

Graph Neural Networks en TinyML voor robuuste cijferreconstructie en automatische uitlezing van analoge meters op resource-beperkte hardware

Lenka Piet (500854373)

Hogeschool van Amsterdam

Faculteit: Digitale Media en Creatieve Industrie

Amsterdam

lenka.piet@hva.nl

Samenvatting—Automatic Meter Reading (AMR) via beeldherkenning biedt een duurzaam en kostenefficiënt alternatief voor het handmatig uitlezen van analoge meters in industriële omgevingen. Echter, uitdagende omstandigheden zoals slechte verlichting, vervuiling, beschadigingen en gedeeltelijk verborgen cijfers hebben een negatieve invloed op de prestaties van bestaande AMR-systemen. Dit onderzoek richt zich op de ontwikkeling van een robuust en energiezuinig AMR-systeem dat geschikt is voor resource-beperkte hardware, zoals microcontrollers.

Door gebruik te maken van verschillende technieken, waaronder YOLOv11nano voor objectdetectie, PP-OCRv3/slim voor Optical Character Recognition (OCR) en Graph Neural Networks (GNNs) voor reconstructie van gedeeltelijk onleesbare cijfers, is een vernieuwde pipeline ontwikkeld. Specifieke optimalisaties zijn toegepast voor Tiny Machine Learning (TinyML), zodat het systeem kan draaien op energiezuinige hardware, met minimale rekenkracht en geheugen.

De prestaties van het systeem worden uitgebreid geëvalueerd op zowel component- als systeenniveau met realistische datasets en gerichte data-augmentatie om industriële omstandigheden te simuleren. Verwacht wordt dat de voorgestelde methode robuuster en nauwkeuriger zal zijn in vergelijking met bestaande methoden. Dit maakt het nieuwe AMR-systeem geschikt voor uitdagende toepassingen bij organisaties zoals Tata Steel, waarbij nauwkeurigheid, schaalbaarheid en energie-efficiëntie belangrijk is.

Index Terms—Automatische Meteruitlezing (AMR), TinyML, Graph Neural Networks (GNN), Optical Character recognition (OCR), YOLO, Image inpainting, Contourreconstructie, Industriële Computer Vision

I. INTRODUCTIE

In de industrie wordt het energieverbruik gemeten door analoge meters. Het uitlezen van deze meters gebeurt momenteel handmatig, wat het proces inefficiënt maakt. Aangezien nauwkeurige en continue monitoring bijdraagt aan het optimaliseren van het energieverbruik [1], biedt automatisering van het uitleesproces potentie voor verbetering.

Het geautomatiseerd uitlezen van meters, ook wel Automatic Meter Reading (AMR) genoemd, wordt in de literatuur voornamelijk geassocieerd met digitale of slimme meters [2]. Omdat het vervangen van bestaande meters door deze digitale en/of slimme meters onpraktisch en tijdrovend is [3], richt dit onderzoek zich op een alternatieve aanpak: beeldgebaseerde

uitezing van analoge meters met behulp van computer vision en deep learning.

Hoewel deep learning-methoden voor AMR al uitgebreid zijn onderzocht [4]–[7], blijven deze methodes gevoelig voor uitdagende omstandigheden. Wisselende lichtomstandigheden, vervuiling van de meter, beschadigingen en veranderende perspectieven kunnen leiden tot misclassificaties en foutieve uitlezingen. Daarnaast zijn deze modellen vaak rekenintensief, waardoor ze minder geschikt zijn voor toepassingen waarin beperkte rekenkracht beschikbaar is of energiezuinigheid belangrijk is.

Tiny Machine Learning (TinyML) biedt een veelbelovende oplossing voor deze beperkingen. TinyML maakt het mogelijk om deep learning-modellen uit te voeren op compacte, energiezuinige en betaalbare hardware zoals microcontrollers. Deze aanpak biedt daarmee een praktische en kostenefficiënte oplossing die eenvoudig binnen bestaande industriële omgevingen kan worden geïntegreerd.

A. Probleemstelling

Binnen industriële omgevingen, zoals bij Tata Steel, bevinden meters zich vaak op moeilijk bereikbare locaties zoals in putten, leidingen of in slecht verlichte ruimtes. Daarnaast is in sommige gevallen een stroomvoorziening niet beschikbaar, waardoor systemen op batterijen moeten werken. Hierdoor wordt energieverbruik een belangrijk aandachtspunt. Een laag energieverbruik verlengt de gebruiksduur van het systeem en voorkomt dat batterijen vaak vervangen moeten worden, wat in de praktijk vaak onpraktisch is.

Bestaande AMR-systemen blijken vaak onvoldoende robuust onder realistische industriële omstandigheden. Problemen zoals vervuiling, beschadiging, reflecties of slechte verlichting kunnen leiden tot gedeeltelijk onleesbare cijfers en dus foutieve uitlezingen. Daarnaast vereisen deze modellen veel rekenkracht en energie, wat ze minder geschikt maakt voor toepassingen op energiezuinige, compacte hardware [4]–[7]. TinyML kan hierbij de oplossing bieden, maar er is binnen het AMR-domein nog geen specifiek onderzoek naar gedaan.

B. Doel

Dit onderzoek heeft als doel een robuust en efficiënt beeldgebaseerd AMR-systeem te ontwikkelen op basis van TinyML, dat geschikt is in uitdagende industriële omgevingen zoals bij Tata Steel. De focus ligt op het verbeteren van de uitleesnauwkeurigheid onder uiteenlopende omstandigheden, waarbij het reconstrueren van gedeeltelijk onleesbare cijfers als mogelijke oplossingsrichting worden onderzocht.

Door gebruik te maken van TinyML kunnen machine learning-modellen worden uitgevoerd op compacte en energiezuinige hardware, zoals microcontrollers. Dit maakt het mogelijk om analoge meters automatisch uit te lezen, zelfs in situaties waar geen/beperkte stroomvoorziening aanwezig is. Dit onderzoek draagt daarmee bij aan de ontwikkeling van een betrouwbare, schaalbare, energiezuinige en kostenefficiënte oplossing voor AMR binnen een industriële omgeving.

C. Scope van het onderzoek

Dit onderzoek richt zich specifiek op het automatisch uitlezen van gecentreerde, vervilde cijfers op analoge meters. De uitdaging zit in het correct reconstrueren van cijfers die (deels) onleesbaar zijn door vervuiling, slijtage, lichtvariaties of beschadiging.

Het probleem van daaiende cijfers, waarbij een cijfer zich tussen twee standen bevind valt buiten de scope van dit onderzoek. In de praktijk kan dit namelijk beter worden opgelost met behulp van historische meetdata. Door deze tijdreeks te analyseren, kunnen tussenliggende waarden worden ingevuld. Op deze manier kunnen uitleefouten als gevolg van rotatie beperkt blijven.

Daarnaast zijn alternatieve methoden mogelijk, zoals het detecteren van twijfelgevallen waarbij twee cijfers even zichtbaar zijn (middenpositie), waarna het systeem automatisch een nieuwe foto neemt of de uitlezing tijdelijk uitstelt. Deze opties worden in dit onderzoek niet verder uitgewerkt, maar vormen mogelijke richtingen voor toekomstig werk.

D. Onderzoeks vragen

Bij dit onderzoek staat de volgende hoofdvraag centraal:
Hoe kan een TinyML-gebaseerd beeldherkenningsysteem worden ontwikkeld en geoptimaliseerd voor de automatische uitlezing van analoge meters in industriële omgevingen, waarbij vervormde of deels verborgen cijfers worden gereconstrueerd om een nauwkeurige meting te garanderen?

Om deze vraag te kunnen beantwoorden wordt er gekken naar de volgende deelvragen:

- Welke invloed hebben lichtomstandigheden, stof en oriëntatieverschillen op de nauwkeurigheid van de meter uitlezing?
- Hoe kan een deep learning model vervormde of deels verborgen cijfers effectief reconstrueren voor een betrouwbare uitlezing?
- Welke technieken en optimalisaties zijn nodig om een TinyML-model efficiënt te laten werken op energiezuinige embedded hardware?
- Welke methoden kunnen worden toegepast om een model met minimale data nauwkeurig een nieuw type meters aan te leren en hoeveel data is daarvoor minimaal nodig?

II. REQUIREMENTS

Tabel I geeft een overzicht van de requirements voor het AMR-systeem, die samen met Tata Steel zijn opgesteld. De requirements zijn voorzien van een toelichting en geprioriteerd volgens de MoSCoW-methode.

III. LITERATUURONDERZOEK

AMR via beeldherkenning en deep learning biedt een efficiënt alternatief voor handmatige uitlezing van analoge meters. In industriële omgevingen wordt dit echter lastig door factoren zoals slechte belichting, vervuiling en perspectieffvervorming.

Dit literatuuronderzoek bespreekt de inzet van deep learning en computer vision voor AMR, met aandacht voor robuustheid onder realistische industriële omstandigheden. Daarnaast wordt onderzocht hoe vervormde of gedeeltelijk onleesbare cijfers kunnen worden gereconstrueerd, en hoe TinyML AMR schaalbaar en toepasbaar maakt op energieuinige hardware.

A. Automatische Meteruitlezing (AMR)

Verschillende studies hebben deep learning-gebaseerde methoden ontwikkeld om de detectie en herkenning van meterstanden te verbeteren. Laroca et al. [4] pasten Fast-YOLO toe voor de detectie van meters en gebruikten CR-NET en CRNN voor de herkenning van cijfers, wat resulteerde in een nauwkeurigheid van 94,13 procent. Hoewel data-augmentatie de prestaties verbeterde, had het model veel trainingsdata nodig en presteerde het minder goed bij slechte lichtomstandigheden en roterende cijfers.

Een soortgelijke methode werd toegepast door Košćević en Subašić [5], die Faster R-CNN combineerden met aangepaste OCR-technieken. Hun methode bleek robuuster en de methode kon ook verschillende typen meters detecteren.

Echter, er ontstonden problemen in de prestaties bij extreme perspectieffveranderingen.

Laroca et al. [6] bouwden voort op eerdere methoden door Fast-YOLOv4-SmallObj te combineren met CDCC-NET voor hoekdetectie en Fast-OCR voor uitlezing. Deze aanpak behaalde een nauwkeurigheid van 99 procent bij het filteren van metingen met een lage betrouwbaarheid. Toch bleven uitdagingen zoals vervuiling, slechte belichting en roterende cijfers een terugkerend probleem. In plaats van deze problemen op te lossen, werden ze vooral vermeden.

Verder onderzochten Martinelli et al. [7] het gebruik van YOLOv5s voor de detectie en herkenning van cijfers op watermeters, waarbij data-augmentatie werd toepast om de nauwkeurigheid te verhogen. Ondanks goede prestaties op een dataset van 1500 afbeeldingen, bleef het model gevoelig voor lichtvariaties, rotatie en onduidelijke cijfers.

Deep learning-gebaseerde methoden hebben bewezen effectief te zijn voor AMR, vooral onder gecontroleerde omstandigheden. In de praktijk blijft de toepasbaarheid in industriële omgevingen echter beperkt. Variabele belichting, perspectieffveranderingen en beeldvervuiling leiden vaak tot fouten, zoals geïllustreerd in Figuur 1 van Laroca et al. [6]. Daarnaast vereisen deze modellen een hoge rekenkracht, wat ze minder geschikt maakt voor embedded systemen. Hierdoor blijft schaalbaarheid en kostenefficiëntie een uitdaging voor grootschalige industriële toepassingen.

ID	Omschrijving	Toelichting	Prioriteit
1	Volledige automatische verwerkingspipeline	Er is geen handmatige input nodig.	Must
2	Verwerkingstijd max 5 min	Van beeldopname tot uitlezing moet het systeem snel genoeg reageren binnen de tijd.	Must
3	Automatische herkenning uitleesgebied	Het systeem moet zelfstandig het relevante deel van de meter lokaliseren.	Must
4	Automatische cijferuitlezing	Het systeem moet automatisch de cijfers in het uitleesgebied uitlezen.	Must
5	Herstellen beschadigde cijfers	Bij vervaging, vervuiling of afdekking moet het systeem cijfers reconstrueren.	Must
6	Robuust bij variabele omstandigheden	Betrouwbare uitlezing bij lichtvariaties, vuil, rotatie en bedekking. De resultaten zijn vergelijkbaar met de prestaties in een gecontroleerde omgeving.	Must
7	Lokale verwerking	Geen afhankelijkheid van cloud of externe servers.	Must
8	Geoptimaliseerd voor beperkte hardware	Modellen geschikt voor beperkte rekenkracht en geheugen.	Must
9	Energieuinig systeem	Omdat meters zich vaak op lastig bereikbare plekken bevinden, is het belangrijk dat het systeem zo lang mogelijk op één batterij werkt.	Must
10	Maximale hardwarekosten €150	De totale kosten voor benodigde hardware, zoals microcontroller en camera, mogen maximaal €150 bedragen. Op deze manier blijft de oplossing schaalbaar en kostenefficiënt.	Must
11	Begrijpelijke en gestructureerde code	Code is duidelijk en goed gedocumenteerd voor overdracht.	Must
12	Leren van nieuwe meters met weinig data	Systeem moet zich kunnen aanpassen aan nieuwe meters met minimale voorbeelden.	Must
13	Foutmeldingen bij errors	Standaard foutcodes als uitlezing faalt (bijvoorbeeld door ruis of geen beeld).	Should
14	Draadloze verzending via LoRa	LoRa wordt gebruikt vanwege laag energieverbruik, groot bereik en betrouwbaarheid in industriële omgevingen. Tata Steel beschikt al over een LoRa-netwerk.	Should
15	Energieverbruik loggen en rapporteren	Het systeem moet het energieverbruik van zichzelf kunnen loggen, zodat duidelijk is hoe lang het systeem nog kan draaien op een batterij.	Could
16	Batterijniveau-waarschuwing	Wanneer de batterij onder een bepaalde drempel komt, stuurt het systeem automatisch een waarschuwing of foutmelding mee via LoRa.	Could

Tabel I: Overzicht van requirements voor AMR-systeem op resource-beperkte hardware.



