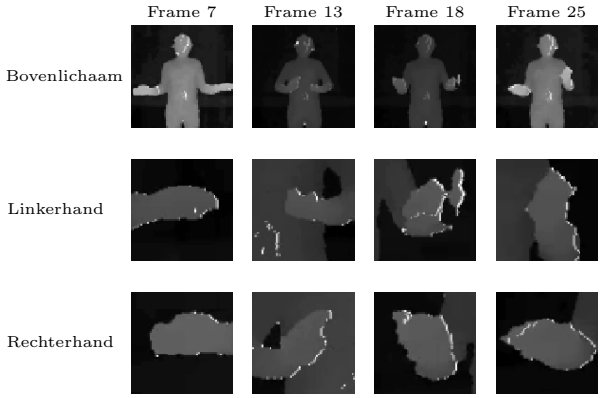
Alle keuzes omtrent hyperparameters en architectuur zijn experimenteel, op basis van gelijkaardige onderzoeken en naar advies van de begeleiding bepaald. Er zijn vele mogelijke combinaties qua hyperparameters en het is zeer tijdrovend om hierin de meest optimale keuze te maken. Dit is ook niet de essentie van dit onderzoek. Er dient een voldoende goed model opgesteld te worden om vergelijkingen te maken tussen het leren van alle gebaren (basismodel) en het bijleren van een gebaar.

3.4.1 Invoer

Het beeld van elk gebaar is 64x64 pixels groot en bestaat uit 32 frames. Hieruit worden er 4 frames met volgende indices geselecteerd: 7,13,18 en 25. Er is gekozen om niet van alle input gebruik te maken om een snellere trainingstijd te bekomen zodat er meer experimenten kunenn worden uitgevoerd. In een latere fase of bij verder onderzoek kan het model steeds uitgebreid worden om meer gebruik te maken van spatio-temporele data.

Er wordt enkel gebruik gemaakt van het dieptebeeld, volgens [Pigou, 2014] en [Wu en Shao, 2014] ligt hierin de nuttigste informatie voor een CNN. Het gebruik van RGB-beeld in combinatie met grijswaardenbeeld voegt meestal weinig toe. Hierin kan dus ook weer rekentijd worden gespaard.

Per voorbeeld zijn er 12 beelden zoals weergegeven in Figuur 3.3. De input van de invoerlaag heeft dus een dimensie van 12x64x64. Merk op dat de beelden van de handen en het volledige



Figuur 3.3: De 12 frames gebruikt voor de invoer van het CNN. De frames hebben een resolutie van 64x64 pixels.

bovenlichaam in hetzelfde CNN worden verwerkt. Alle uitgevoerde experimenten hanteren deze invoerdimensie.

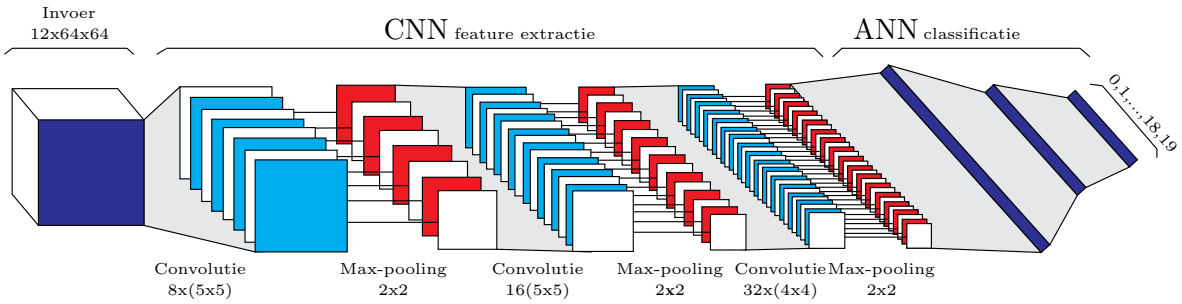
3.4.2 Architectuur

Het opgestelde basismodel is weergegeven in figuur 3.4 Het CNN bestaat uit drie convolutionele en max-pooling lagen. Alle max-pooling gebeurt met vensters van 2x2 zodat het beeld telkens gehalveerd wordt in grootte. De eerste convolutionele laag bevat 8 5x5 filters, de tweede 16 5x5 filters en de laatste 32 4x4 filters.

De output van dit CNN wordt gebruikt als featurevector in een ANN voor classificatie. Het ANN bevat ook drie lagen, de twee verborgen lagen hebben respectievelijk 800 en 100 knopen. Alle verborgen knopen zijn ReLU's en aan de uitgangslaag wordt een softmax gebruikt met 20 units voor de classificatie van de verschillende gebaren.

Een samenvatting van alle hyperparameters en gekozen architectuur is te vinden in Tabel 3.1. Voor de regularisatie wordt er gebruik gemaakt van dropout. De gewichten worden geïnitiaseerd met Glorot gewichten gesampled uit een uniforme verdeling. Deze gewichtsinitialisatie beschreven in [Glorot en Bengio, 2010] is ontwikkeld voor deep learning methodes en zorgt ervoor dat het signaal tot diep in het netwerk kan reiken. Het is de standaard initialisatie methode van het Lasagne framework.

Er wordt ook aan data-augmentatie gedaan om de prestatie van het predictief model te verbeteren. De parameters van deze augmentatie staan beschreven in Tabel 3.1. Deze parameters bleken later nogal conservatief gekozen, in Sectie 3.5.2 wordt hierover uitgebreid.



