

# 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



# Toelating tot bruikleen

“De auteur geeft de toelating deze scriptie voor consultatie beschikbaar te stellen en delen van de scriptie te kopiëren voor persoonlijk gebruik.

Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze scriptie.”

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

# Inhoudsopgave

<b>Voorwoord</b>	<b>iii</b>
<b>Overzicht</b>	<b>iv</b>
<b>Afkortingen</b>	<b>vii</b>
<b>1 Inleiding</b>	<b>1</b>
1.1 Gebarentaal . . . . .	1
1.2 Automatische gebarentaalherkenning . . . . .	2
1.2.1 Gebarensegmentatie . . . . .	4
1.2.2 Gebarenherkenning . . . . .	4
1.2.3 Grammaticale samenstelling . . . . .	5
1.3 One-shot learning . . . . .	5
1.3.1 Uitbreidbaarheid herkenningssysteem . . . . .	5
1.3.2 Bijleren bij mensen . . . . .	6
1.3.3 One-shot learning in de literatuur . . . . .	6
1.4 Doelstelling . . . . .	7
<b>2 Technische aspecten</b>	<b>8</b>
2.1 Machine Learning . . . . .	8
2.1.1 Inleiding . . . . .	8
2.1.2 Gesuperviseerde classificatie . . . . .	9
2.1.3 Overfitting . . . . .	10
2.2 Artificieel neurale netwerk . . . . .	12
2.2.1 Inleiding . . . . .	12
2.2.2 Leren . . . . .	15
2.3 Convolutioneel neurale netwerk . . . . .	15
2.3.1 Tweedimensionale convolutie . . . . .	15
2.3.2 Maximum pooling . . . . .	15
2.4 Hyperparameters . . . . .	15
2.4.1 Momentum . . . . .	15
2.4.2 Dropout . . . . .	15
2.4.3 Data-augmentatie . . . . .	15

---

<b>3</b>	<b>Uitvoering &amp; resultaten</b>	<b>16</b>
3.1	Gebruikte technologieën . . . . .	16
3.2	Dataset . . . . .	17
3.3	Onderzoeksopzet . . . . .	19
3.4	Basismodel . . . . .	20
3.4.1	Invoer . . . . .	20
3.4.2	Architectuur . . . . .	21
3.4.3	Training . . . . .	23
3.4.4	Baseline resultaten . . . . .	23
3.5	Een gebaar bijleren . . . . .	24
3.5.1	Softmax20 model . . . . .	24
3.5.2	Softmax19+1 model . . . . .	28
3.5.3	Softmax18+1+1 model . . . . .	31
<b>4</b>	<b>Conclusie</b>	<b>33</b>
4.1	Besluit . . . . .	33
4.2	Verder onderzoek . . . . .	33
	<b>Bibliografie</b>	<b>34</b>
	<b>Lijst van figuren</b>	<b>37</b>

# 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 communicatiemiddel 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 armbewegingen wordt een woord “uitgesproken” en vervolgens visueel waargenomen.

Een gebarentaal ontstaat, 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 "finger-spitzengefühl" 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 gecommuniceerd 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 objectcategorieën 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 технологиën 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 wanneer 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 gebaarherkenningssysteem 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 Levensteijnafstand 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 gedetecteerd met een nauwkeurigheid lager dan 45 %.

[Caelles et al., 2016] stelt een 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 prestatie maat 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 categorieën 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 gelijkens (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 categorieën om bijvoorbeeld objecten te herkennen zoals in [Krizhevsky et al., 2012]. 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  $T$ :

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

met  $x^{(n)}$  het  $n$ -de datavoorbeeld en  $y^{(n)}$  zijn klasselabel.  $C$  staat voor het aantal discrete categorieën of klassen,  $D$  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  $\theta$ :

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

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

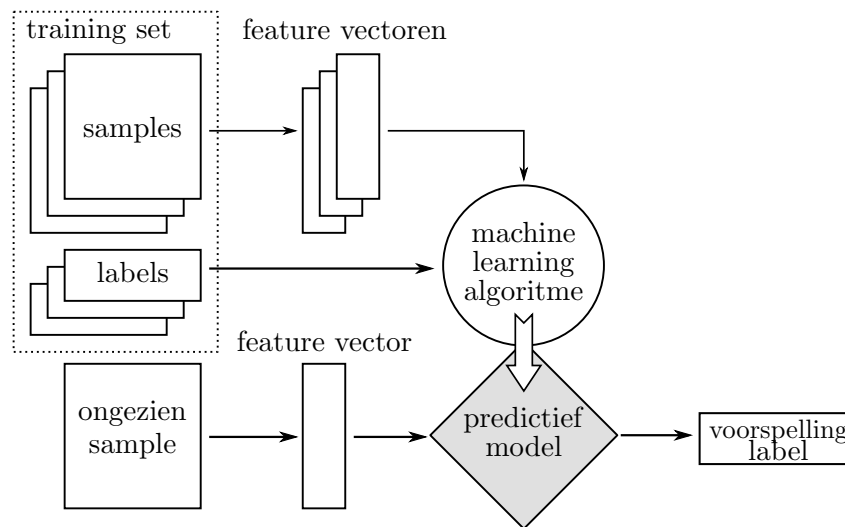
Figuur 2.1 geeft een schematische weergave van de opbouw van een classificatiemodel. Het schatten van  $f$  en  $\theta$  wordt uitgevoerd door een techniek uit de machine learning en geeft ons het uiteindelijke predictief model  $f(x, \theta) = y$ . De samples worden meestal verwerkt voor ze als input 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 neurale netwerk, een veel gebruikt predictief model voor classificatie. In sectie 2.2 wordt deze techniek dan ook verder besproken.