Figuur 1: Voorbeelden van misclassificaties onder industriële omstandigheden. Boven: werklijke meterstanden. Onder: voorspellingen door het model. Overgenomen uit Laroca et al. [6].

B. Herstel van vervormde incomplete cijfers

Hoewel deep learning-technieken verbeteringen hebben gebracht binnen AMR, blijven situaties waarin cijfers vervormd raken of slechts gedeeltelijk zichtbaar zijn door vervuiling, beschadiging of reflecties een uitdaging [4]–[7]. In zulke gevallen kan het nuttig zijn om ontbrekende of beschadigde cijferdelen te reconstrueren. Een benadering binnen de computer vision is inpainting, waarbij een model ontbrekende beeldinformatie aanvult op basis van de visuele context. Dit maakt het mogelijk om zelfs bij gedeeltelijke obstructie of schade, de oorspronkelijke cijferweergave zo nauwkeurig mogelijk te herstellen, zodat de uitlezing nauwkeuriger en betrouwbaarder wordt.

1) Klassieke inpainting-methoden: Klassieke methoden voor image inpainting bestaan voornamelijk uit diffusie-gebaseerde en patch-gebaseerde technieken. Diffusie-gebaseerde methoden verspreiden pixelinformatie vanuit omliggende gebieden naar het beschadigde deel, maar zijn vooral geschikt voor kleine beschadigingen vanwege het gebrek aan semantisch begrip [8], [9]. Patch-gebaseerde methoden vullen ontbrekende regio's aan door gebruik te maken van vergelijkbare texturen binnen de afbeelding [10]. Hoewel deze methode krachtig is, vergen ze veel rekenkracht en werken ze minder goed bij grote of ongestructureerde beschadigingen [9].

De klassieke inpainting-methoden hebben een beperking bij complexe structuren, grotere gebieden en ongestructureerde beschadigingen. Deze inzichten vormen de basis voor verder onderzoek naar geavanceerdere methoden, zoals deep learning-gebaseerde methodes. Dankzij een dieper begrip van beeldinhoud en context kunnen deze technieken nauwkeurigere resultaten opleveren.

2) Deep learning inpainting-methodes: Deep learning-technieken hebben de beperkingen van de klassieke methoden aangepakt. Convolutionele Neurale Netwerken (CNNs) en Generative Adversarial Networks (GANs) zijn de twee meest gebruikte neurale netwerken als het gaat om deep learning-based inpainting methodes [11].

CNN-gebaseerde methoden zijn populair vanwege hun vermogen om ruimtelijke structuren in afbeeldingen te leren. De methodes kunnen worden ingedeeld in drie typen: single-shot, two-stage en progressieve methodes. Single-shot methoden voeren beschadigde afbeeldingen direct in één CNN-model, zoals bijvoorbeeld de Context Encoder-model van Pathak et al. [12], [13]. Hoewel deze methode snel werkt en relatief eenvoudig is, leveren ze soms onscherpere resultaten op bij complexe beschadigingen [12], [13]. Two-stage methoden hanteren een hiërarchische aanpak waarbij eerst een grove reconstructie wordt uitgevoerd, gevolgd door een verfijning. Dit levert betere resultaten, maar eist meer rekenkracht en is gevoelig voor fouten in de eerste fase [13], [14]. Progressieve methoden, zoals de LSTM-gebaseerde aanpak van Zhang et al. [11], [13], vullen ontbrekende gebieden iteratief in. Dit is effectief voor grotere gaten maar vereisen extra rekenkracht en kan bij het gebrek aan interne referentiepunten minder goed werken.

GAN-gebaseerde methoden maken gebruik van twee tegengestelde netwerken (generator en discriminator) om realistische reconstructies te creëren [11]. Verschillende varianten zijn:

- DCGAN: gebruikt convolutielagen voor stabielere trainingen en betere beeldrepresentaties [15].
- Wasserstein GAN (WGAN): verbetert stabiliteit en consistentie door gebruik te maken van de Wasserstein-

afstand, maar vereist meer rekenkracht [16].

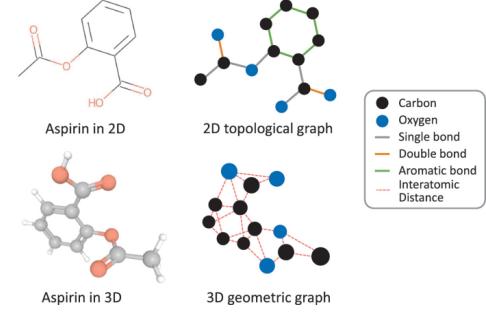
- Conditional GAN (CGAN): voegt contextuele informatie toe, waardoor specifieke controle mogelijk wordt over de gegenereerde invulling [17].
- PatchGAN: richt zich op lokale details door kleinere patches te analyseren, wat resulteert in realistischer gegenereerde beelden [18].

CNNs en GANs hebben ondanks vooruitgang in image inpainting nog steeds beperkingen bij specifieke toepassingen zoals cijferreconstructie. CNNs vertrouwen vooral op lokale patronen doordat ze met een vast raster werken, waardoor ze moeite hebben met nauwkeurige reconstructie van contouren [19]. GANs leveren dankzij hun generatieve aanpak vaak realistische resultaten. Echter, doordat ze geen expliciete kennis hebben van de onderliggende structuur, ontstaan er regelmatig kleine afwijkingen. Dit vormt een probleem bij taken waar nauwkeurigheid belangrijk is, zoals het uitlezen met behulp van een OCR-model (Optical Character Recognition) [14].

Daarnaast vragen GANs veel rekenkracht en grote hoeveelheden trainingsdata [20]. Hierdoor zijn ze minder geschikt voor implementatie in omgevingen met beperkte hardware, zoals TinyML-toepassingen op een Raspberry Pi met een maximale verwerkingstijd van slechts vijf minuten. Om toch nauwkeurige reconstructies te realiseren binnen deze beperkingen, is een lichtgewicht model nodig dat expliciet rekening houdt met de structurele eigenschappen. Graph Neural Networks (GNNs) bieden hiervoor een veelbelovende oplossing, doordat zij structurele relaties expliciet modelleren en daardoor mogelijk betere resultaten leveren met minder rekenkracht.

3) *Graph Neural Networks*: GNNs vormen een veelbelovend alternatief voor traditionele inpainting-methoden doordat ze afbeeldingen niet behandelen als een raster van losse pixels, maar als een gestructureerde graaf. Een graaf is een wiskundige structuur bestaande uit knopen (nodes) en randen (edges). Het wordt veel gebruikt om complexe relaties te modelleren bijvoorbeeld in sociale netwerken, transportmodellen of moleculaire structuren [21].

In de moleculaire chemie kunnen bijvoorbeeld atomen worden voorgesteld als knopen en chemische bindingen als randen, zie Figuur 2. Door eigenschappen van naburige knopen en verbindingen te analyseren, kunnen globale eigenschappen van het molecuul worden afgeleid [22]. Op vergelijkbare wijze kunnen bij cijferreconstructie de contouren van een cijfer worden opgenomen in de graafstructuur. Zo kan contextuele informatie over de vorm en opbouw van het cijfer worden vastgelegd en gebruikt worden door een GNN.



Figuur 2: Graaf voor GNN van moleculen [22].

Hoewel GNNs nog zelden worden toegepast bij image inpainting [13], laat een recente benadering genaamd GraphFill zien dat deze modellen effectief gebruik kunnen maken van structurele samenhang in beelden. GraphFill combineert een ruwe reconstructie via een GNN met een verfijningsnetwerk, wat leidt tot realistische resultaten. Een uitdaging binnen deze aanpak is het gebruik van superpixels, waarbij aangrenzende pixels met vergelijkbare eigenschappen worden gegroepeerd tot één segment. Deze techniek vermindert de rekencomplexiteit, maar kan ook leiden tot verlies van fijne details in de reconstructie. Daarnaast zorgt het gebruik van een verfijningsnetwerk voor een verhoogde rekenlast [23]. GNNs hebben ook hun waarde bewezen in gerelateerde domeinen zoals handschriftherkenning [24], waar het herkennen en behouden van structurele patronen belangrijk is.

C. Tiny Machine Learning

TinyML richt zich op het uitvoeren van machine learning (ML)-modellen op energiezuinige en beperkte hardware, zoals microcontrollers en edge-apparaten [25]. Dit is relevant omdat in de requirements staat dat het model lokaal moet draaien op een apparaat met beperkte rekenkracht, zoals een Raspberry Pi. TinyML maakt dit mogelijk en biedt daarbij voordelen zoals lage latentie, verbeterde privacy en minimaal energieverbruik, zonder afhankelijk te zijn van een cloudverbinding [25].

1) *Optimalisatietechnieken voor TinyML*: Voor inzetbaarheid op beperkte hardware worden ML-modellen geoptimaliseerd op geheugengebruik, rekenefficiëntie en energieverbruik [25]. Belangrijke optimalisatietechnieken zijn:

a) *Quantisatie*: Quantisatie verlaagt de numerieke precisie van netwerkparameters, vaak van 32-bit floating point naar 8-bits integers. Dit reduceert geheugen- en rekenbelasting aanzienlijk, meestal zonder significant verlies in nauwkeurigheid [25].

b) *Pruning*: Pruning verwijdert overbodige gewichten en verbindingen om netwerken kleiner en sneller te maken. Er bestaan diverse methoden:

- Magnitude-based pruning verwijdert gewichten met kleine absolute waarden.
- Structured pruning verwijdert hele lagen of neuronen.
- Unstructured pruning verwijdert individuele gewichten [25].

c) *Gewichtsdeling*: Gewichtsdeling beperkt geheugen en rekenbelasting door gewichten te groeperen en te delen. Om het verlies van nauwkeurigheid te beperken, wordt gewichtsdeling meestal toegepast na training of verfijnd met technieken zoals soft-weight sharing en parameter hashing [25].

d) *Neural Architecture Search (NAS)*: NAS automateert het ontwerp van efficiënte neurale netwerkarchitecturen voor hardware met beperkte rekenkracht. NAS omvat een zoekalgoritme, zoekruimte en evaluatiestrategie. Voorbeelden zijn SpArSe, dat efficiënte CNNs voor microcontrollers ontwikkelt [26] en TinyNAS, specifiek voor TinyML-toepassingen [27].

2) *Software voor TinyML*: Softwareframeworks zoals TensorFlow Lite for MCUs (Google) [28] en Embedded Learning Library (Microsoft) [29] maken het trainen en implementeren van ML-modellen op beperkte hardware mogelijk. Diverse tools en bibliotheken zorgen voor automatische omzetting naar geoptimaliseerde code voor directe integratie [25].

D. Few-shot learning

Het verzamelen van uitgebreide geannoteerde datasets voor elk nieuw type meter is vaak onpraktisch. Few-shot learning (FSL), geïnspireerd op de menselijke manier van leren, biedt een oplossing door modellen te trainen met minimale data. Centraal hierbij staat meta-learning, waarbij een model op meerdere gerelateerde taken wordt getraind om snel te kunnen aanpassen aan nieuwe taken [30]. Hiervoor worden verschillende technieken ingezet:

a) *Siamese en prototypische netwerken*: Siamese netwerken classificeren objecten door hun gelijkenis te berekenen in een feature space [31]. Prototypische netwerken maken een prototype per klasse aan en classificeren nieuwe voorbeelden op basis van afstand tot deze prototypes [32].