Figuur 3.4: Het basismodel bestaande uit een drielagig CNN voor feature-extractie en een drielagig ANN (met softmax uitvoerlaag) voor classificatie

Hyperparameter	*Waarde*
Filters CNN:	
Laag 1	8x(5x5)
Laag 2	16x(5x5)
Laag 2	32x(4x4)
Verborgen units ANN:	
Laag 1	800
Laag 2	100
Max-pooling vensters	2x2
Learning rate	10^{-4}
Batch grootte	32
Dropout kans	0.5
Nesterov momentum	0.9
Initialisatie:	
Gewichten CNN	Glorot initialisatie (uit uniforme verdeling $[-a, a]$)
Bias CNN	0
Gewichten ANN	Glorot initialisatie (uit uniforme verdeling $[-a, a]$)
Bias ANN	0
Data-augmentatie:	
Zoom	[83.33,120] %, uniforme verdeling
Rotatie	[-2,2]°, uniforme verdeling
2D translatie	[-2,2] pixels, uniforme verdeling

Tabel 3.1: Hyperparameters van het basismodel

3.4.3 Training

Mini-batch gradient descent aangevuld met Nesterov Augmented Momentum worden gebruikt als optimalisatie-algoritme. Alle training wordt uitgevoerd op de GPU waardoor we de CPU kunnen gebruiken om de voorbeelden aan te voeren. Dit verloopt als volgt:

- een werk-proces selecteert 32 random gekozen klasselabels
- per gekozen klasselabel wordt een random voorbeeld van die klasse gekozen uit de trainingset
- elk voorbeeld wordt geaugmenteerd met behulp van een transformatiematrix
- de voorbeelden worden in een array in het gedeelde geheugen geplaatst en opgehaald voor training van het model
- tijdens de training op deze mini-batch wordt de volgende klaargezet

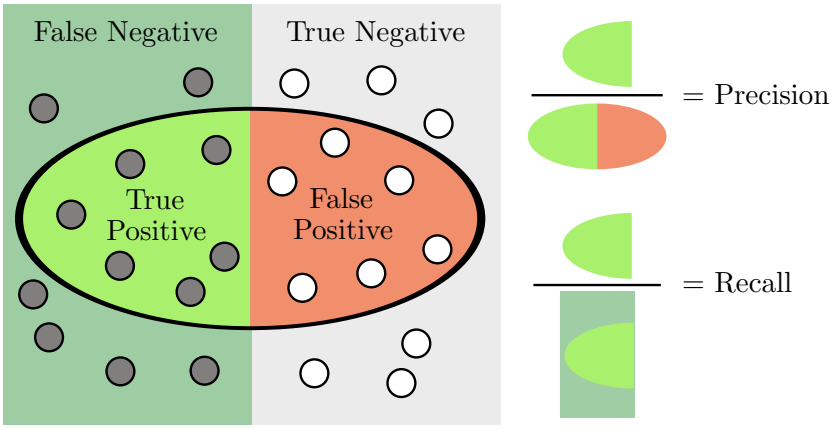
Na 250 iteraties, die we verder een epoch zullen noemen, wordt het model geëvalueerd met de validatieset. Het minimum van de validatiefout wordt bijgehouden en telkens we een lagere waarde bereiken worden de modelparameters opgeslaan. De training stopt wanneer er na een aantal epochs geen verbetering meer te zien is in de validatiefout.

3.4.4 Baseline resultaten

Om de baseline op te stellen wordt de volledige trainingset gebruikt om de 20 gebaren van de dataset te leren classificeren. Het voorgestelde model convergeert na ongeveer 250 epochs en behaalt een totale nauwkeurigheid van 81,20 %.

Precision, recall en confusion

In Figuur 3.6 is de genormaliseerde confusion matrix van het basismodel te zien. De mate van verwarring valt best mee, voor alle klassen ligt de waarde op de diagonaal relatief hoog. Een aantal klassen worden wel vaak verward zoals bijvoorbeeld klasse 3 en 9 ($\sim$15 % van klasse 3 en $\sim$20 % van klasse 9). Ook klasse 13 en 14 worden ondanks hun relatief hoge ware voorspellingswaarde ook vaak verward ($\sim$7 % van klasse 13 en $\sim$13 % van klasse 14)



Figuur 3.5: Visualisatie van de betekenis van precision en recall. Links zijn alle relevante elementen te zien die zouden moeten geselecteerd worden. Het omcirkelde gedeelte zijn de effectief geselecteerde elementen of de voorspellingen van het model

In Tabel 3.2 zijn alle precision en recall scores van het basismodel opgelijst. Deze scores zullen gebruikt worden als baseline voor de experimenten met het bijleren en one-shotten van een gebaar. Het zijn de waardes die we nastreven bij het hertrainen.