### 2.1.3 Overfitting

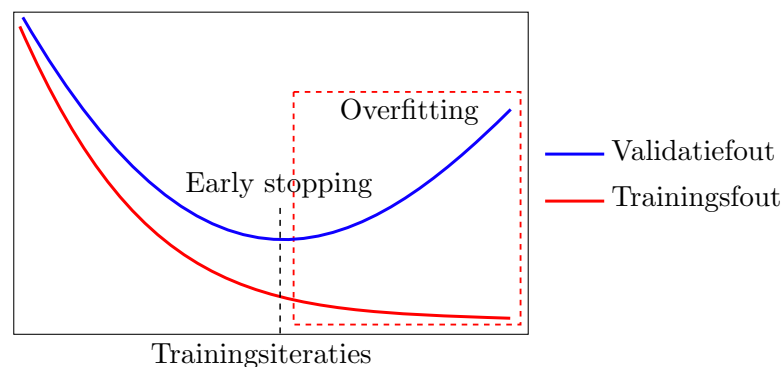
Stel dat je naar een symfonisch orkest gaat luisteren in een schouwburg en je wilt het meest klare en pure geluid mogelijk ervaren. Je koopt een hooggevoelige microfoon en hoorapparaat en probeert zo alle geluiden op te nemen in het auditorium. Nu hoor je naast het geluid van het symfonisch orkest ook alle ruis. Je hoort de mensen schuiven in hun stoelen, de muzikanten die hun muziekpapier omslaan en het gezwiep van de dirigent zijn baton. Dit fenomeen wordt overfitting genoemd. Als je op een concert bent is er zowel de symfonie als de willekeurige ruis. Een perfect en algemeen model luistert enkel naar de symfonie. Overfitting is wanneer je meer ruis hoort dan noodzakelijk, of erger, als je de ruis boven de symfonie hoort.



**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 prestatie maat 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. De testset mag nooit gebruikt worden om aanpassingen aan het model te maken maar dient enkel voor een eindevaluatie.

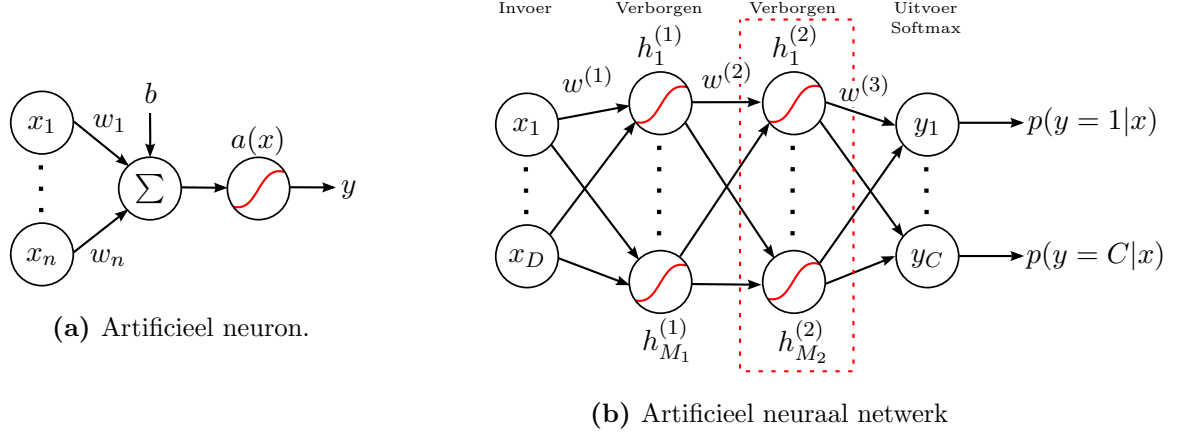
## 2.2 Artificieel neurale netwerk

### 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 bedrever 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 neurale netwerk (ANN) uit te bouwen. Het menselijke brein is een erg complex neurale 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 2.3a is een weergave te zien van een artificieel neuron, de bouwsteen van het ANN. De functionaliteit van dit neuron bootst deze van het biologische neuron na en wordt als volgt gedefinieerd:

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



**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 2.3b. 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.

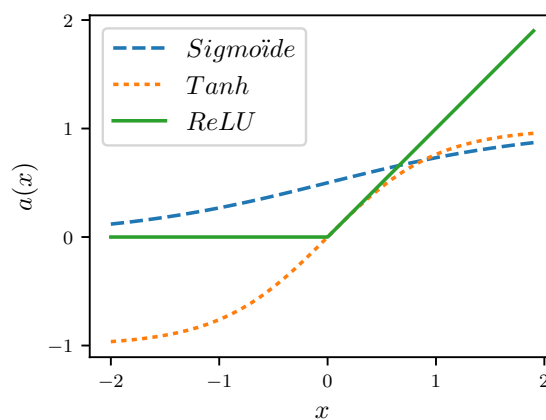
Hier beschouwen we enkel *feedforward neurale netwerken*. Aan het begin van het netwerk hebben we de invoerlaag met evenveel knopen als de dimensie van de featurevectoren. De invoer propageert zich van links naar rechts door het netwerk tot het bij de uitvoerlaag komt. Omdat de uitvoer van de binnenste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

De uitvoer van de verschillende lagen uit het ANN in figuur 2.3b kunnen we als volgt beschrijven:

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

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 2.3b 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 neurale 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:

$$a(x) = \max(0, x) \quad (2.5)$$

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.

### 2.2.2 Leren

## 2.3 Convolutioneel neuraal netwerk

De vloek van de dimensionaliteit:

### 2.3.1 Tweedimensionale convolutie

### 2.3.2 Maximum pooling

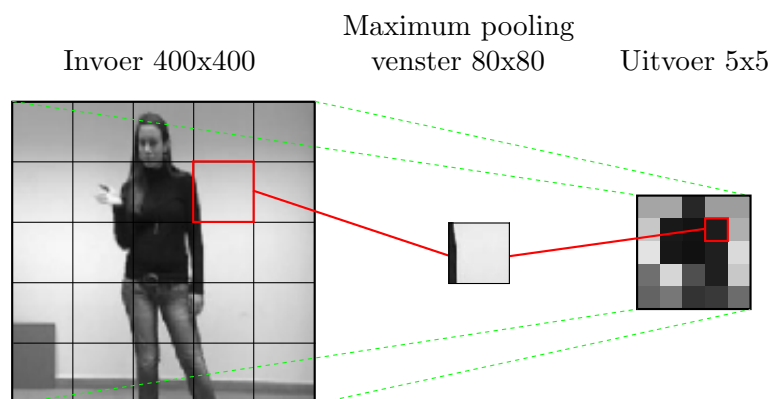
## 2.4 Hyperparameters

### 2.4.1 Momentum

### 2.4.2 Dropout

[Srivastava et al., 2014]

### 2.4.3 Data-augmentatie



**Figuur 2.5:** Maximum pooling: uit elk vak in de invoer afbeelding wordt het maximum bepaald en overgenomen in de bijhorende pixel in het uitvoerbeeld.

## Hoofdstuk 3

# Uitvoering & resultaten

### 3.1 Gebruikte technologieën

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] Non me me  
frigga niente

[10] OK



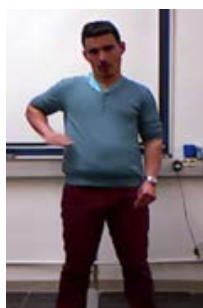
[11] Cosa ti farei



[12] Basta

[13] Le vuoi  
prendere

[14] Non ce ne piu



[15] Ho fame

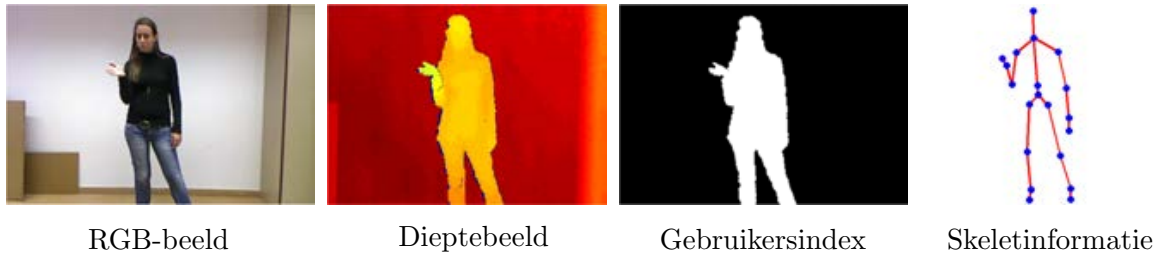
[16] Tanto  
tempo fa

[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”



**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 neurale netwerk worden opgezet die een aanvaardbare nauwkeurigheid behaalt 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

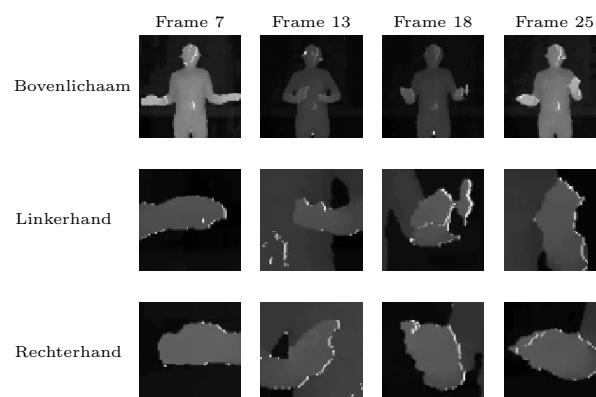
Alle keuzes omtrent hyperparameters en architectuur zijn experimenteel, op basis van gelijkaardige onderzoeken en naar advies van de begeleiding bepaald. Er zijn vele mogelijke combinaties qua hyperparameters en het is zeer tijdrovend om hierin de meest optimale keuze te maken. Dit is ook niet de essentie 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 kunnen 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.