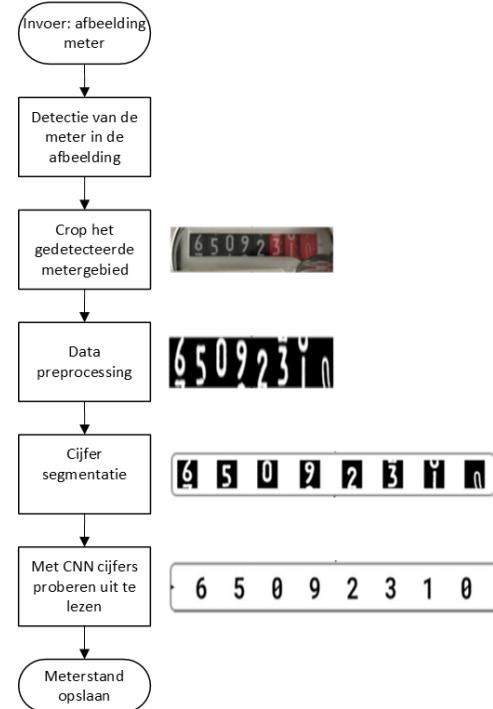
b) *Transfer learning*: Transfer learning gebruikt bestaande deep learning-modellen, die eerder op grote datasets zijn getraind. Hierbij worden alleen de laatste lagen aangepast of gefine-tuned met beperkte nieuwe data, waardoor het snel aan te passen is aan specifieke meters [30].

c) *Data augmentation en generatieve modellen*: Data augmentation voegt variaties toe aan beperkte datasets door technieken zoals roteren, spiegelen en toevoegen van ruis. Generatieve modellen zoals GANs en VAEs genereren synthetische data om robuustere modellen te trainen [30].

FSL met technieken zoals siamese en prototypische netwerken, transfer learning en data augmentation, maakt het mogelijk om nieuwe meters met minimale data nauwkeurig uit te lezen. Dit vermindert de noodzaak van uitgebreide datasets, verlaagt kosten en versnelt de implementatie van AMR.

IV. METHODE

Huidige AMR-systeem worden vaak ontwikkeld volgens een vaste procesflow, zoals weergegeven in Figuur 3. Echter, studies die deze aanpak hanteren ondervinden dat het systeem niet robuust genoeg is. Dit leidt tot fouten bij de uitlezing, vooral wanneer de beelden vervuild of onvolledig zijn [4]–[7].



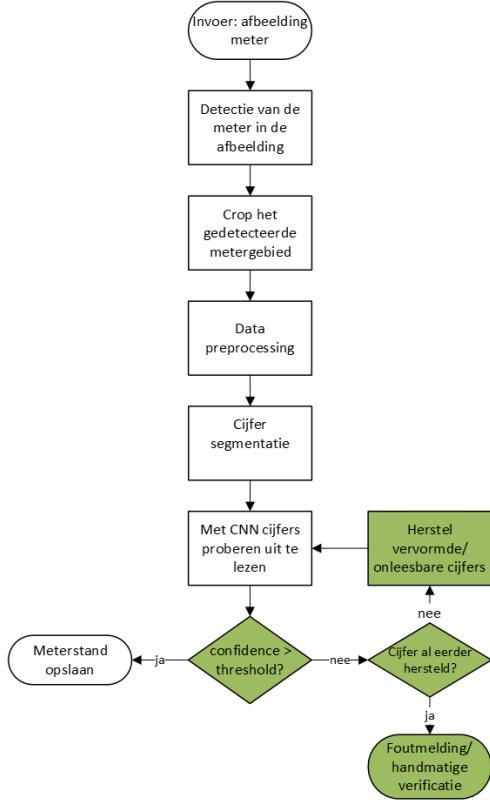
Figuur 3: Basis procesflow AMR.

Dit onderzoek introduceert een verbeterde en gestructureerde pipeline voor AMR-systeem, met als doel de robuustheid van het systeem te verbeteren. Dit wordt gedaan door een extra stap toe te voegen aan de bestaande workflow: het reconstrueren van cijfers wanneer het CNN deze niet correct kan uitlezen, bijvoorbeeld als gevolg van vervuiling of vervorming. De beslissing om tot reconstructie over te gaan is gebaseerd op de confidence score van het CNN-model. De geoptimaliseerde workflow is weergegeven in Figuur 4.

Naast het optimaliseren van de workflow wordt het AMR-systeem verder geoptimaliseerd voor TinyML, zodat het efficiënt kan draaien op een Raspberry Pi.

A. Data

In dit onderzoek zijn verschillende datasets gebruikt voor het trainen van drie afzonderlijke modellen: een detectiemodel (YOLO), een classificatiemodel (CNN) en een reconstructiemodel (GNN). Voor het detectiemodel is gebruikgemaakt van bestaande beelddata van analoge meters, waarbij bounding boxes zijn toegepast op het volledige cijferdisplay. Voor het CNN- en GNN-model is daarentegen gekozen voor volledig synthetisch gegenereerde cijfers, om maximale controle te hebben over variatie in vorm, dikte en vervorming.



Figuur 4: Procesflow met reconstructie van cijfers.

In de volgende secties worden de gebruikte datasets en hun verwerking per model besproken. Om consistentie en eerlijke evaluaties te garanderen, wordt bij alle datasets een vaste 60/20/20 split gehanteerd. Dit betekent dat 60% van de data wordt gebruikt voor training, 20% voor validatie tijdens het trainen en 20% wordt gebruikt als testset.

1) *Detectiemodel*: Voor de training van het detectiemodel is gebruikgemaakt van meerdere datasets. In eerste instantie werd de UFPR-AMR dataset geselecteerd als primaire dataset. Deze dataset bestaat uit 2000 afbeeldingen, genomen in een magazijn van de Energy Company of Paraná (Copel), een Braziliaans energiebedrijf [4]. Zie Figuur 5 voor voorbeeld foto's.



Figuur 5: Voorbeeld foto's van de UFPR-AMR dataset [4].



Figuur 6: Voorbeeld van te krappe bounding boxes in de UFPR-AMR dataset.

Na verdere beoordeling bleek echter dat de meegeleverde annotaties onvoldoende aansloten bij het doel van dit onderzoek, waarin het volledige cijferdisplay gedetecteerd moet worden. De bounding boxes in de UFPR-AMR dataset omlijken enkel het cijfergedeelte van de meter, wat resulteert in te krappe afbakening (zie Figuur 6).

Een model getraind op basis van deze annotaties presteerde dan ook niet goed. Hierbij werden de voorspelde bounding boxes veel te krap waardoor cijfers soms niet volledig waren. Aangezien de volledige contouren van cijfers belangrijk zijn voor dit onderzoek, is er gekozen om de dataset deels handmatig te annoteren, met correcte bounding boxes.

Aanvullend zijn twee extra datasets geselecteerd via Roboflow om meer variatie in de trainset te brengen:

- MeterRead Computer Vision Project [33]: deze dataset had oorspronkelijk 2561 afbeeldingen van zowel digitale als analoge meters. Aangezien dit onderzoek zich specifiek richt op analoge meters, worden alle digitale varianten verwijderd. Na het verwijderen van alle digitale varianten bleven er in totaal 474 afbeeldingen over.
- AMR-RAD-DATA Computer Vision Project [34]: heeft 545 bruikbare beelden met grotere diversiteit in type meters en omstandigheden.

Alle bounding boxes zijn handmatig gecontroleerd en indien nodig gecorrigeerd. Om het model robuuster te maken tegen false positives, zijn er ook 22 achtergrondafbeeldingen toegevoegd uit de ‘Wired 3-Pole MCB’ dataset [35], waarin géén meters zichtbaar zijn. Deze beelden zijn specifiek geselecteerd als negatieve voorbeelden, in lijn met aanbevelingen van Roboflow om ongeveer 0–10% achtergrondbeelden toe te voegen [36]. Daarbij is bewust gekozen voor meterkasten die visueel overeenkomt met de omgeving waarin meters zich vaak bevinden.

De uiteindelijke dataset voor het detectiemodel bestaat uit 1442 afbeeldingen met gecontroleerde bounding boxes en 22 achtergrondbeelden zonder annotaties.

2) *Synthetische data (voor CNN en GNN)*: Voor dit onderzoek is er bewust gekozen om het classificatie- en reconstructiemodel uitsluitend te trainen op synthetische data.

Synthetische data biedt hierbij drie voordelen: controle over visuele vormgeving, efficiëntie in grootschalige generatie en de mogelijkheid tot gerichte verstoring.

De clean dataset bestaat uit centraal geplaatste cijfers van 0 t/m 9, die gegenereerd zijn in diverse digitale fonts. Deze fonts zijn geselecteerd op basis van een visuele analyse van de UFPR-AMR, waarbij duidelijk werd dat cijfers in de praktijk sterk variëren in vorm. In Bijlage A is een overzicht te zien van deze visuele variatie. Hierbij is te zien dat sommige cijfers smal, dun of geometrisch strak zijn, terwijl andere dikgedrukt of afgerond lijken.

Op basis van deze variatie zijn meerdere fonts geselecteerd en voor elke zijn de cijfers 0 t/m 9 gegenereerd, zie Tabel II. Per font zijn verschillende diktevariante toegevoegd, zodat het model kan leren omgaan met variaties in dikte. In Tabel III zijn de verschillende variante toegelicht.

Font	Voorbeeldcijfers (0-9)
Avenir Next LT Pro Demi	1234567890
Microsoft YaHei UI	1234567890
Agency FB	1234567890
Bahnschrift	1234567890
Consolas	1234567890
Courier New	1234567890
Segoe UI	1234567890
Gill Sans MT	1234567890
Posterama	1234567890
Lucida Console	1234567890
Verdana	1234567890
Franklin Gothic Medium Cond	1234567890
Daytona	1234567890
Biome	1234567890
OCRB	1234567890
Aptos Display	1234567890

Tabel II: Geselecteerde fonts met voorbeeldcijfers

Om ook visueel een beeld te krijgen van de verschillende diktevarianten is er in Figuur 7 een overzicht te zien waarin de effecten van de morfologische bewerkingen op cijfers worden weergegeven.

3) *Cijferuitezing (CNN)*: Om het classificatiemodel (CNN) te trainen op invoer die representatief is voor industriële omstandigheden, is er gebruikgemaakt van gerichte data-augmentatie op de eerder beschreven synthetische cijfers. Er is gekozen om Albumentations-bibliotheek te gebruiken, omdat deze bibliotheek een groter aanbod heeft van beeldvervormingen die beter aansluiten bij realistische toepassingen. De augmentaties zijn gericht geselecteerd om realistische variatie te simuleren, zoals die voorkomt in industriële omgevingen.

Deze augmentaties worden niet primair ingezet om overfitting tegen te gaan, maar om het model te trainen op invoer die lijkt op realistische data. Tijdens het trainen worden de augmentaties willekeurig en automatisch toegepast, zodat het model leert omgaan met verschillende visuele verstoringen.

Variant	Bewerking	Structuurelement	Iteraties	Beschrijving
Dun	Erode	2x2 kernel	1	Versterkte erosie om een sterk verdunde cijferstructuur te verkrijgen.
Dun_h	Erode	1x2 kernel	2	Matige erosie, gericht op horizontale verunning over meerdere stappen.
Dun_v	Erode	2x1 kernel	2	Lichte erosie, gericht op verticale verunning.
Normaal	-	-	0	Geen morfologische wijziging; origineel beeld behouden.
Dik_v	Dilate	2x1 kernel	2	Lichte dilatatie, zorgt voor een subtile verdikking in verticale richting.
Dik_h	Dilate	1x2 kernel	2	Matige dilatatie, verdikking in horizontale richting.
Dik	Dilate	2x2 kernel	1	Sterke dilatatie die het cijfer symmetrisch verdikt.

Tabel III: Overzicht van morfologische bewerkingen voor diktevarianten.

Om inzicht te geven in het effect van individuele augmentaties, is een selectie gemaakt van vijf cijfers waarop telkens één specifieke bewerking is toegepast. De resultaten zijn weergegeven in Tabel IV, waarin duidelijk zichtbaar is hoe elke transformatie de vorm van een cijfer beïnvloedt.

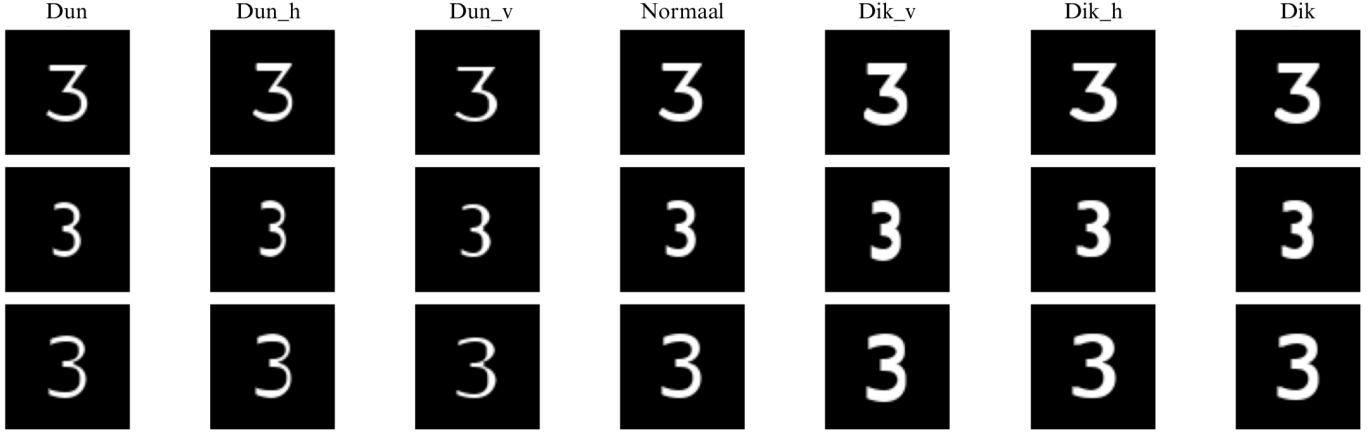
Daarnaast toont Figuur ?? vijftien voorbeelden van cijfers waarop meerdere augmentaties tegelijk zijn toegepast. Deze gegenereerde beelden zijn representatief voor de trainingsvoorbeelden waarmee het CNN wordt getraind. Deze aanpak zorgt ervoor dat het CNN robuust leert classificeren, ook onder verstoorende omstandigheden zoals die voorkomen in realistische industriële omgevingen.