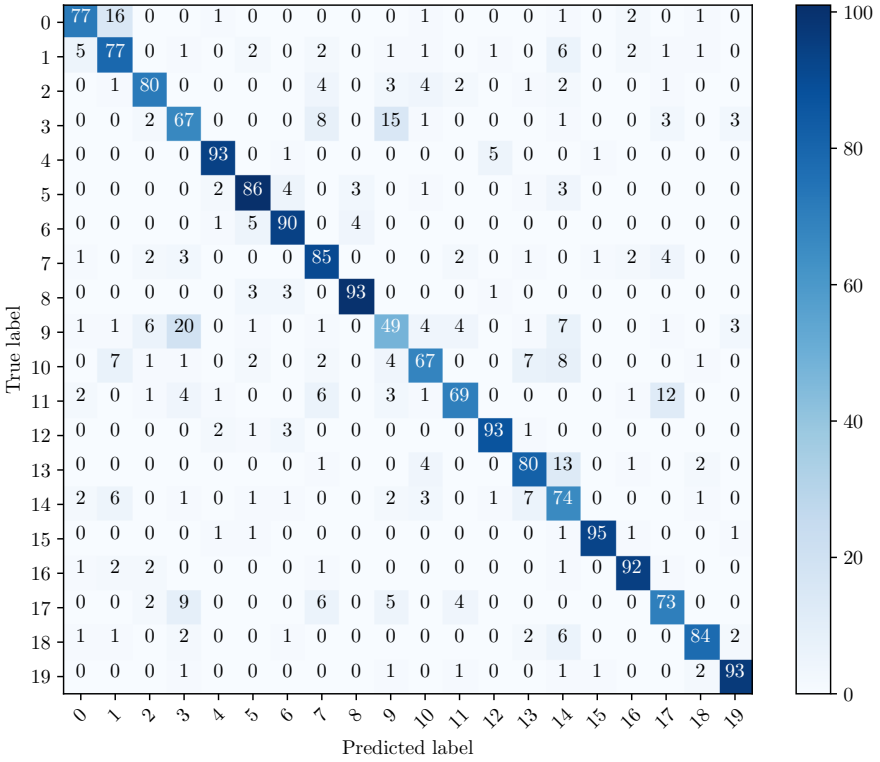
3.5 Een gebaar bijleren

3.5.1 Softmax20 model

De eerste experimenten rond het bijleren van een gebaar worden uitgevoerd op een model volledig gelijk aan het basismodel (zoals afgebeeld in Figuur 3.4). Eerst wordt het model getraind met een trainingset van 19 gebaren. Als het model geconvergeerd en genoeg geregulariseerd is, zou het een aantal algemene features van een nieuw gebaar moeten herkennen.

De pretraining gebeurt op een softmax van 20 units, een unit wordt dus niet gebruikt. Deze opstelling is niet bruikbaar om een dynamisch uitbreidbaar herkenningssysteem te maken omdat ze ervan uit gaat dat er slechts een gebaar zal worden toegevoegd. Het is wel een goede eerste test van het potentiëel van het model.

Door het model te hertrainen met de trainset van 19 gebaren aangevuld met een aantal samples van het nieuwe gebaar leren we het model ook op deze klasse te activeren. Om de hyperparameters van het model te optimaliseren wordt het na elke epoch geëvalueerd met de volledige validatieset. Deze bevat alle beschikbare voorbeelden van het nieuwe gebaar zodat



Figuur 3.6: Genormaliseerde confusion matrix van baseline voorspellingen van het basismodel, getraind op alle voorbeelden van alle gebaren

Label	Precision	Recall	Label	Precision	Recall
0	84,71 %	77,42 %	10	77,27 %	67,33 %
1	71,17 %	77,45 %	11	84,33 %	69,31 %
2	81,82 %	80,00 %	12	91,57 %	92,55 %
3	62,04 %	67,00 %	13	80,39 %	79,61 %
4	92,15 %	93,07 %	14	56,41 %	74,16 %
5	87,04 %	90,38 %	**15**	**96,88** %	**94,90** %
6	84,71 %	77,42 %	16	91,35 %	92,23 %
7	74,59 %	85,05 %	17	75,27 %	72,92 %
8	93,46 %	93,46 %	18	90,69 %	83,87 %
9	58,54 %	49,48 %	19	91,51 %	93,27 %

Tabel 3.2: Precision en recall voor de voorspellingen van het basismodel getraind op alle voorbeelden van alle gebaren.

er kan gelet worden op overfitting van het model op het kleine aantal trainingsvoorbeelden.

Een finale evaluatie wordt gemaakt met alle voorbeelden uit de testset. Uit de voorspelling op deze testset worden de precision en recall scores van de klasse berekend en geëvalueerd.

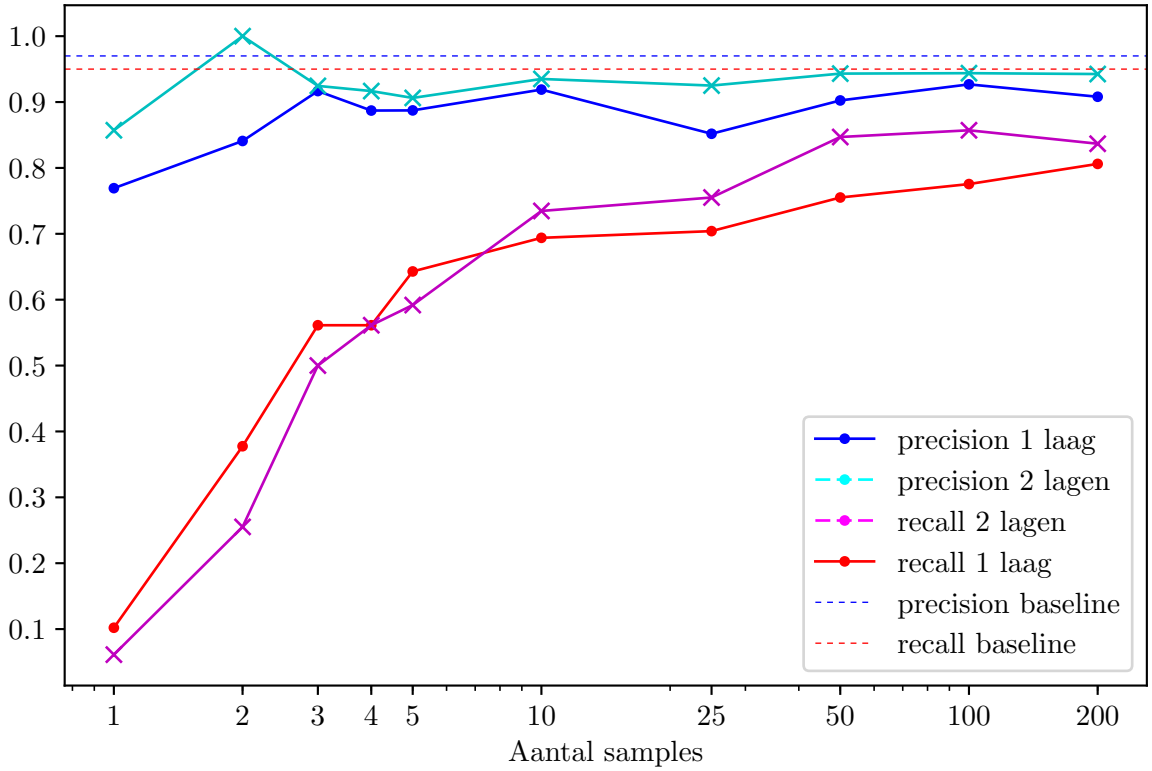
Eerst en vooral wordt het aantal samples gevariëerd. Het model wordt getraind met 200, 100, 50, 25, 10, 5, 4, 3, 2 voorbeelden en ten slotte wordt er 1 voorbeeld gebruikt voor een poging tot one-shot learning.