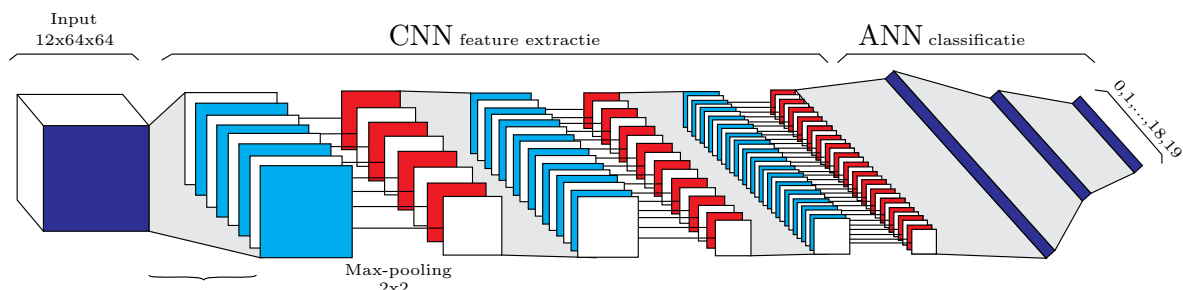
### 3.4.2 Architectuur

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  $2 \times 2$  zodat het beeld telkens gehalveerd wordt in grootte. De eerste convolutionele laag bevat 8  $5 \times 5$  filters, de tweede 16  $5 \times 5$  filters en de laatste 32  $4 \times 4$  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 geïnitieerd 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

<i>Hyperparameter</i>	<i>Waarde</i>
<b>Filters CNN:</b>	
Laag 1	8x(5x5)
Laag 2	16x(5x5)
Laag 2	32x(4x4)
<b>Verborgen units ANN:</b>	
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
<b>Initialisatie:</b>	
Gewichten CNN	Glorot initialisatie (uit uniforme verdeling $[-a, a]$ )
Bias CNN	0
Gewichten ANN	Glorot initialisatie (uit uniforme verdeling $[-a, a]$ )
Bias ANN	0
<b>Data-augmentatie:</b>	
Zoom	[83.33,120] %, uniforme verdeling
Rotatie	$[-2,2]^\circ$ , 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

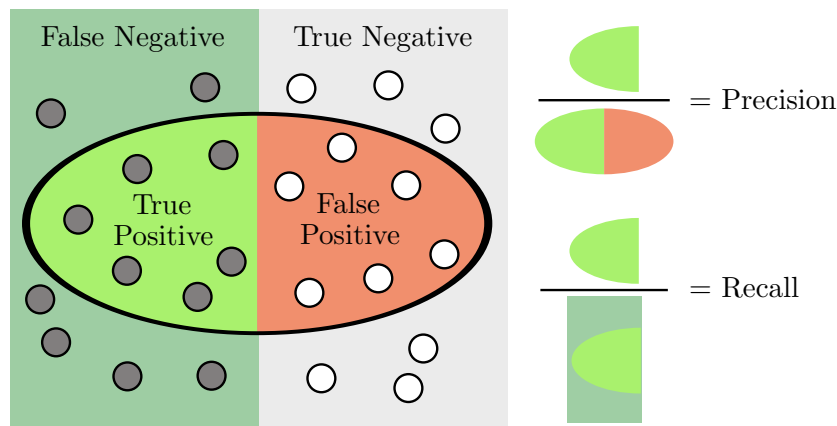
Na 250 iteraties, die we verder een epoch zullen noemen, wordt het model geëvalueerd met de validatieset. Het minimum van de validatiefout wordt bijgehouden en telkens we een lagere waarde bereiken worden de modelparameters opgeslaan. De training stopt wanneer er na een aantal epochs geen verbetering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