4) *Reconstructie (GNN)*: Voor het trainen van het reconstructiemodel (GNN) is de augmentatie-methode aangepast, in vergelijking met die van het CNN. De augmentatie bij het GNN richt zich op het genereren van visueel nette en leesbare referentiebeelden (targets). Deze targets vormen het gewenste outputbeeld dat het model moet leren reconstrueren op basis van corrupte input.

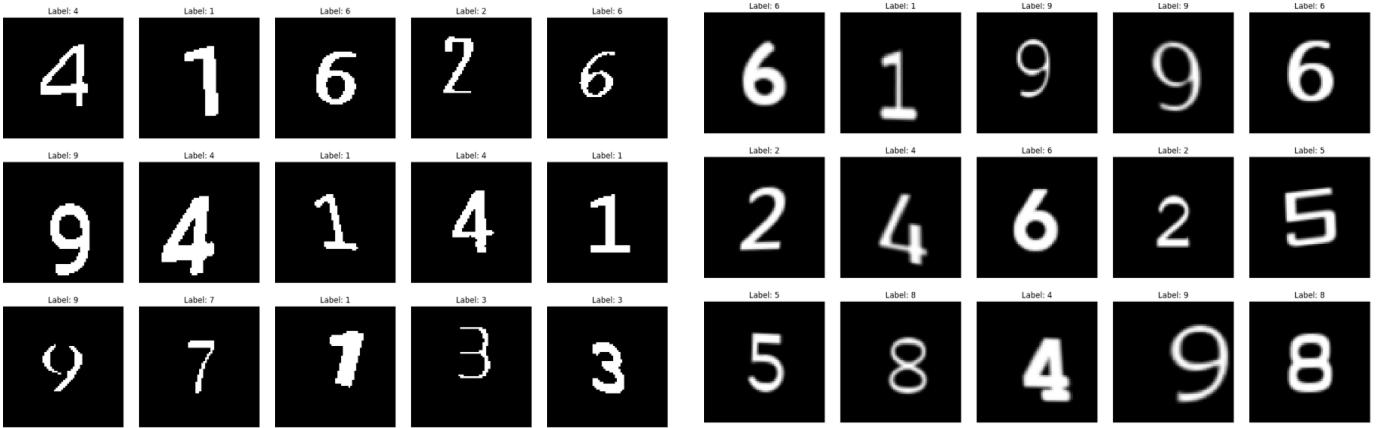
De targets zijn bewust niet volledig 'perfect', maar bevatten wel subtiële variatie. De gekozen augmentaties zijn gericht op het nabootsen van natuurlijke variaties, zonder de structuur van het cijfer te beschadigen of te verstören.

Om deze reconstructietaak te ondersteunen, zijn de augmentatieparameters strikter afgesteld:

- Shift: beperkt tot maximaal 5% van het beeldformaat, zodat cijfers gecentreerd blijven en niet buiten beeld raken.
- Schaal: variatie tussen 95–105%, om kleine schaalverschillen na te bootsen zonder vervorming van proporties.
- Shear: beperkt tot $\pm 5^\circ$, zodat de vorm behouden blijft.



Figuur 7: Enter Caption



Figuur 8: Enter Caption

Figuur 9: Enter Caption

- Geen ElasticTransform: Niet toegepast vanwege de onvoorspelbare lokale vervorming die kan optreden.
- Geen ruis aan de randen: uitgesloten om de scherpte van de cijfercontouren te behouden.

Figuur 9 toont een voorbeeld van vijftien gegenereerde targets met lichte variatie.

5) *Evaluatie:* Voor de evaluatie wordt er naast de al bestaande testset uit de eerder benoemde dataset, ook gebruik gemaakt van een eigen verzamelde dataset. Deze dataset bestaat uit 100 foto's van meters onder vijf verschillende omstandigheden: krassen, overbelichting, rotatie, schaduw, vlekken van stift en wazige foto's (Zie Figuur 10). Per situatie zijn er 20 foto's gemaakt met 5 verschillende meterstanden. Met deze dataset kan er geanalyseerd worden hoe het model presteert onder verschillende specifieke uitdagende omstandigheden. De foto's zijn gemaakt met een Raspberry Pi Cam 2 om dezelfde kwaliteit foto's te gebruiken die later ook in de praktijk worden gebruikt.

B. Metric

De prestaties van de verschillende onderdelen van het systeem worden geëvalueerd met passende metrics. In sommige

gevallen is alleen een metric niet voldoende en is visuele controle ook belangrijk, bijvoorbeeld bij reconstructie.

1) *Detectiemodel:* Voor het detectiemodel wordt gebruik gemaakt van de Average Precision (AP), een standaardmaat in objectdetectie die precision als recall combineert. Er is specifiek gekozen voor AP@0.5:0.95, waarbij over meerdere IoU-drempels het gemiddelde wordt genomen. Deze methode is strenger dan AP@0.5 en geeft beter weer of het model ook echt nauwkeurig lokaliseer [37].

2) *Classificatiemodel:* Het classificatiemodel wordt geëvalueerd met accuracy. Deze metric geeft het percentage correct voorspelde cijfers ten opzichte van het totaal. Het gaat hier om één juiste cijferklasse per voorspelling, wat accuracy een logische metric maakt.

3) *Reconstructiemodel (GNN):* Voor het reconstructiemodel wordt de binary F1-score gebruikt, waarin precisie en recall gecombineerd worden. Aparte scores voor precision en recall blijven echter wel belangrijk, omdat een model met hoge recall maar lage precision alsnog onbruikbare resultaten kan geven. Visuele evaluatie van de reconstructies is dus ook belangrijk. Zo kan een verbinding die eigenlijk fout is, niet erg zijn, zolang de globale cijferstructuur nog klopt. Een 3 die wordt

Augmentatie	Realistische simulatie	Gekozen waarde	Voorbeeldweergave
Rotatie	Kleine afwijkingen blijven mogelijk over na preprocessing en segmentatie. Deze augmentatie verhoogt robuustheid voor kleine hoeken die toch over blijven.	A.Rotate(limit=-5) → roteert beeld met een willekeurige hoek tussen -5° en +5°. Dekt restafwijkingen zonder onnatuurlijke vervorming.	2 7 6 7 4
Zoom (in/uit)	Verschillende typen meters tonen cijfers in iets andere groottes. Deze schaalvariatie wordt gesimuleerd door in- of uitzoomen.	A.Affine(scale=(0.85, 1.15)) → schaalt het cijfer tussen 85% en 115% rond het centrum. Simuleert variatie in cijfergrootte.	2 7 6 7 4
Shear (x en y)	Perspectivische vervorming door scheve cameraposities of hoekige displays.	A.Affine(shear={"x": (-10, 10), "y": (-10, 10)}) → vervormt het beeld diagonaal tot ±10°. Bootst hoekige opnames na.	2 7 6 7 4
Shifting (horizontaal/verticaal)	De locatie van cijfers kan variëren binnen het display, afhankelijk van het metertype of de geprinte versie. Shifting vergroot locatie-onafhankelijkheid.	A.ShiftScaleRotate(shift_limit=0.15, scale_limit=0, rotate_limit=0) → verplaatst het cijfer max 15% van het beeldformaat.	2 7 6 7 4
Perspectiefvervorming	Cijfers staan op een draaiende cilinder, waardoor kromming en hoekvervorming optreedt afhankelijk van rotatiepositie. Simuleert de projectie op een gebogen oppervlak.	A.Perspective(scale=(0.05, 0.1), keep_size=True) → past een perspectiefprojectie toe die kromming op een cilinder benadert.	2 7 6 7 4
Elastische vervorming	Simuleert lokale vervormingen in cijfers door kromme displays of fouten van binair segmentatie. Cijfers kunnen licht vervormd raken door beeldcompressie of slechte contourdetectie.	A.ElasticTransform(alpha=10, sigma=2, alpha_affine=2) → introduceert subtiele lokale vervorming in contouren.	2 7 6 7 4
Ruis toevoegen	Simuleert onnauwkeurigheden in de binair segmentatie van cijfers, zoals kleine rafelranden, onregelmatige pixelovergangen of lichte vervaging. Het model leert hiermee om te gaan met	$\sigma = 0.02$ (Gaussian noise), voldoende om binair contouren subtiel te verstören.	2 7 6 7 4

Tabel IV: Overzicht van augmentatietechnieken en hun parameters.

dichtgemaakt tot een 9 is wel een probleem, maar een extra streepje bij de 5 wat het cijfer langer maakt is niet erg.

Naast de F1-score wordt bij het eindmodel ook de Intersection over Union (IoU) berekend tussen de originele en gereconstrueerde contouren. Omdat het om binair vormen gaat, geeft de IoU duidelijk aan of het model de contouren van het cijfer echt correct reconstrueert [38].

4) *End-to-end uitleesnauwkeurigheid:* De uiteindelijke evaluatie betreft de volledige pipeline. Hierbij wordt gekeken naar hoeveel meterstanden het systeem volledig correct uitleest. Ook hier wordt gebruikgemaakt van accuracy. Deze score laat zien hoe goed het totale systeem presteert in een realistische situatie.

C. Detectiemodel

De eerste stap binnen het AMR-systeem is het detecteren van de meter en het uitleesgebied. Hiervoor wordt een objectdetectiemodel gebruikt. Binnen dit domein zijn diverse state-of-the-art-modellen ontwikkeld [39], waarbij de YOLO-architectuur (You Only Look Once) [40] een van de meest populaire is vanwege zijn snelheid en nauwkeurigheid [41].

Ook binnen AMR-toepassingen is YOLO een populaire keuze. Verschillende studies laten zien dat YOLO effectief kan worden ingezet bij het herkennen van displays op meters [4]–[7]. Recentere literatuur laat zien dat YOLOv8 effectief ingezet wordt voor het detecteren van meters [42]. Inmiddels

zijn er nieuwere versies beschikbaar met verbeterde prestaties, zoals YOLOv11. Bij deze versie is ook de rekenkrachtverbruik geoptimaliseerd in vergelijking met YOLOv8 [43].

1) *Trainingsproces:* In dit onderzoek wordt er gekozen voor YOLOv11nano, een lichtere variant van YOLOv11. Deze keuze is gebaseerd op de vereisten van TinyML-toepassingen, waarbij het model zo efficiënt en compact mogelijk moet zijn om inzetbaar te blijven op een Raspberry Pi. Hoewel inmiddels YOLOv12 is uitgebracht (februari 2025) [44], is er nog onvoldoende bekend over de stabiliteit en prestaties van dit model. Daarom is ervoor gekozen om vast te houden aan YOLOv11 als betrouwbare en bewezen basis.

Voor de training van YOLOv11nano is er gebruik gemaakt eerder benoemde samengestelde dataset. Het pre-trained YOLOv11nano model van Ultralitics is ingeladen en werd getraind gedurende maximaal 500 epochs. Hierbij is gebruikgemaakt van *early stopping* met een patience van 30 om overfitting te voorkomen.

Op de traingsdata is ook data-augmentatie toegepast. Het toepassen van data-augmentatie tijdens het trainen van een detectiemodel leidt vrijwel altijd tot een verbeterde performance, doordat het model leert omgaan met uiteenlopende visuele omstandigheden [45]. De toegepaste augmentaties zijn weergegeven in Tabel V.

Het model convergeerde na ongeveer 200 epochs, waarbij *early stopping* werd geactiveerd. Dit model is geselecteerd



Figuur 10: Voorbeelden van eigen verzamelde data. Van boven naar beneden: (1) krassen, (2) overbelichting, (3) rotatie, (4) schaduw, (5) stift rood en zwart, (6) stift groen en geel, (7) wazig.

als definitieve versie van het detectiemodel, aangezien de prestaties voldeden aan de verwachtingen. Een uitgebreide analyse van de resultaten is te vinden in de sectie Resultaten.

D. Preprocessing en segmentatie

Voor een betrouwbare uitlezing van de cijfers is een nauwkeurige preprocessing- en segmentatiestap belangrijk. De kwaliteit en nauwkeurigheid van deze bewerkingen heeft direct invloed op de classificatie later in de pipeline. Omdat de beelden in de dataset veel variaties hebben, zoals schaduw, rotatie, vlekken en slechte belichting, is er gekozen voor een methode die verschillende technieken combineert om de cijfers zo schoon en bruikbaar mogelijk te maken.