Ook wordt het effect van het aantal hertrainde lagen onderzocht. Eerst wordt enkel de laatste laag, de softmax, hertraind. Alle andere parameters van het model blijven dus ongewijzigd. In een tweede experiment worden de softmax en de laatste verborgen laag, deze met 100 units, hertraind (dense100 laag in Figuur 3.4).

In figuur 3.7 zijn de resultaten van deze experimenten met het *softmax20* model te zien. Het bijgeleerde gebaar is het gebaar nummer 15: "non ce ne piu". Het basismodel getraind op alle voorbeelden behaalde een 96.88% precision en 94.90% recall score op dit gebaar, de hoogste scores van alle gebaren. Enerzijds zou dit gebaar dus gemakkelijk moeten aan te leren zijn aangezien het heel nauwkeurig voorspeld wordt. Anderzijds kan het model zich niet meer beroepen op de features geleerd uit dit gebaar aangezien ze niet in de pretraining is opgenomen.

Het is opvallend dat de precision scores voor beide experimenten heel dicht liggen bij de baseline. Het model is dus redelijk zeker dat het om het nieuwe gebaar gaat als het dit gebaar herkent. Bij het hertrainen van de twee laatste lagen met twee voorbeelden gaat de precisie zelfs boven die van de baseline.

De recall score volgt niet dezelfde lijn als die van de precision. Hier zien we duidelijk dat de recall bij gebruik van weinig samples heel laag ligt. De stijlste klimming bij het model waar enkel de softmax wordt hertraind zien we tot aan het gebruik van 5 samples. Bij het hertrainen van twee lagen loopt deze stijging door tot 10 samples. Dit kan verklaard worden doordat er meer parameters geoptimaliseerd moeten worden bij het het hertrainen van de twee lagen. Meer parameters vragen om meer voorbeelden, bij gebruik van meer dan 10 samples scoort dit model dan ook beter.



Figuur 3.7: Precision en recall voor de classificatie van gebaar 15 met het softmax20 model in functie van het aantal gebruikte samples voor bijleren. De streepjeslijnen geven de baseline weer van het model getrained op alle voorbeelden.

Het voorgetrainde model, GARBAGE CLASS

3.5.2 Softmax19+1 model

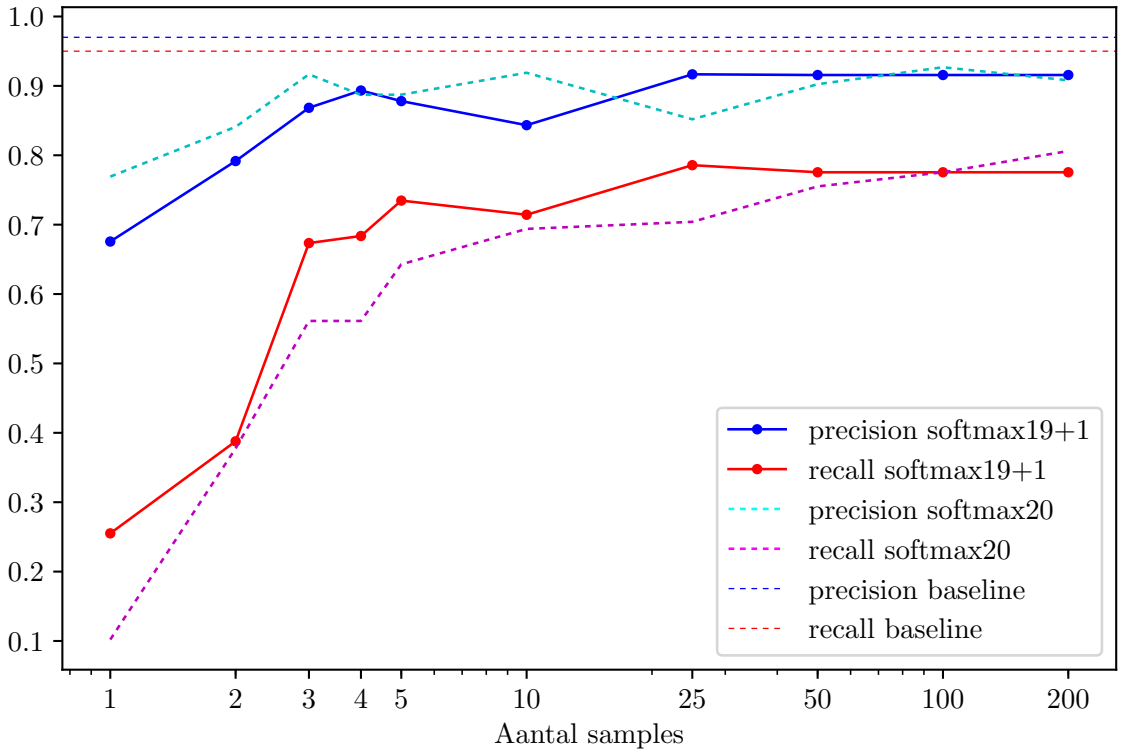
We willen naar een dynamisch uitbreidbaar systeem toewerken. Het vorige model gaat uit van slechts een bij te leren gebaar en is hier dus niet bruikbaar voor. Om aan deze uitbreidbaarheid te voldoen en om verder onderzoek te voeren wordt het *Softmax19+1* model opgesteld, te zien in Figuur **??**.