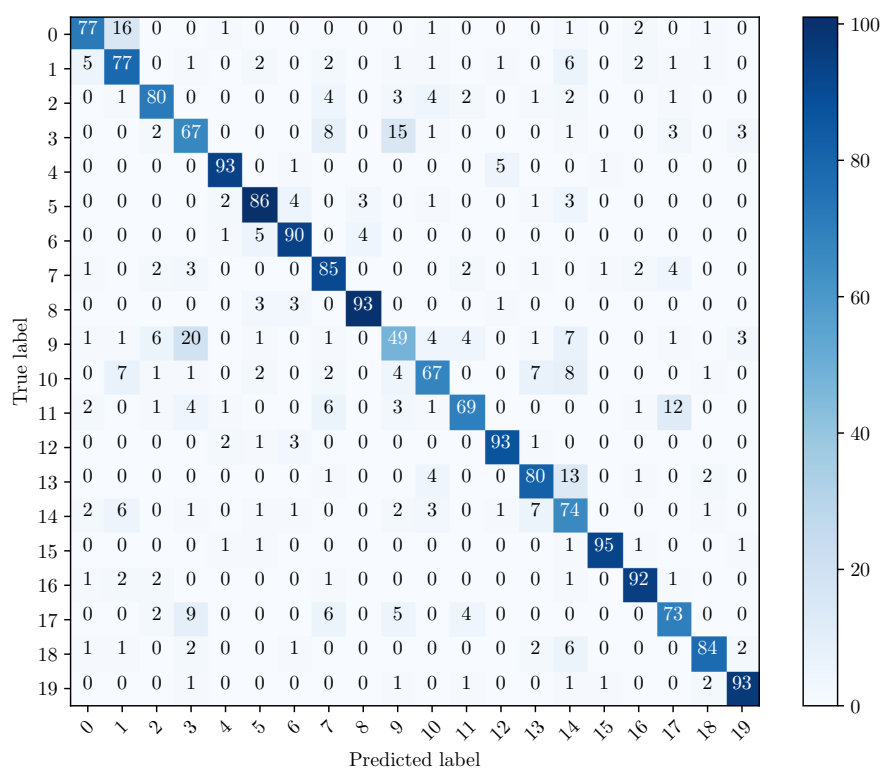
## 3.5 Een gebaar bijleren

### 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 is en genoeg geregulariseerd, 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 natuurlijk niet erg bruikbaar om een dynamisch uitbreidbaar herkenningssysteem te maken maar is wel een goede eerste test van het potentiële 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 (die meerdere voorbeelden van het nieuwe gebaar bevat). Deze bevat alle voorbeelden van het nieuwe gebaar zodat er goed kan gelet worden op overfitting.



**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 %	<b>15</b>	<b>96,88 %</b>	<b>94,90 %</b>
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.



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.

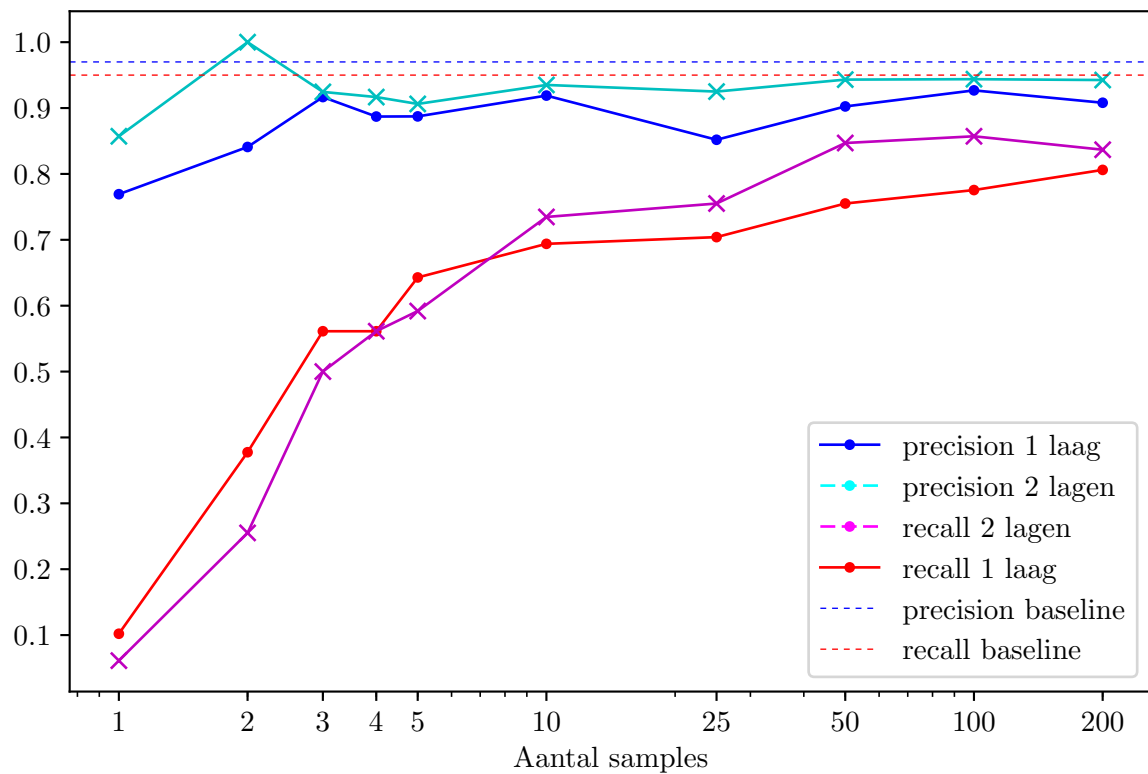
Eerst en vooral wordt het aantal samples gevarieerd. Het model wordt getraind met 200, 100, 50, 25, 10, 5, 4, 3, 2 voorbeelden en ten slotte wordt er 1 voorbeeld gebruikt voor een poging tot one-shot learning.