De opbouw van de preprocess-pipeline is gebaseerd op een methode die ook in de literatuur wordt toegepast bij kentekenplaat herkenning [46]. In de praktijk bleek deze aanpak echter niet voldoende om alle voorkomende verstoringen voldoende te corrigeren. Daarom zijn op basis van visuele beoordelingen verschillende technieken toegevoegd of aangepast. Denk hierbij aan het specifiek verwijderen van displayranden en een geavanceerdere binarisatie-techniek. Door deze aanpassingen sluit de pipeline beter aan op de eigenschappen van de data en worden de cijfers consistenten en schoner doorgegeven

Augmentatie	Toelichting
Degrees = 5	Er wordt tot ± 5 graden rotatie toegepast om kleine afwijkingen in camerahoek te simuleren.
Shear = 3	Er wordt een lichte shear-vervorming toegepast tot ± 3 graden, om perspectiefvervorming te simuleren.
flplr = 0	Horizontale spiegeling is expliciet uitgeschakeld. Dit is belangrijk omdat displays een vaste oriëntatie hebben en spiegeling leidt tot onrealistische voorbeelden. In de standaardinstellingen van Ultralytics is deze augmentatie automatisch geactiveerd, maar dat is hier ongewenst.
hsv_h = 0.003, hsv_s = 0.1, hsv_v = 0.1	Er worden kleine variaties in kleurtint, verzagding en helderheid toegepast. Deze augmentaties simuleren veranderingen in kleur en verlichting, zonder de herkenbaarheid van het display te beïnvloeden.
Mosaic = 0	Mosaic-augmentatie, waarbij meerdere afbeeldingen tot één trainingsvoorbeeld worden gecombineerd, is uitgeschakeld. Voor deze toepassing, waarbij telkens slechts één meter aanwezig is, is dit overbodig. Samenvoegen zou leiden tot onnatuurlijke voorbeelden die niet voorkomen in de praktijk.

Tabel V: Toegepaste data-augmentatie op traindata voor YOLO

aan het classificatiemodel. De uiteindelijke preprocessing-stappen worden hieronder toegelicht. Om ook visueel een goed beeld van het proces te geven, is in Figuur 11 per stap een tussenresultaat weergegeven.

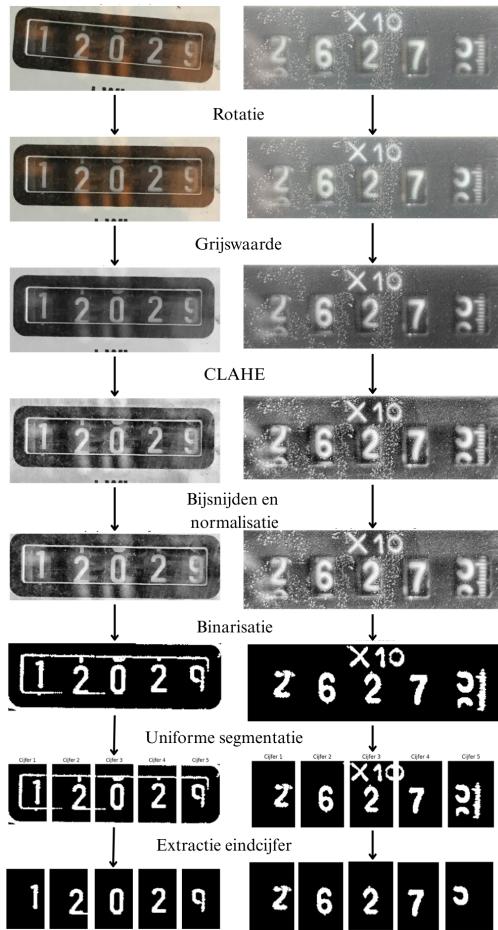
1) *Croppen en rotatiecorrectie*: Nadat het display is gedetecteerd met het YOLO-model, wordt eerst de bounding box gebruikt om het display uit de volledige afbeelding te croppen. Omdat sommige displays niet altijd recht op de foto staan, is een rotatiecorrectie nodig. Hiervoor worden eerst de randen gedetecteerd met Canny edge detection. Vervolgens wordt met Hough Line Transform gezocht naar rechte lijnen om de gemiddelde rotatiehoek te schatten. Deze wordt toegepast met een affine transformatie, zodat het display netjes rechtgezet word.

2) *Contrast verbetering en trimming*: Het beeld wordt daarna omgezet naar grijswaarde, waarbij er vervolgens CLAHE wordt toegepast om het lokale contrast te verbeteren. Hierna wordt de functie *smart_trim* gebruikt om witte randen rondom het display te verwijderen. Deze functie analyseert de buitenste rijen en kolommen van de afbeelding en bepaalt per rand of deze grotendeels uit witte pixel bestaat (minimaal 90% witte pixels).

Om te voorkomen dat de functie stopt bij kleine onderbrekingen in de rand (bijvoorbeeld door een pijl of vlek, waardoor niet meer dan 90% van de pixel in die rij/kolom wit zijn), is er een lookahead van vijf lijnen toegepast. Alleen wanneer ook de volgende lijnen niet wit zijn stopt het trimmen. Deze stap voorkomt dat er teveel foutieve structuren worden mee genomen bij de binarisatie.

3) *Normalisatie en binarisatie*: Na het bijnijden wordt de afbeelding in hoogte genormaliseerd, zodat alle beelden een consistente grootte hebben. Dit maakt de geselecteerde drempelwaarden en latere verwerkingen nauwkeuriger.

Voor binarisatie is gekozen voor een methode gebaseerd op 'two-stage adaptive thresholding met Sauvola' [47], omdat



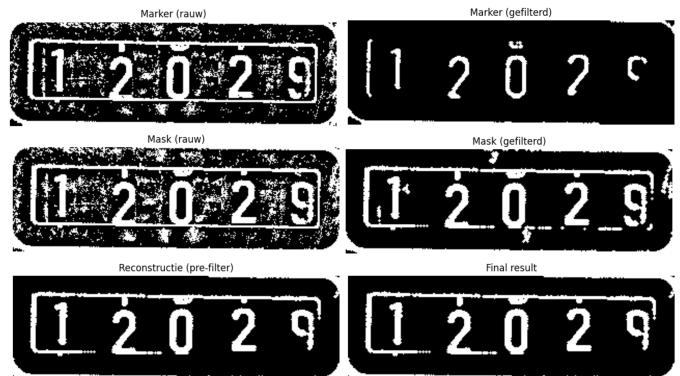
Figuur 11: Preprocessing-pipeline.

deze methode beter omgaat met variaties in belichting en ruis dan bijvoorbeeld Otsu of standaard thresholding. In plaats van één enkel threshold worden twee binaire versies van het beeld gemaakt:

- Een strikte drempel (lage k -waarde, $k = -0,1$) levert een beeld op met alleen de meest duidelijke cijferstructuren, ook wel de marker genoemd.
- Een zachtere drempel (hogere $k = 0,5$) levert het masker, dat ook zwakkere of deels vervaagde elementen meeneemt, maar ook meer ruis.

Met behulp van morfologische reconstructies wordt het markerbeeld uitgebreid binnen de grenzen van het masker. Na deze reconstructie worden aanvullende filterstappen toegepast, zoals het verwijderen van kleine objecten en morfologische openingen. Deze parameters zijn afgestemd op de dataset: groot genoeg om ruis te verwijderen, maar niet te sterk dat smalle cijfers zoals de '1' en '7' verloren gaan.

Een gedetailleerde beschrijving van deze binarisiatiemethode, gemaakte keuzes per stap en gebruikte parameters is toegelicht in Bijlage B. Voor een visuele weergave van de tussenstappen is in Figuur 12 het verloop van de marker, masker en eindresultaat weergegeven.



Figuur 12: Binarisatie stappen marker, masker en eindresultaat

4) *Segmentatie per cijfer*: Als het binaire beeld gemaakt is, wordt het display opgedeeld in gelijke segmenten. Elk segment komt overeen met één cijferpositie. De uniforme segmentatie werkt goed omdat de displays eerst zijn genormaliseerd, waardoor de cijfers netjes op vaste plekken staan.

5) *Opschoning en contourselectie*: Binnen elk segment wordt eerst ruis weggehaald met een horizontale lijn-filter (zoals displayranden). Daarna wordt een standaard morfologische opening toegepast om kleine vlekken en losse pixels te verwijderen.

Contouren worden vervolgens gedetecteerd met *OpenCV* en gefilterd op basis van de volgende criteria:

- Verticale positie: contouren die te hoog liggen in het segment worden genegeerd. Dit voorkomt dat ruis aan de bovenrand (zoals displayranden) onterecht als hoofdcomponent worden geselecteerd.
- Vormkenmerken: alleen contouren met een aspect ratio tussen 0,1 en 3,0 een een extend groter dan 0,2 worden behouden [48]. Een te lage aspect ratio duidt vaak op smalle verticale lijnen, terwijl een te hoge waarde kan wijzen op horizontale strepen. De extent geeft aan hoeveel van de bouding box daadwerkelijk gevuld is. Hiermee worden contouren uitgesloten die wel een grote omhullende bounding box hebben, maar aan de lege kant zijn, zoals ruis.
- Fall-backstrategie: als geen enkel contour aan alle eisen voldoet, wordt de grootste beschikbare contour alsnog geselecteerd.

Na filtering wordt binnen elk segment het grootste contour geselecteerd als eindcomponent van het cijfer. Het eindresultaat is een binaire afbeelding waarin per segment één component overblijft. Elk van deze segmenten is het cijferbeeld voor verdere verwerking in de pipeline.

E. Cijfer classificatie

Een nauwkeurig classificatiemodel is belangrijk voor een effectief AMR-systeem, omdat fouten in cijferclassificatie direct leiden tot foutieve uitlezingen. Hoewel PP-OCRv3 volgens het onderzoek van Nguyen et al. [42] de hoogste nauwkeurigheid behalen onder zes vergeleken OCR-modellen in een AMR-context, is voor dit onderzoek bewust gekozen om een eigen

CNN te trainen. PP-OCRv3 is een framework dat is ontworpen voor algemene tekstherkenning en bevat meerdere modules [49], die overbodig zijn voor AMR-toepassing waarbij cijfers netjes gesegmenteerd zijn.

Daarnaast heeft het volledige PP-OCRv3-model ongeveer 17 miljoen parameters en de slim variant PP-OCRv3-slim heeft er zelfs nog 1,1 miljoen [49], wat relatief groot is voor de toepassing binnen TinyML. In dit onderzoek is daarom gekozen voor een compacter, zelf ontworpen CNN-model. Deze aanpak zorgt voor eenvoud, volledig controle over de architectuur, transparantie en is beter af te stemmen op resource beperkte hardware.

1) *Trainingsopzet*: Als startmodel is gekozen voor een eenvoudig CCN-model, zoals beschreven in de Keras-documentatie. Dit model wordt als startmodel gebruikt vanwege zijn eenvoud en bewezen effectiviteit bij het classificeren van handgeschreven cijfers uit de grootschalige MNIST-dataset [50].

In tegenstelling tot de MNIST-dataset die uit tienduizenden afbeeldingen bestaat, is de dataset die in dit onderzoek wordt gebruikt aanzienlijk kleiner. Dit vergroot het risico op overfitting en beperkt de generaliseerbaarheid van het model. Om hiermee rekening te houden is gekozen voor K-Fold cross-validation. Hierbij wordt het model meerdere keren getraind en gevalideerd op wisselende subsets van de data. Deze methode zorgt voor een efficiënter gebruik van de volledige data en zorgt voor een robuustere inschatting van de modelprestaties dan enkel op een vaste validatieset [51].

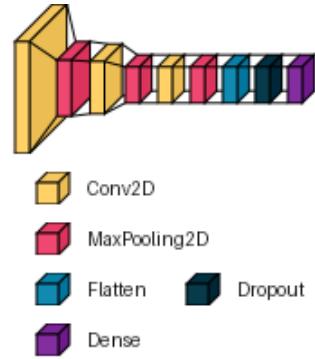
Zoals eerder besproken in het hoofdstuk Synthetische data, is data-augmentatie niet specifiek bedoeld als regularisatietechniek, maar vooral om een realistisch beeld te creëren. Deze augmentaties zijn direct geïntegreerd in het trainingsproces van het CNN-model.

Tijdens de training wordt een hybride aanpak toegepast, de trainingsdata wordt real-time gegaugmenteerd, zodat het model bij elke epoch nieuwe variante van de afbeeldingen krijgt. Voor de validatieset is vooraf een vaste, gegaugmenteerde set gegenereerd die tijdens het trainen niet veranderd. Hierdoor kan de evaluatie consistent worden uitgevoerd, zonder risico op enige datalekken of vertekeningen van de validatiescore.

Om te voorkomen dat dit model ook te lang door traint, is er opnieuw gebruik gemaakt van *early stopping*. Hoewel de training was ingesteld op maximaal 200 epochs, stabiliseerde de prestaties al eerder en werd de *early stopping* geactiveerd. In de trainingsresultaten is te zien dat het model niet overfit, omdat de prestaties op de validatieset beter zijn dan die op de trainset (zie Figuur 1C).