Hier wordt eerst een model met 19 softmax units getraind voor de classificatie van 19 gebaren. Behalve het verschil in aantal softmax units is de architectuur volledig gelijk aan die van het basismodel. Daarna wordt het model uitgebreid met een extra unit voor de classificatie van het nieuwe gebaar. De softmax activatiefunctie van de laag met 19 eenheden wordt vervangen door een lineaire activatie ($a(x) = x$). De signalen van deze units worden samengebracht met het signaal van de nieuwe unit, ook met een lineaire activatiefunctie, in een *ConcatLayer*. Op deze laatste laag, die nu 20 eenheden telt, wordt dan de softmax uitgevoerd voor classificatie.

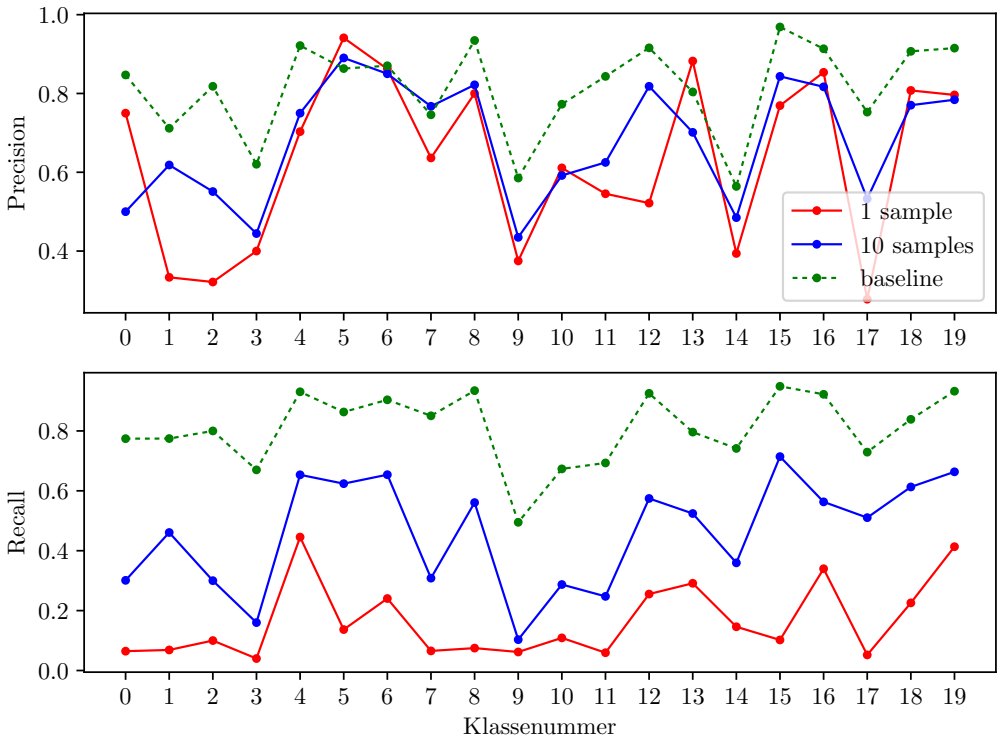
Het leren gebeurt enkel op de ene unit van het nieuwe gebaar. Aan de verborgen laag van de 19 al geleerde gebaren wordt niet geraakt zodat het herkenningsvermogen op deze gebaren gelijk blijft. Dit zal ook een vereiste zijn van een uitbreidbaar herkenningssysteem, het bijleren van een nieuw gebaar mag geen negatief effect hebben op de huidige prestatie van het model.

Invloed van het aantal samples

Het softmax19+1 model wordt getoetst tegen de prestatie van het softmax20 model. In Figuur 3.8 is de precision en recall van de classificatie van gebaar 15 te zien in functie van het aantal samples dat gebruikt wordt in de trainingset. Ook de prestatie van het softmax20 model bij hertrainen van enkel de softmax laag is geplot in streepjeslijnen. De waarden liggen niet zo heel ver uiteen. De precision van het softmax19+1 model ligt iets lager bij gebruik van minder dan 10 samples. Wel ligt de recall score globaal hoger dan die van het softmax20 model. Bij het trainen op een voorbeeld is dit 25,51 % tegenover 10,20%. De recall curve volgt dezelfde stijle klim tot aan het gebruik van 5 samples, tot 73,47% waarna het nog licht stijgt bij 25 samples en dan rond de 78% blijft.



Figuur 3.8: Precision en recall voor de classificatie van gebaar 15 met het softmax19+1 model in functie van het aantal gebruikte samples voor bijleren.



Figuur 3.9: Barplots per gebaar van de precision en recall van de baseline en na het hertrainen met respectievelijk 1 en 10 samples.

Invloed van het bijgeleerde gebaar

De prestatie van het softmax19+1 model volgt dus dezelfde trend als die van het softmax20 model. De resultaten hier weergegeven zijn allemaal na het bijleren van gebaar 15. Welk gebaar er bijgeleerd wordt heeft uiteraard ook een invloed op de prestatie van het model. Sommige gebaren hebben vele features gemeenschappelijk met andere gebaren, andere zijn eerder uniek en zullen moeilijk herkend worden.

Om een beeld te scheppen van de invloed van het bij te leren gebaar werd voor elk gebaar eerst een model getraind exclusief dit gebaar, waarna het werd bijgeleerd met 1 en met 10 samples. Op Figuur 3.9 zijn de resultaten van dit experiment te zien. Bovenaan is de precision uitgeplot van alle gebaren, onderaan de recall.