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 laag en de laatste verborgen laag hertraind, met 100 units: de dense100 laag (zie Figuur 3.4).

In figuur 3.7 zijn de resultaten van deze experimenten met het “softmax20” model te zien. Het bijgeleerde gebaar is 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 makkelijk 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 eventjes 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 stijkste 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 moet worden bij het hertrainen van de twee lagen ( $100 \cdot 800 + 100 \cdot 20$  ipv  $100 \cdot 20$ ) en 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 getraind op alle voorbeelden.

### 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 model *Softmax19+1* opgesteld, te zien in Figuur ??.

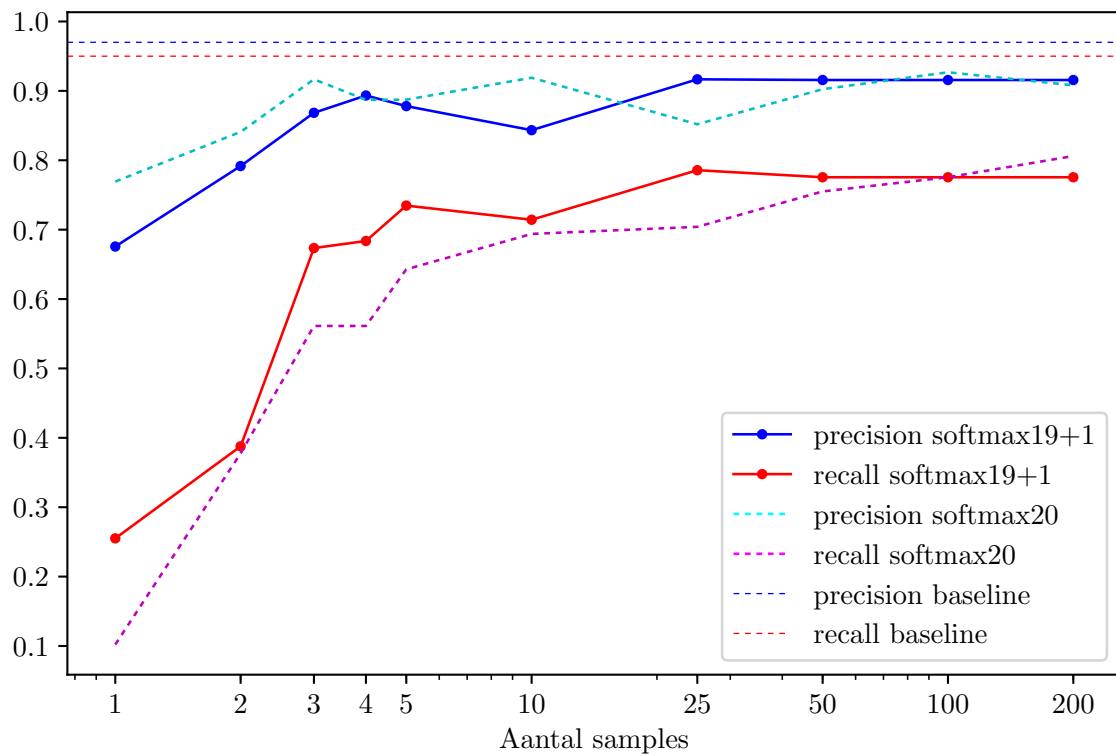
Hier wordt eerst een model met 19 softmax units getraind voor de classificatie van 19 gebaren. Behalve het verschil in aantal softmax units is de architectuur volledig gelijk aan die van het basismodel. Daarna wordt het model uitgebreid met een extra unit voor de classificatie van het nieuwe gebaar. 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 herkenningssysteem, het bijleren van een nieuw gebaar mag geen negatief effect hebben op de huidige prestatie van het model.