Aangezien het model underfit is, is de modelcomplexiteit verhoogt door een extra convolutie laag toe te voegen. Dit model behaalde nu maximale resultaten op de validatiedata waarbij er bij elke fold een acc wordt behaald van 100 procent (zie Figuur 4C). Ook waren er geen signalen van underfitting meer (zie Figuur 3C).

Op basis van deze resultaten is gekozen om deze architectuur te gebruiken voor het eindmodel. De architectuur is weergegeven in Figuur 13. Voor de definitieve training is deze



Figuur 13: Architectuur CNN eindmodel.

architectuur gebruikt om het eindmodel te trainen op de train- en validatieset bij elkaar, zodat de beschikbare data optimaal wordt benut voor de training.

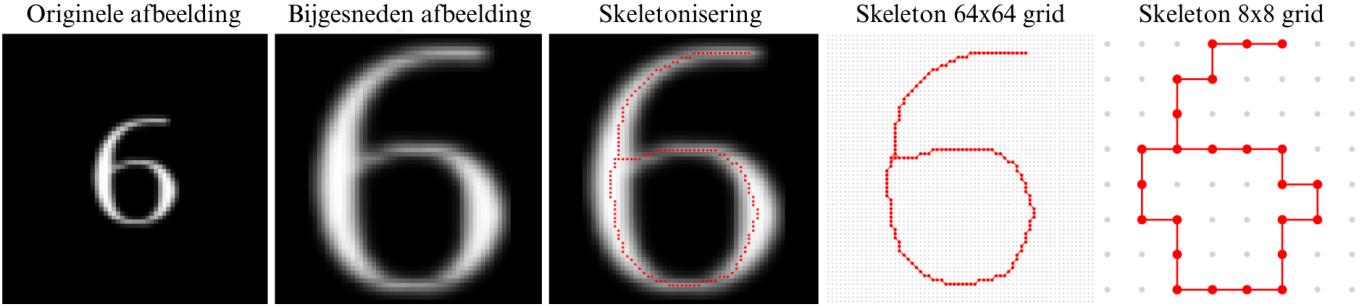
2) *Threshold bepaling*: Het selecteren van een juiste confidence-threshold is belangrijk binnen de pipeline. Een te lage threshold verhoogt de kans op foutieve classificaties, wat direct leidt tot een verkeerde meteruitlezing. Daar in tegen zorgt een te hoge threshold ervoor dat correcte voorspellingen onnodig worden doorgestuurd naar het reconstructiemodel, wat extra rekenkracht vereist. Een goede threshold is daarom belangrijk om een balans te vinden tussen classificatirouwkeurigheid en systeemefficiëntie.

Aanvankelijk was het doel om per fold in de K-Fold cross-validatie de accuracy te analyseren bij verschillende confidence-thresholds. Echter, deze aanpak was niet informatief, omdat het model in alle fold een accuracy van 100% behaalde (zie Figuur 4C). Daardoor bleef de accuracy constant, ongeacht de gekozen threshold. In plaats daarvan is er gekozen voor een retentie-analyse, waarbij per threshold wordt gekeken welk percentage van de voorspellingen een confidence-score boven de drempel haalt. Op basis van deze analyse is gekozen voor een confidence-threshold van 0,80, waarbij 95% van de voorspellingen automatisch verwerkt kan worden zonder reconstructie (zie Figuur 6C).

F. Herstel van vervormde of verborgen cijfers met Graph Neural Networks

Wanneer het OCR-model er niet in slaagt om cijfers met voldoende zekerheid te classificeren, wordt een aanvullend model ingezet om incomplete cijfers te reconstrueren. Hiervoor wordt gebruikgemaakt van een GNN dat in staat is om de structurele vorm van cijfers te begrijpen en te herstellen.

1) *Representatie als graaf*: Om cijfers te reconstrueren met een GNN, moeten de oorspronkelijke beelden omgezet worden naar een geschikte graafstructuur. Dit begint met het voorbereiden van de cijferafbeeldingen. In de eerste stap word het relevante deel van de afbeelding bijgesneden, zodat alleen het daadwerkelijke cijfer over blijft en de hoeveelheid achtergrond word geminaliseerd. Dit uitsnijden gebeurt met een kleine marge rondom het cijfer, zodat de graaf niet te krap is en verbindingen te dicht langs de rand liggen. Vervolgens wordt



Figuur 14: Proces van beeld naar graafrepresentatie.

de afbeelding herschaald naar een vast formaat van 64x64 en gepositioneerd met padding, zodat het cijfer altijd gecentreerd staat. Op deze manier ontstaan een uniforme invoervorm, die onafhankelijk is van de orginele afmetingen of de positie van het cijfer.

De voorbewerkt afbeelding wordt daarna geskeletoniseerd. Dit betekent dat alleen de lijnen overblijven die de vorm van het cijfer representeren. De resultaten worden eerst gebruik om een graaf op te stellen op de originele 64x64 afmeting. Elke pixel op het skelet wordt gezien als een knoop en knopen worden onderling verbonden als ze elkaars directe buren zijn (ook diagonaal).

Omdat de ruwe graaf direct is afgeleid vanuit de skeletonpixels, heeft deze graaf veel knopen (64x64) en verbindingen. Dit zorgt niet alleen voor een zware invoer voor het model, maar maakt de graaf ook erg gevoelig voor kleine verstoringen in het beeld zoals ruis of lichte vervormingen. Om dit te verbeteren, wordt de graaf vereenvoudigd door de skeleton te projecteren op een vast raster. Hierbij is er gekozen voor een grid van 8 bij 8 cellen.

De keuze voor een 8 bij 8 grid is bewust gemaakt. Omdat de afmetingen van het orginele beeld 64x64 is, kan een grid van 8x8 het beeld exact en symmetrisch verdelen in vlakken van 8x8 pixels. Op deze manier blijven de proporties behouden en krijgt elk deel een gelijke representatie binnen de graaf. Kleinere grids, zoals een 4x4 is getest maar bleek te grof om voldoende onderscheid te maken tussen de verschillende cijfers, doordat er teveel detail verloren gaat.

Daarnaast is enkel het skeleton, zonder projectie niet bruikbaar voor het type reconstructie dat dit model uitvoert. Het model moet namelijk voorspellen welke knopen met elkaar verbonden moet zijn en daarvoor is een consistente set knopen nodig die in elke graaf voorkomen. Door de skeletonstructuur te projecteren op een vast grid, ontstaan er 64 vaste knopen die het model leert te gebruiken. Dit maakt het mogelijk om verbindingen te reconstrueren, ook al ze (deels) ontbreken in de invoer.

Een 8x8 grid beïndt hierin de juiste balans. Het verminder de complexiteit van de invoer en beïndt daarbij voldoende structuur om de vorm van het cijfer te reconstrueren. In Figuur 14 is het proces van de skeletonisering weergegeven.

Om de positie-informatie van elke knoop mee te geven aan

het model, krijgt elke knoop een unieke one-hot vectore als feature. Deze features samen vormen een identity metrix van 64x64. Op deze manier weet het model welke knoop het is in het rast, zonder gebruik te maken van expliciete coördinaten. Het model is zo alleen afhankelijk van de relatieve structuur.

2) Trainingsopzet en iteratieve optimalisatie: Het model is getraind als een denoising Variational Graph Autoencoder (VGAE). Hierbij krijgt het model niet de volledige structuur van een graaf als input, maar een versie waarbij bewust een deel van de verbindingen is verwijderd. Het doel is om op basis van deze onvolledige graaf de oorspronkelijke verbindingen zo goed mogelijk te reconstrueren. Dit sluit goed aan bij de toepassing waarin het model gebruikt gaat worden, reconstrueren van incomplete cijfers.

Om het vermogen om te reconstrueren te trainen, wordt tijdens iedere epoch de inputgraaf van de trainset opnieuw gecorrumpeerd. Dit gebeurt door willekeurig 25% van de bestaande randen te verwijderen. Omdat dit bij elke epoch opnieuw gebeurd, krijgt het model continu andere corrupties te zien, wat zorgt voor een robuust model. Voor de validatie is de corruptie vastgezet. Elke graaf in de validatieset wordt eenmalig gecorrumpeerd en blijft daarna gelijk. Op deze manier kunnen de prestaties van het model op een consistente manier worden geëvalueerd.

De reconstructietaak wordt geformuleerd als een link prediction-probleem: op basis van de latente knooprepresentaties voorspelt het mode of er een verbinding hoort ze zijn tussen twee knopen. Hiervoor worden er ook random negatieve voorbeelden gegenereerd, wat verbindingen zijn die in de graaf geen verbinding hebben. Deze aanpak leert het model niet alleen om juiste verbindingen te reconstrueren, maar ook foutieve verbindingen te vermijden.

De modelarchitectuur is gebaseerd op de VGAE van Kipf & Welling [52], waarbij de encoder bestaat uit GNN-lagen die knopen omzetten in latente vectorrepresentaties, en de decoder op basis hiervan voorspelt welke knopen verbonden zouden moeten zijn.

Om de prestaties te verbeteren, is het model via een iteratief proces verbeterd. In de eerste stappen zijn de GCN-lagen vervangen door GINConv-lagen en is het netwerk verdiept voor een groter contextbereik, gebaseerd op Xu et al. [53]. Ook is Jumping Knowledge toegevoegd om informatie uit eerdere

lagen te behouden [53]. Bij de decoder zijn een bilineaire decoder [54] en een MLP-decoder [55] getest om complexere relaties te modelleren.

Daarnaast zijn dropout en batch normalisatie toegevoegd voor regularisatie[53] en is contrastive learning geïmplementeerd met NT-Xent loss [56]. Verder is geëxperimenteerd met hard negative sampling [57] en met een verschillende verhoudingen tussen negatieve en positieve voorbeelden. Een volledig overzicht van alle iteraties, inclusief onderbouwing en resultaten, is te vinden in Bijlage D.

3) Geselecteerde architectuur: Het eindmodel combineert de elementen uit de verschillende iteraties. De encoder bestaat uit drie GINConv-lagen met batch normalisatie, dropout ($p=0.3$) en Jumping Knowledge in concat-modus, zoals aanbevolen door Xu et al. [53]. Voor de decoder is gekozen voor een MLP [55], omdat deze de beste balans gaf tussen precisie en volledigheid van de reconstructie.

Daarnaast is gebruikgemaakt van hard negative sampling \cite{yang_understanding_2020} in een 3:1 verhouding ten opzichte van positieve voorbeelden. Dit leverde robuuste resultaten op, vooral bij hogere drop rates. Contrastive learning is bewust niet meegenomen in het eindmodel, omdat dit in de experimenten alleen effectief bleek bij lichte corruptie.

G. Optimalisatie voor TinyML

Om de ontwikkelde modellen geschikt te maken voor embedded systemen met beperkte rekenkracht en geheugen, worden verschillende TinyML-optimalisatietechnieken toegepast. Deze optimalisaties zorgen ervoor dat het volledige AMR-systeem, inclusief detectie, OCR en reconstructie, efficiënt kan draaien op een Raspberry Pi. De volgende optimalisaties worden per model toegepast:

- Quantisatie naar INT8: Na training wordt het model gequantiseerd naar 8-bit integers (INT8) met behulp van post-training quantization. Dit verkleint en versnel het model [25].
- Modelpruning: Overbodige parameters en filters worden verwijderd om het model compacter en efficiënter te maken, met behoud van herkenningsnauwkeurigheid [25].
- Export naar TensorFlow Lite: Het getrainde model wordt geëxporteerd naar een geschikt formaat om integratie mogelijk te maken [25].