De recall score na bijleren met 1 en met 10 samples volgt dezelfde lijn als die van de baseline. Gebaren die een lagere recall score hebben na trainen op alle voorbeelden van alle gebaren scoren ook lager als we ze bijleren. Hetzelfde geldt voor gebaren die hoog scoren. Zeker bij vergelijking van het gebruik van 10 samples is dit opmerkelijk.

De precision volgt ook dezelfde trend maar de verschillen zijn veel kleiner. De bevindingen uit het eerste experiment op gebaar 15 zijn dus door te trekken naar de andere gebaren.

Invloed van data-augmentatie

Ten slotte wordt ook nog een experiment opgezet om de invloed van data-augmentatie te onderzoeken. Tot voor dit experiment werd data-augmentatie toegepast met de parameters beschreven in Tabel 3.1. Nieuwe testen worden opgezet die geen augmentatie gebruiken en hier is duidelijk dat de augmentatie weinig tot geen effect heeft. De gekozen parameters blijken erg conservatief en brengen te weinig variatie in de data. Er wordt geprobeerd om een meer drastische data-augmentatie toe te passen. Om met de grotere variatie om te kunnen gaan moet het aantal parameters van het model verhoogd worden. Dit is uitgetest met een model met het dubbele aantal filterbanken in de convolutielagen en twee maal 512 units in de verborgen lagen van het ANN. Na analyse blijkt het model te underfitten: het is nog niet complex genoeg om de geaugmenteerde data te beschrijven. Om deze opstelling te optimaliseren dient er opnieuw een iteratief proces van hyperparameteroptimalisatie aangevat

te worden. Hier is niet verder op ingegaan, het gebruik van data-augmentatie kan wel verder onderzocht in toekomstig werk.

3.5.3 Softmax18+1+1 model

Om de uitbreidbaarheid verder te onderzoeken wordt analoog aan het principe van het softmax19+1 model een architectuur gemaakt dat twee gebaren na elkaar bijleert (Figuur **??**). Eerst wordt een model met 18 softmax units getraind. Na deze pretraining wordt er een unit voor het bij te leren gebaar toegevoegd en net zoals het vorige model worden de uitgangen van de 18 units en de nieuwe unit samengevoegd met een ConcatLayer.

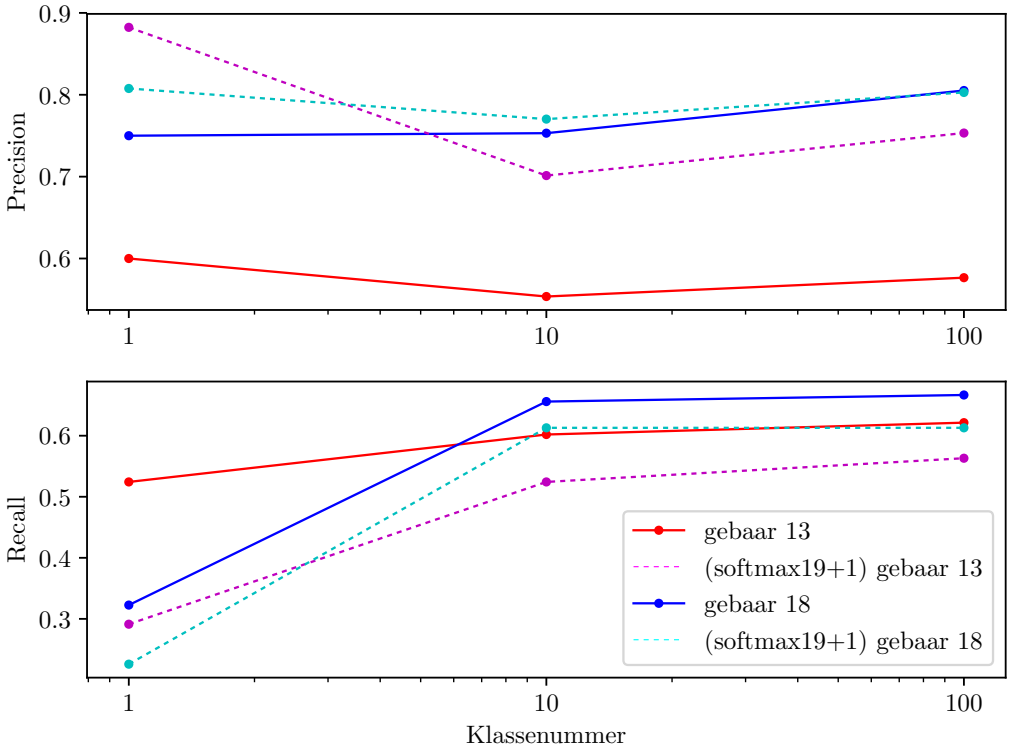
Het model leert een gebaar bij door de training van het nieuw toegevoegde unit en wordt daarna opnieuw uitgebreid met nog een extra unit voor het tweede gebaar. Ook deze unit wordt geconcateneerd zodat er uiteindelijk een softmax van 20 units is (18+1+1).

Door op deze manier te werk te gaan is het theoretisch mogelijk om gebaren te blijven toevoegen. Na het toevoegen van een aantal gebaren kunnen de units effectief samengevoegd worden tot een laag waarna het proces opnieuw kan beginnen.