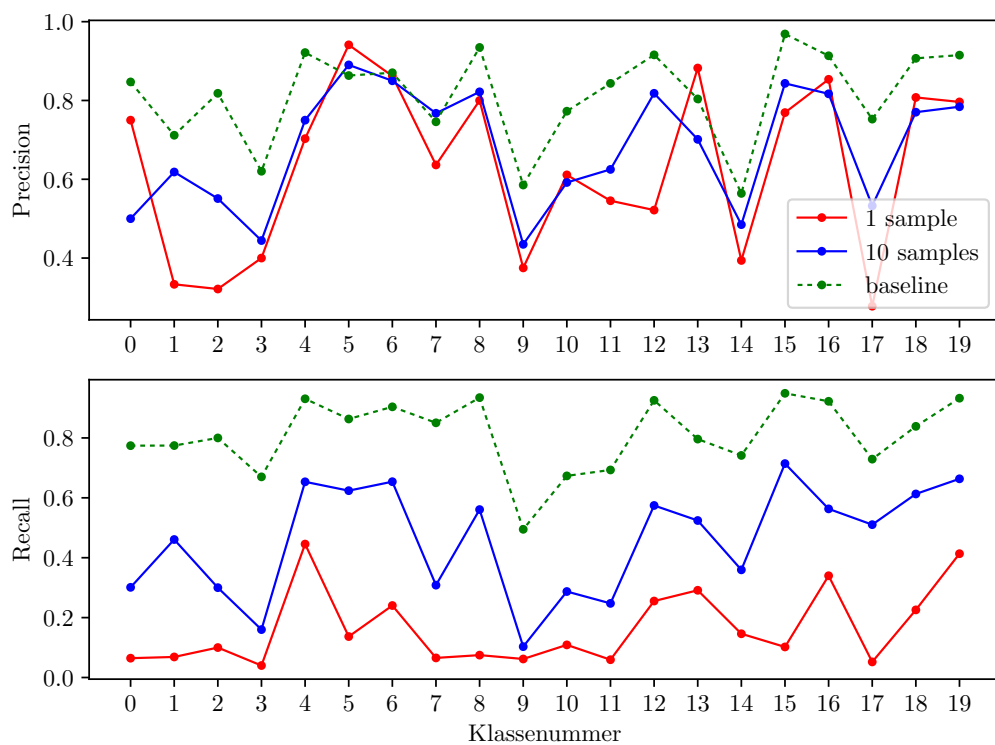
#### Invloed van het aantal samples

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.

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



**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.

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.

### **Invloed van het bijgeleerde gebaar**

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 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 er aan data-augmentatie gedaan met de parameters beschreven in Tabel 3.1. Er werden nieuwe test-opstellingen gemaakt zonder gebruik van data-augmentatie. Er was haast geen verschil tussen het model met en zonder augmentatie. Bij nader inzien waren de gekozen parameters erg conservatief en brachten ze weinig variatie in de data. Er werd geprobeerd om een meer drastische data-augmentatie toe te passen. Om met de grotere variatie om te kunnen gaan moest 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. Dit model bleek te underfitten: het was nog niet complex genoeg om de gaugmenteerde data te beschrijven. Om deze opstelling te optimaliseren diende er opnieuw een iteratief proces van hyperparameteroptimalisatie aangevat te worden. Hiervoor was er te weinig tijd. Het gebruik van data-augmentatie kan wel verder onderzocht kan worden 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.

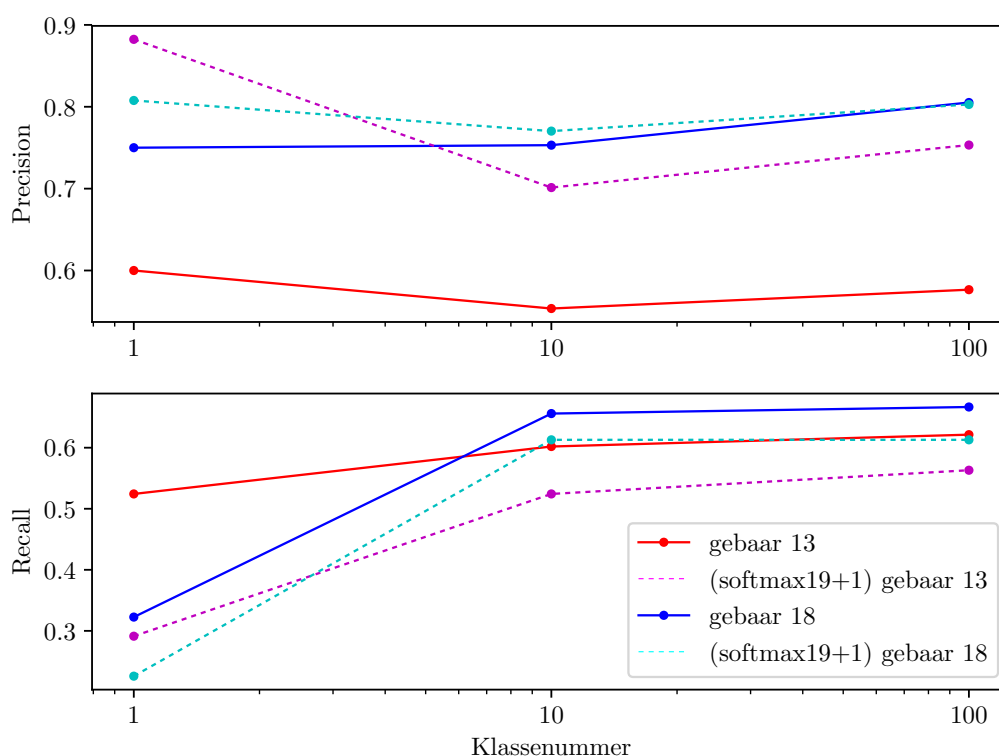
Het model leert een gebaar bij door de training van het nieuw toegevoegde unit en wordt daarna opnieuw uitgebreid met nog een extra unit voor het tweede gebaar. Ook deze unit wordt geconcateneerd zodat er uiteindelijk een softmax van 20 units is (18+1+1).