REFERENTIES

- [1] S. Malik, S. Agarwal, S. Aluvala, J. S. Bali, R. Garg en M. Hussien, “Internet of Things Enabled Automatic Meter Reading”, in *2023 International Conference on Smart Devices (ICSD)*, mei 2024, p. 1–5. doi: 10.1109/ICSD60021.2024.10751420. adres: <https://ieeexplore.ieee.org/abstract/document/10751420> (bezocht op 22-03-2025).
- [2] Y. Kabalci, “A survey on smart metering and smart grid communication”, *Renewable and Sustainable Energy Reviews*, jrg. 57, p. 302–318, mei 2016, issn: 1364-0321. doi: 10.1016/j.rser.2015.12.114. adres: <https://www.sciencedirect.com/science/article/pii/S1364032115014975> (bezocht op 17-03-2025).
- [3] G. Salomon, R. Laroca en D. Menotti, “Deep Learning for Image-based Automatic Dial Meter Reading: Dataset and Baselines”, in *2020 International Joint Conference on Neural Networks (IJCNN)*, arXiv:2005.03106 [cs], jul 2020, p. 1–8. doi: 10.1109/IJCNN48605.2020.9207318. adres: <http://arxiv.org/abs/2005.03106> (bezocht op 05-03-2025).
- [4] R. Laroca, V. Barroso, M. A. Diniz, G. R. Gonçalves, W. R. Schwartz en D. Menotti, “Convolutional Neural Networks for Automatic Meter Reading”, *Journal of Electronic Imaging*, jrg. 28, nr. 01, p. 1, feb 2019, arXiv:1902.09600 [cs], issn: 1017-9909. doi: 10.1117/1.JEI.28.1.013023. adres: <http://arxiv.org/abs/1902.09600> (bezocht op 06-03-2025).
- [5] K. Koščević en M. Subašić, “Automatic Visual Reading of Meters Using Deep Learning”, en, feb 2019, p. 1–6. doi: 10.20532/ccvw.2018.0002. adres: <https://www.fer.unizg.hr/crv/ccvw.2018.0002> (bezocht op 06-03-2025).
- [6] R. Laroca, A. B. Araujo, L. A. Zanlorensi, E. C. De Almeida en D. Menotti, “Towards Image-Based Automatic Meter Reading in Unconstrained Scenarios: A Robust and Efficient Approach”, *IEEE Access*, jrg. 9, p. 67 569–67 584, 2021, Conference Name: IEEE Access, issn: 2169-3536. doi: 10.1109/ACCESS.2021.3077415. adres: <https://ieeexplore.ieee.org/document/9422699/?arnumber=9422699> (bezocht op 06-03-2025).
- [7] F. Martinelli, F. Mercaldo en A. Santone, “Water Meter Reading for Smart Grid Monitoring”, en, *Sensors*, jrg. 23, nr. 1, p. 75, dec 2022, issn: 1424-8220. doi: 10.3390/s23010075. adres: <https://www.mdpi.com/1424-8220/23/1/75> (bezocht op 10-03-2025).
- [8] M. Bertalmio, G. Sapiro, V. Caselles en C. Ballester, “Image inpainting”, en, in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, Not Known: ACM Press, 2000, p. 417–424, ISBN: 978-1-58113-208-3. doi: 10.1145/344779.344972. adres: <http://portal.acm.org/citation.cfm?doid=344779.344972> (bezocht op 12-03-2025).
- [9] Z. Qin, Q. Zeng, Y. Zong en F. Xu, “Image inpainting based on deep learning: A review”, *Displays*, jrg. 69, p. 102 028, sep 2021, issn: 0141-9382. doi: 10.1016/j.displa.2021.102028. adres: <https://www.sciencedirect.com/science/article/pii/S0141938221000391> (bezocht op 12-03-2025).
- [10] A. Criminisi, P. Perez en K. Toyama, “Region Filling and Object Removal by Exemplar-Based Image Inpainting”, en, *IEEE Transactions on Image Processing*, jrg. 13, nr. 9, p. 1200–1212, sep 2004, issn: 1057-7149. doi: 10.1109/TIP.2004.833105. adres: <http://ieeexplore.ieee.org/document/1323101/> (bezocht op 12-03-2025).
- [11] X. Zhang, D. Zhai, T. Li, Y. Zhou en Y. Lin, “Image inpainting based on deep learning: A review”, *Information Fusion*, jrg. 90, p. 74–94, feb 2023, issn: 1566-2535. doi: 10.1016/j.inffus.2022.08.033. adres: <https://www.sciencedirect.com/science/article/pii/S1566253522001324> (bezocht op 12-03-2025).
- [12] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell en A. A. Efros, “Context Encoders: Feature Learning by Inpainting”, en, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, jun 2016, p. 2536–2544, ISBN: 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.278. adres: <http://ieeexplore.ieee.org/document/7780647/> (bezocht op 12-03-2025).
- [13] W. Quan, J. Chen, Y. Liu, D.-M. Yan en P. Wonka, “Deep Learning-Based Image and Video Inpainting: A Survey”, en, *International Journal of Computer Vision*, jrg. 132, nr. 7, p. 2367–2400, jul 2024, issn: 0920-5691, 1573-1405. doi: 10.1007/s11263-023-01977-6. adres: <https://link.springer.com/10.1007/s11263-023-01977-6> (bezocht op 12-03-2025).
- [14] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu en T. S. Huang, *Generative Image Inpainting with Contextual Attention*, arXiv:1801.07892 [cs], mrt 2018. doi: 10.48550/arXiv.1801.07892. adres: <http://arxiv.org/abs/1801.07892> (bezocht op 17-03-2025).
- [15] A. Radford, L. Metz en S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, arXiv:1511.06434 [cs], jan 2016. doi: 10.48550/arXiv.1511.06434. adres: <http://arxiv.org/abs/1511.06434> (bezocht op 17-03-2025).
- [16] M. Arjovsky, S. Chintala en L. Bottou, “Wasserstein Generative Adversarial Networks”, en, in *Proceedings of the 34th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, jul 2017, p. 214–223. adres: <https://proceedings.mlr.press/v70/arjovsky17a.html> (bezocht op 17-03-2025).

- [17] M. Mirza en S. Osindero, *Conditional Generative Adversarial Nets*, arXiv:1411.1784 [cs], nov 2014. doi: 10.48550/arXiv.1411.1784. adres: <http://arxiv.org/abs/1411.1784> (bezocht op 17-03-2025).
- [18] P. Isola, J.-Y. Zhu, T. Zhou en A. A. Efros, “Image-To-Image Translation With Conditional Adversarial Networks”, 2017, p. 1125–1134. adres: https://openaccess.thecvf.com/content_cvpr_2017/html/Isola_Image - To - Image_Translation_With_CVPR_2017_paper.html (bezocht op 17-03-2025).
- [19] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao en B. Catanzaro, *Image Inpainting for Irregular Holes Using Partial Convolutions*, arXiv:1804.07723 [cs], dec 2018. doi: 10.48550/arXiv.1804.07723. adres: <http://arxiv.org/abs/1804.07723> (bezocht op 17-03-2025).
- [20] O. Elharrouss, N. Almaadeed, S. Al-Maadeed en Y. Akbari, “Image inpainting: A review”, *Neural Processing Letters*, jrg. 51, nr. 2, p. 2007–2028, apr 2020, arXiv:1909.06399 [cs], ISSN: 1370-4621, 1573-773X. doi: 10.1007/s11063-019-10163-0. adres: <http://arxiv.org/abs/1909.06399> (bezocht op 17-03-2025).
- [21] B. Sanchez-Lengeling, E. Reif, A. Pearce en A. B. Wiltschko, “A Gentle Introduction to Graph Neural Networks”, en, *Distill*, jrg. 6, nr. 9, e33, sep 2021, ISSN: 2476-0757. doi: 10.23915/distill.00033. adres: <https://distill.pub/2021/gnn-intro> (bezocht op 21-03-2025).
- [22] Y. Wang, Z. Li en A. Barati Farimani, “Graph Neural Networks for Molecules”, en, in *Machine Learning in Molecular Sciences*, C. Qu en H. Liu, red., Cham: Springer International Publishing, 2023, p. 21–66, ISBN: 978-3-031-37196-7. doi: 10.1007/978-3-031-37196-7_2. adres: https://doi.org/10.1007/978-3-031-37196-7_2 (bezocht op 21-03-2025).
- [23] S. Verma, A. Sharma, R. Sheshadri en S. Raman, “GraphFill: Deep Image Inpainting using Graphs”, en, in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA: IEEE, jan 2024, p. 4984–4994, ISBN: 9798350318920. doi: 10.1109/WACV57701.2024.00492. adres: <https://ieeexplore.ieee.org/document/10483644/> (bezocht op 12-03-2025).
- [24] A. Sharma, S. Singh en S. Ratna, *Graph Neural Network based Handwritten Trajectories Recognition*, arXiv:2405.09247 [cs], mei 2024. doi: 10.48550/arXiv.2405.09247. adres: <http://arxiv.org/abs/2405.09247> (bezocht op 12-03-2025).
- [25] V. Tsoukas, <https://orcid.org/0000-0003-0081-2052>, View Profile e.a., “A Review on the emerging technology of TinyML”, *ACM Computing Surveys*, jrg. 56, nr. 10, p. 1–37, jun 2024, Publisher: Association for Computing Machinery. doi: 10.1145/3661820. adres: <https://dl.acm.org/doi/full/10.1145/3661820> (bezocht op 17-03-2025).
- [26] I. Fedorov, R. P. Adams, M. Mattina en P. Whatmough, “SpArSe: Sparse Architecture Search for CNNs on Resource-Constrained Microcontrollers”, in *Advances in Neural Information Processing Systems*, deel 32, Curran Associates, Inc., 2019. adres: https://proceedings.neurips.cc/paper_files/paper/2019/hash/044a23cadb567653eb51d4eb40acaa88 - Abstract.html (bezocht op 17-03-2025).
- [27] T. Elsken, J. H. Metzen en F. Hutter, “Neural Architecture Search: A Survey”, *Journal of Machine Learning Research*, jrg. 20, nr. 55, p. 1–21, 2019, ISSN: 1533-7928. adres: <http://jmlr.org/papers/v20/18-598.html> (bezocht op 17-03-2025).
- [28] R. David, J. Duke, A. Jain e.a., “TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems”, en,
- [29] *microsoft/ELL*, original-date: 2016-09-27T19:12:29Z, mrt 2025. adres: <https://github.com/microsoft/ELL> (bezocht op 17-03-2025).
- [30] Y. Song, T. Wang, P. Cai, S. K. Mondal en J. P. Sahoo, “A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities”, en, *ACM Computing Surveys*, jrg. 55, nr. 13s, p. 1–40, dec 2023, ISSN: 0360-0300, 1557-7341. doi: 10.1145/3582688. adres: <https://dl.acm.org/doi/10.1145/3582688> (bezocht op 18-03-2025).
- [31] G. Koch, R. Zemel en R. Salakhutdinov, “Siamese Neural Networks for One-shot Image Recognition”, en, 2015.
- [32] J. Snell, K. Swersky en R. Zemel, “Prototypical Networks for Few-shot Learning”, in *Advances in Neural Information Processing Systems*, deel 30, Curran Associates, Inc., 2017. adres: https://proceedings.neurips.cc/paper_files/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42 - Abstract.html (bezocht op 18-03-2025).
- [33] EMR, *MeterRead Dataset*, en, mrt 2023. adres: <https://universe.roboflow.com/emr-2fz0i/meterread> (bezocht op 15-05-2025).
- [34] EGH400AMRSOLUTION, *AMR-RAD-DATA Dataset*, en, okt 2024. adres: <https://universe.roboflow.com/egh400amrsolution/amr-rad-data> (bezocht op 15-05-2025).
- [35] Juanna, *Wired 3-Pole MCB Dataset*, en, feb 2025. adres: <https://universe.roboflow.com/juanna/wired-3-pole-mcb> (bezocht op 15-05-2025).
- [36] Ultralytics, *Tips for Best Training Results*, en, 2024. adres: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results (bezocht op 15-05-2025).
- [37] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto en E. A. B. da Silva, “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit”, en, *Electronics*, jrg. 10, nr. 3, p. 279, jan 2021, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2079-9292. doi: 10.3390/electronics10030279. adres: <https://www.mdpi.com/2079-9292/10/3/279> (bezocht op 19-03-2025).
- [38] Y.-J. Cho, “Weighted Intersection over Union (wIoU) for Evaluating Image Segmentation”, *Pattern Recognition Letters*, jrg. 185, p. 101–107, sep 2024, arXiv:2107.09858 [cs], ISSN: 01678655. doi: 10.1016/j.patrec.2024.07.011. adres: <http://arxiv.org/abs/2107.09858> (bezocht op 26-03-2025).
- [39] Z. Li, Y. Dong, L. Shen e.a., “Development and challenges of object detection: A survey”, *Neurocomputing*, jrg. 598, p. 128–102, sep 2024, ISSN: 0925-2312. doi: 10.1016/j.neucom.2024.128102. adres: <https://www.sciencedirect.com/science/article/pii/S0925231224008737> (bezocht op 19-03-2025).
- [40] J. Redmon, S. Divvala, R. Girshick en A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, jun 2016, p. 779–788. doi: 10.1109/CVPR.2016.91. adres: <https://ieeexplore.ieee.org/document/7780460> (bezocht op 19-03-2025).
- [41] T. Diwan, G. Anirudh en J. V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications”, en, *Multimedia Tools and Applications*, jrg. 82, nr. 6, p. 9243–9275, mrt 2023, ISSN: 1573-7721. doi: 10.1007/s11042-022-13644-y. adres: <https://doi.org/10.1007/s11042-022-13644-y> (bezocht op 19-03-2025).
- [42] B. Nguyen van, A. Nguyen, K. Tran-Trung e.a., “Water Meter Reading Based on Text Recognition Techniques and Deep Learning”, *IEEE Access*, jrg. 13, p. 41422–41434, 2025, Conference Name: IEEE Access, ISSN: 2169-3536. doi: 10.1109/ACCESS.2025.3547225. adres: <https://ieeexplore.ieee.org/document/10908809/?arnumber=10908809> (bezocht op 19-03-2025).
- [43] N. Jegham, C. Y. Koh, M. Abdellati en A. Hendawi, *YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions*, arXiv:2411.00201 [cs] version: 3, feb 2025. doi: 10.48550/arXiv.2411.00201. adres: <http://arxiv.org/abs/2411.00201> (bezocht op 19-03-2025).
- [44] Y. Tian, Q. Ye en D. Doermann, *YOLOv12: Attention-Centric Real-Time Object Detectors*, arXiv:2502.12524 [cs], feb 2025. doi: 10.48550/arXiv.2502.12524. adres: <http://arxiv.org/abs/2502.12524> (bezocht op 19-03-2025).
- [45] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens en Q. V. Le, *Learning Data Augmentation Strategies for Object Detection*, arXiv:1906.11172 [cs], jun 2019. doi: 10.48550/arXiv.1906.11172. adres: <http://arxiv.org/abs/1906.11172> (bezocht op 20-03-2025).
- [46] D. Islam, T. Mahmud en T. Chowdhury, “An efficient automated vehicle license plate recognition system under image processing”, en, *Indonesian Journal of Electrical Engineering and Computer Science*, jrg. 29, nr. 2, p. 1055, feb 2023, ISSN: 2502-4760, 2502-4752. doi: 10.11591/ijeecs.v29.i2.pp1055-1062. adres: <https://ijeeecs.iaescore.com/index.php/IJEECS/article/view/29643> (bezocht op 26-03-2025).
- [47] J. Song, W. Jiao, K. Lankowicz, Z. Cai en H. Bi, “A two-stage adaptive thresholding segmentation for noisy low-contrast images”, en, *Ecological Informatics*, jrg. 69, p. 101–132, jul 2022, ISSN: 15749541. doi: 10.1016/j.ecoinf.2022.101632. adres: <https://linkinghub.elsevier.com/retrieve/pii/S1574954122000814> (bezocht op 18-05-2025).
- [48] OpenCV: *Contour Properties*. adres: https://docs.opencv.org/4.x/d1/d32/tutorial_py_contour_properties.html (bezocht op 18-05-2025).
- [49] Model - *PaddleOCR Documentation*. adres: https://paddlepaddle.github.io/PaddleOCR/main/en/ppoer/model_list.html (bezocht op 20-03-2025).
- [50] K. Team, *Keras documentation: Simple MNIST convnet*, en. adres: https://keras.io/examples/vision/mnist_convnet/ (bezocht op 17-05-2025).