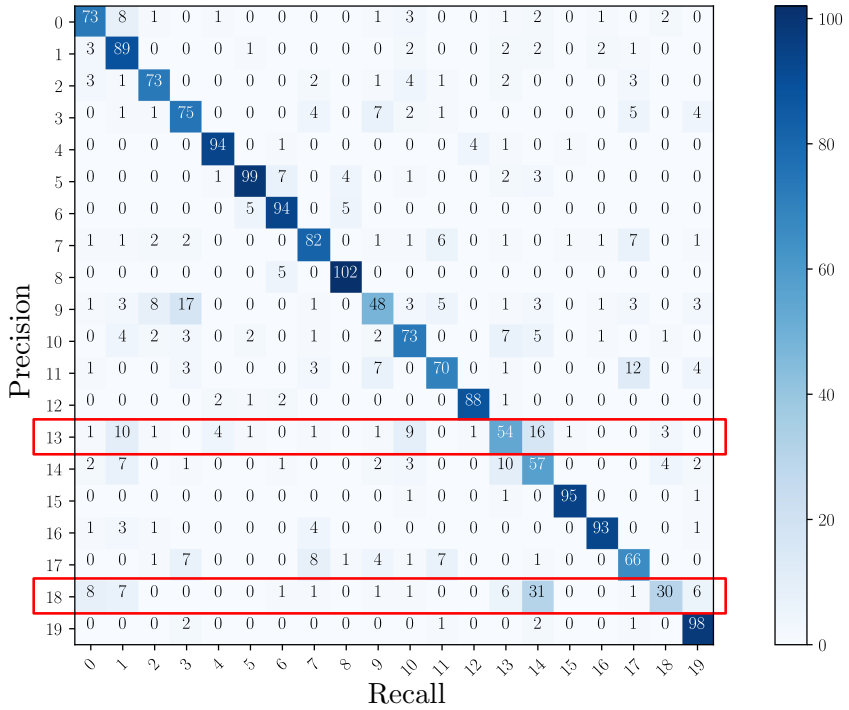
In Figuur 3.10 zijn de resultaten te zien van het bijleren van gebaar 13 en gebaar 18. Eerst wordt gebaar 13 bijgeleerd, daarna gebaar 18, telkens met even veel voorbeelden. De recall score van beide gebaren ligt hoger dan wanneer ze elk afzonderlijk worden bijgeleerd in het softmax19+1 model. De precision ligt dan wel weer lager, vooral bij gebaar 13 dat eerst werd aangeleerd.

Waarom de recall hoger ligt dan bij het softmax19x1 model valt moeilijk te verklaren. Het bijleren van een gebaar blijkt in al de experimenten erg afhankelijk van welk gebaar wordt bijgeleerd en uit welke gebaren wordt voorgeleerd. Om dit verder uit te zoeken zouden er meer experimenten met andere gebaren moeten worden uitgevoerd.

Figuur 3.11 geeft de genormaliseerde confusion matrix weer van het bijleren uit een voorbeeld. Hierop is te zien dat er een kleine verwarring is tussen de twee nieuw aangeleerde gebaren: 6% van gebaar 18 wordt verward met 13 en omgekeerd 3%. De grootste verwarring bij gebaar 18 is met gebaar 14: 31%. In het baseline model is dit ook de grootste mate van verwarring bij gebaar 18 (6%, te zien in Figuur 3.6).



Figuur 3.10: Precision en recall na het bijleren van gebaar 13 en gebaar 18 in het *softmax18+1+1* model. Ter vergelijking is het resultaat van het *softmax19+1* model ook uitgetekend.



Figuur 3.11: Genormaliseerde confusion matrix van het *softmax18+1+1* model na leren van gebaar 13 en gebaar 18 met een voorbeeld.

Hoofdstuk 4

Conclusie

4.1 Besluit

In deze masterproef wordt aangetoond dat het mogelijk is om een gebaar met een beperkt aantal voorbeelden bij te leren aan een convolutioneel neuraal netwerk, met behulp van het dieptebeeld van de Microsoft Kinect en GPU-acceleratie.

In alle experimenten is te zien dat een groter aantal voorbeelden een positieve invloed heeft op de sensitiviteit of recall van de herkenning van het bijgeleerde gebaar. De stijging in deze waarde is het sterkst tussen het gebruik van 1 tot 10 samples. Voor het ideaal van one-shot learning ligt deze waarde voor 10 van de 20 gebaren boven de 10%. Wanneer er 10 samples gebruikt worden zijn er 11 gebaren die meer dan 50% sensitiviteit tonen. Het gebruikte model is relatief eenvoudig gehouden om meer experimenten te kunnen uitvoeren maar behaalt toch behoorlijke resultaten op deze uitdagende taak.

Zowel precision als recall van het bijleren van een gebaar liggen in lijn met de prestaties van het basismodel waarin alle gebaren uit de trainset samen worden aangeleerd met alle voorbeelden.

Door het bijleren van twee nieuwe gebaren na elkaar wordt aangetoond dat het model dynamisch uitbreidbaar is. De sensitiviteit op de twee nieuw bijgeleerde gebaren van het experiment ligt zelfs hoger dan wanneer ze elk afzonderlijk worden bijgeleerd.

De kloof tussen het herkenningsvermogen na bijleren met een beperkt aantal voorbeelden en

het meteen aanleren van alle gebaren is wel nog steeds groot.

4.2 Verder onderzoek

Er is een eerste aanleiding gegeven tot one-shot learning en het dynamisch uitbreiden van het lexicon van een gebarenherkenningssysteem. Er is veel ruimte voor verbetering en verder onderzoek.

Ten eerste is er weinig voorverwerking gedaan op de gebruikte data. Een extra voorverwerkingsstap kan bijvoorbeeld de achtergrond filteren en betere ruisonderdrukking voorzien. Zo zullen er betere en meer algemene features worden aangeleerd en gedetecteerd. Er kan ook gebruik gemaakt worden van alle frames van het beeld en driedimensionale convoluties om rekening te houden met het tijdsaspect.