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.

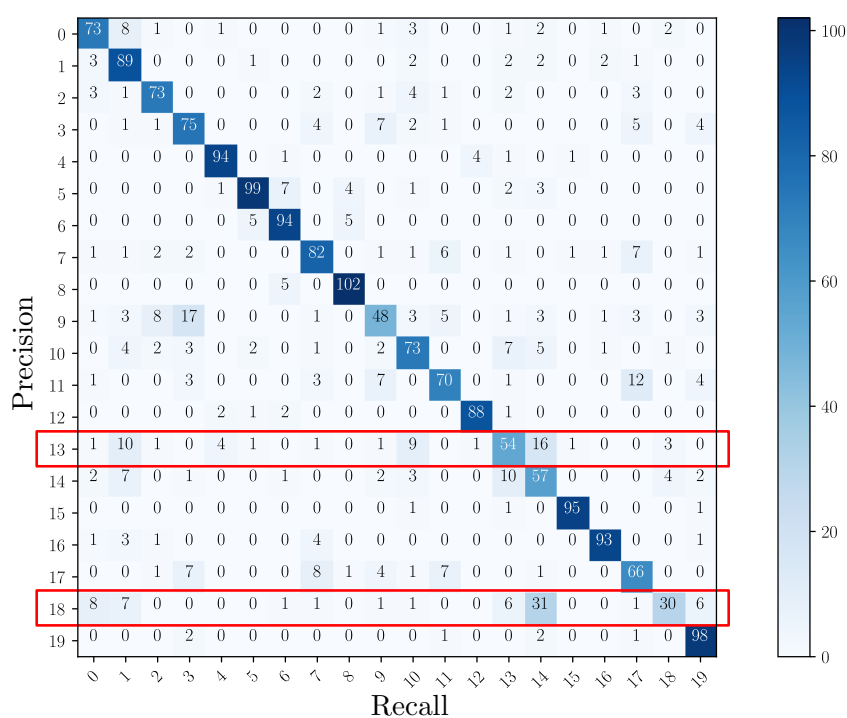
In Figuur 3.10 zijn de resultaten te zien van het bijleren van gebaar 13 en gebaar 18. Eerst wordt gebaar 13 bijgeleerd, daarna gebaar 18, telkens met even veel voorbeelden. De recall score van beide gebaren ligt hoger dan wanneer ze elk afzonderlijk worden bijgeleerd in het softmax19+1 model. De precision 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 van het bijleren uit een voorbeeld. Hierop is te zien dat er een kleine verwarring 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

### 4.2 Verder onderzoek



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

- [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

2.1	Schematische weergave van het opstellen van een predictief model met behulp van machine learning technieken . . . . .	11
2.2	Plot van trainings- en validatiefout ter illustratie van overfitting en de early stopping techniek. . . . .	11
2.3	Een ANN met een invoerlaag, twee verborgen lagen en een softmax uitvoerlaag samen met zijn bouwsteen, het artificieel neuron. . . . .	13
2.4	Weergave van de drie meest populaire activatiefuncties voor classificatie doeleinden . . . . .	14
2.5	Maximum pooling: uit elk vak in de invoer afbeelding wordt het maximum bepaald en overgenomen in de bijhorende pixel in het uitvoerbeeld. . . . .	15
3.1	De Italiaanse gebaren gebruikt in de “ChaLearn Looking At People Challenge 2014, Track 3: gesture spotting” . . . . .	18
3.2	De vier datastromen beschikbaar in de “ChaLearn Looking At People Challenge 2014, Track 3: gesture spotting” dataset . . . . .	19
3.3	De 12 frames gebruikt voor de invoer van het CNN. De frames hebben een resolutie van 64x64 pixels. . . . .	20
3.4	Het basismodel bestaande uit een drielagig CNN voor feature-extractie en een drielagig ANN (met softmax uitvoerlaag) voor classificatie . . . . .	21
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 . . . . .	24

---

3.6	Genormaliseerde confusion matrix van baseline voorspellingen van het basis-model, getraind op alle voorbeelden van alle gebaren . . . . .	25
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 getraind op alle voorbeelden. . . . .	27
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. . . . .	29
3.9	Barplots per gebaar van de precision en recall van de baseline en na het hertrainen met respectievelijk 1 en 10 samples. . . . .	29
3.10	Precision en recall na het bijleren van gebaar 13 en gebaar 18 in het <i>softmax18+1+1</i> model. Ter vergelijking is het resultaat van het <i>softmax19+1</i> model ook uitgetekend. . . . .	32
3.11	Genormaliseerde confusion matrix van het <i>softmax18+1+1</i> model na leren van gebaar 13 en gebaar 18 met een voorbeeld. . . . .	32