Het kan interessant zijn om een meer complex en state-of-the-art netwerk op te stellen en te trainen met een grotere dataset zoals de "Chalearn Isolated Gesture Recognition (ICPR '16)" dataset uit [Wan et al., 2016]. Deze bevat 48000 voorbeelden, bijna vijf keer zo veel als de ChaLearn LAP 2014 dataset die hier gebruikt werd, en beslaat 249 verschillende gebaren. Doordat er zo veel verschillende gebaren in deze dataset zijn kunnen er nog meer algemene features geëtraheerd worden. Meer algemene features levert meer potentiëel voor het bijleren van een gebaar en de dynamische uitbreiding van het herkenbaar lexicon.

Er kan enkel met deze nieuwe dataset gewerkt worden op een analoge manier als in deze masterproef of er kan gekozen worden om gebaren bij te leren uit een andere dataset, zoals bijvoorbeeld de LAP 2014 dataset.

Vanuit intuïtie moet data-augmentatie veel helpen bij het bijleren vanuit weinig voorbeelden. Een doordachte data-augmentatie techniek die translaties, rotaties, vervorming en ruis toevoegt kan gunstige resultaten brengen op voorwaarde dat het model dat hiervoor wordt gebruikt genoeg parameters heeft om de grotere variatie te beschrijven.

Bibliografie

[Bergstra et al., 2010] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., en Bengio, Y. (2010). Theano: A CPU and GPU math compiler in Python. In *Proc. 9th Python in Science Conf*, pages 1–7.

[Botev et al., 2016] Botev, A., Lever, G., en Barber, D. (2016). Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent. *arXiv:1607.01981 [cs, stat]*. arXiv: 1607.01981.

[Buyens, 2003] Buyens, M. (2003). *Gebarentaaltolken*. Garant.

[CAB-Vlaanderen, 2017] CAB-Vlaanderen (2017). Welzijn en werk in het Nederlands.

[Caelles et al., 2016] Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-TaixÃ©, L., Cremers, D., en Van Gool, L. (2016). One-Shot Video Object Segmentation. *arXiv:1611.05198 [cs]*. arXiv: 1611.05198.

[Carbonell et al., 1983] Carbonell, J. G., Michalski, R. S., en Mitchell, T. M. (1983). An Overview of Machine Learning. In Michalski, R. S., Carbonell, J. G., en Mitchell, T. M., editors, *Machine Learning*, Symbolic Computation, pages 3–23. Springer Berlin Heidelberg. DOI: 10.1007/978-3-662-12405-5_1.

[Dieleman et al., 2015] Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., et al. (2015). Lasagne: First release. *Zenodo: Geneva, Switzerland*.

[Escalera et al., 2014] Escalera, S., Baro, X., Gonzalez, J., Bautista, M. A., Madadi, M., Reyes, M., Ponce-Lopez, V., Escalante, H. J., Shotton, J., en Guyon, I. (2014). ChaLearn

Looking at People Challenge 2014: Dataset and Results. In *Computer Vision - ECCV 2014 Workshops*, pages 459–473. Springer, Cham.

[Fei-Fei et al., 2006] Fei-Fei, L., Fergus, R., en Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

[Glorot en Bengio, 2010] Glorot, X. en Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

[Glorot et al., 2011] Glorot, X., Bordes, A., en Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. In *Aistats*, volume 15, page 275.

[Guo en Yang, 2017] Guo, X.-L. en Yang, T.-T. (2017). Gesture recognition based on HMM-FNN model using a Kinect. *Journal on Multimodal User Interfaces*, 11(1):1–7.

[Hubel en Wiesel, 1968] Hubel, D. H. en Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243.

[Jhuang et al., 2007] Jhuang, H., Serre, T., Wolf, L., en Poggio, T. (2007). A biologically inspired system for action recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. Ieee.

[Ji et al., 2013] Ji, S., Xu, W., Yang, M., en Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231.

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., en Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

[Krizhevsky et al., 2012a] Krizhevsky, A., Sutskever, I., en Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Krizhevsky et al., 2012b] Krizhevsky, A., Sutskever, I., en Hinton, G. E. (2012b). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., en Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Kuehne et al., 2011] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., en Serre, T. (2011). Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE.

[Kühn, 2011] Kühn, T. (2011). The kinect sensor platform. *Proc. Advances in Media Technology*, pages 1–4.

[Lake et al., 2011] Lake, B. M., Salakhutdinov, R., Gross, J., en Tenenbaum, J. B. (2011). One shot learning of simple visual concepts. In *CogSci*, volume 172, page 2.

[Perronnin et al., 2010] Perronnin, F., Sánchez, J., en Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. *Computer Vision–ECCV 2010*, pages 143–156.

[Pigou, 2014] Pigou, L. (2014). Gebarentaalherkenning met convolutionele neurale.

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., en Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[Van Herreweghe en Vermeerbergen, 2009] Van Herreweghe, M. en Vermeerbergen, M. (2009). Flemish Sign Language standardisation. *Current Issues in Language Planning*, 10(3):308–326.

[Wan et al., 2016] Wan, J., Zhao, Y., Zhou, S., Guyon, I., Escalera, S., en Li, S. Z. (2016). Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 56–64.

[Wu et al., 2016] Wu, D., Pigou, L., Kindermans, P.-J., Nam, L. E., Shao, L., Dambre, J., en Odobez, J.-M. (2016). Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition.

[Wu en Shao, 2014] Wu, D. en Shao, L. (2014). Deep dynamic neural networks for gesture segmentation and recognition. In *Workshop at the European Conference on Computer Vision*, pages 552–571. Springer.

[Wu et al., 2012] Wu, D., Zhu, F., en Shao, L. (2012). One shot learning gesture recognition from RGBD images. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–12.

[Yang et al., 2010] Yang, R., Sarkar, S., en Loeding, B. (2010). Handling Movement Epenthesis and Hand Segmentation Ambiguities in Continuous Sign Language Recognition Using Nested Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):462–477.

Lijst van figuren