- [51] F. Chollet, *Deep learning with Python*, en, Second edition. Shelter Island: Manning Publications, 2021, OCLC: on1289290141, ISBN: 978-1-61729-686-4.
- [52] T. N. Kipf en M. Welling, *Variational Graph Auto-Encoders*, arXiv:1611.07308 [stat], nov 2016. doi: 10.48550/arXiv.1611.07308. adres: <http://arxiv.org/abs/1611.07308> (bezocht op 21-03-2025).
- [53] K. Xu, W. Hu, J. Leskovec en S. Jegelka, *How Powerful are Graph Neural Networks?*, en, arXiv:1810.00826 [cs], feb 2019. doi: 10.48550/arXiv.1810.00826. adres: <http://arxiv.org/abs/1810.00826> (bezocht op 21-03-2025).
- [54] R. v. d. Berg, T. N. Kipf en M. Welling, *Graph Convolutional Matrix Completion*, en, arXiv:1706.02263 [stat], okt 2017. doi: 10.48550/arXiv.1706.02263. adres: <http://arxiv.org/abs/1706.02263> (bezocht op 21-05-2025).
- [55] Y. Wang, B. Hooi, Y. Liu, T. Zhao, Z. Guo en N. Shah, “Flashlight: Scalable Link Prediction With Effective Decoders”, en, in *Proceedings of the First Learning on Graphs Conference*, ISSN: 2640-3498, PMLR, dec 2022, 14:1-14:17. adres: <https://proceedings.mlr.press/v198/wang22a.html> (bezocht op 21-05-2025).
- [56] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang en Y. Shen, “Graph Contrastive Learning with Augmentations”, in *Advances in Neural Information Processing Systems*, deel 33, Curran Associates, Inc., 2020, p. 5812–5823. adres: https://proceedings.neurips.cc/paper_files/paper/2020/hash/3fe230348e9a12c13120749e3f9fa4cd-Abstract.html (bezocht op 21-05-2025).
- [57] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou en J. Tang, “Understanding Negative Sampling in Graph Representation Learning”, en, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual Event CA USA: ACM, aug 2020, p. 1666–1676, ISBN: 978-1-4503-7998-4. doi: 10.1145/3394486.3403218. adres: <https://dl.acm.org/doi/10.1145/3394486.3403218> (bezocht op 21-05-2025).
- [58] P. Geometric, *models.VGAE*. adres: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.models.VGAE.html (bezocht op 21-05-2025).

index	Meter	0	1	2	3	4	5	6	7	8	9
1	ELO2101A	0	1	2	3	4	5	6	7	8	9
2	ELO2102A	0	1	2	3	4	5	6	7	8	9
3	ELO 2103A	0	1	2	3	4	5	6	7	8	9
4	nansen	0	1	2	3	4	5	6	7	8	9
5	nansen (2)	0	1	2	3	4	5	6	7	8	
6	nansen (3)	0		2	3		5		7	8	
7	nansen M1AT-A	0	1	2	3	4	5	6	7	8	9
8	elster ME21A	0	1	2	3	4	5	6	7	8	9
9	fae	0	1	2	3	4	5	6	7	8	9
10	fae MF-79G		1			4	5	6	7	8	9
11	FAE MF-97G	0	1	2	3	4	5		7	8	9
12	FAE MFT-120G	0	1	2		4	5	6	7	8	9
13	fae Cronos 6001-A	0	1	2	3	4	5	6	7	8	9
14	fae cronos 6003	0	1	2	3	4	5	6	7	8	9
15	FAE MFB-04G	0	1	2	3	4	5	6	7	8	9
16	Schlumberger	0	1	2	3	4	5	6	7	8	9
17	Landis + Gyr M12AM	0	1	2	3	4	5	6	7	8	
18	Landis & Gyr	0	1	2	3	4	5	6	7	8	9
19	Landis & Gyr D-58JC	0	1	2	3	4	5	6	7	8	9
20	inepar		1	2	3		5	6	7	8	9
21	General electric	0	1	2	3	4	5	6	7	8	9

Figuur 1A: Enter Caption

BIJLAGE A

VISUELE VARIATIE IN CIJFERS OP ANALOGE METERS

BIJLAGE B

UITWERKING BINARISATIE-METHODE

Deze bijlage beschrijft de volledige binarisatiestap zoals toegepast in de preprocessingpipeline. Het doel van deze stappen is het isoleren van betrouwbare cijferstructuren uit displaybeelden. Door het combineren van meerdere beeldverwerkingsstappen ontstaat een robuust binair beeld dat geschikt is voor verdere segmentatie en classificatie.

A. Achtergrond egaliseren (flattening)

Om de invloed van ongelijke belichting te minimaliseren, is het grijswaardenbeeld eerst geëgaliseerd. Hiervoor is een Gaussiaanse vervaging toegepast met een groot kernel (35x35), die variaties in de achtergrond vervaagt zonder scherpe details te verliezen.

Onderstaande afbeelding toont een overzicht van cijfers (0–9) afkomstig uit echte beelden van analoge meters. De voorbeelden tonen de visuele variatie in typografie, dikte, vorm, achtergrond en contrast. Deze analyse vormde de basis voor de selectie van fonts bij het genereren van synthetische data.

De vervaagde achtergrond wordt vervolgens van het originele beeld afgetrokken, waardoor het contrast tussen de voorgrond en de achtergrond toeneemt. Het resultaat wordt daarna opnieuw geschaald naar het standaardbereik van 0-255.

B. Threshold berekenen (Sauvola)

Voor de binairisatie is gebruik gemaakt van de Sauvola-methode. Er zijn twee thresholds berekend:

- Een strikte drempel met $k = -0,1$, bedoeld om alleen de sterkste en meest betrouwbare structuren te behouden (marker).
- Een zachtere drempel met $k = 0,5$, die ook zwakkere of deels vervaagde structuren meeneemt (masker).

C. Binaire beelden maken

Op basis van bovenstaande drempels zijn twee binaire versies van het beeld gemaakt, de marker en het masker. De marker bevat uitsluitend duidelijke cijferdelen. Het masker heeft een ruimer gebied waarin de volledige cijfers zich mogelijk bevinden.

D. Marker filteren

Om in het markerbeeld alleen zekere en relevante pixels te behouden, zijn de volgende stappen uitgevoerd:

- Het verwijderen van kleine vlekjes die te klein zijn om een cijfer te zijn. De waarde 150 is gebaseerd op de minimale grootte van cijferstructuren in de dataset.
- Een morfologische opening met een relatief grote structuur (disk2) verwijdert smalle verbindingen en bruggetjes tussen ruis en het cijfer.
- Fijnere correctie met morfologische opening (disk1) na de voorgaande stap om resterende oneffenheden te corrigeren.
- Twee keer erosie (disk1) wordt toegepast om de structuur compacter te maken, zodat alleen de kern van het cijfer overblijft. De gekozen structuur (disk1) voorkomt overmatig verlies van dunne cijfers zoals de '1' of '7'.
- Tot slot worden kleine restjes die na erosie ontstaan verwijderd met een grootte van 50. De lagere drempel van 50 is afgestemd op de kleinere omvang van cijfers na eerdere bewerkingen.

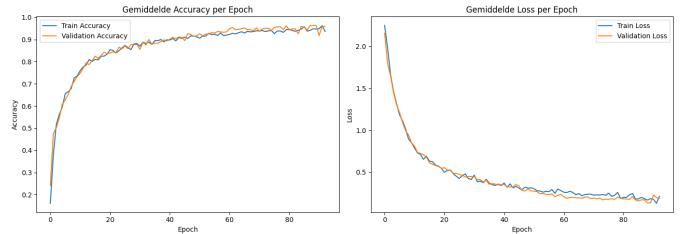
E. Masker filteren

Het masker wordt minder agressief opgeschoond om de volledig vorm van de cijfers te behouden:

- Het verwijderen van kleine objecten met een drempel van 150, zoals bij de marker.
- Een morfologische opening (disk1) verbreekt ongewenste verbindingen tussen ruis en cijferdelen, zodat er niet teveel ruis wordt meegenomen tijdens de reconstructie.

F. Reconstructie

De reconstructiestap bestaat uit het uitbreiden van de marker binnen de grenzen van het masker via morfologische reconstructie. Hierdoor worden de cijfers volledig hersteld, terwijl ruis die buiten het masker valt wordt uitgesloten.



Figuur 1C: Accuracy en loss curve training CNN strart model

G. Nabewerking

Na de reconstructie wordt een extra morfologische opening toegepast (disk1) om kleine verbindingen te verwijderen die bij contourdetectie tot verstoring kunnen zorgen.

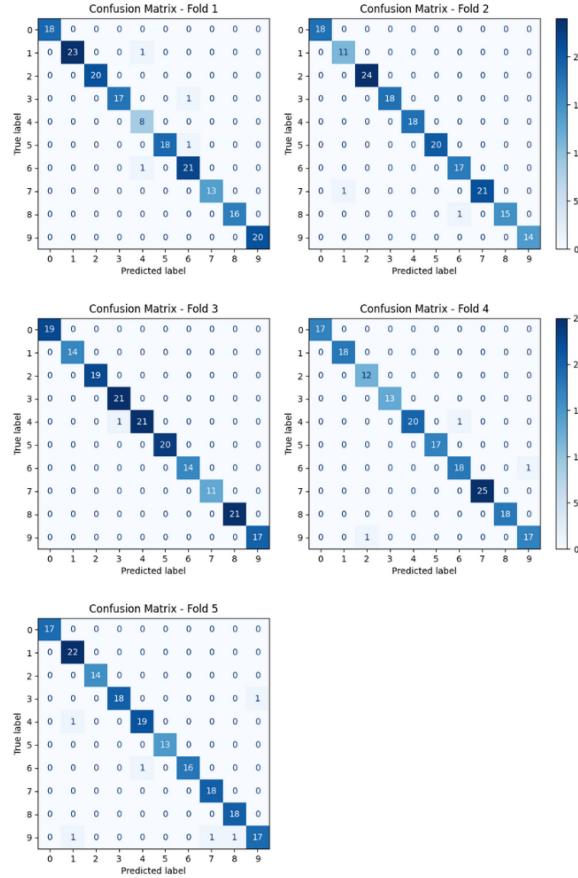
H. Laatste filtering van ruis

Als laatste stap worden overgebleven irrelevante structuren verwijderd met een drempel van 150.

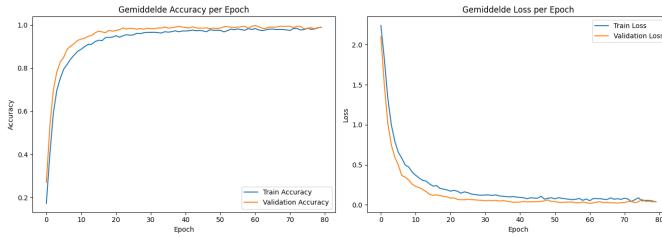
BIJLAGE C TRAINRESULTATEN CNN EN THRESHOLDSELECTIE

In deze bijlage zijn de trainresultaten weergegeven van de verschillende CNN-modellen die tijdens dit onderzoek zijn getest. Per model worden de accuracy en loss curve gegeven, samen met de confusion matrix per fold. Dit geeft een duidelijk beeld van het leerproces en mogelijke overfitting.

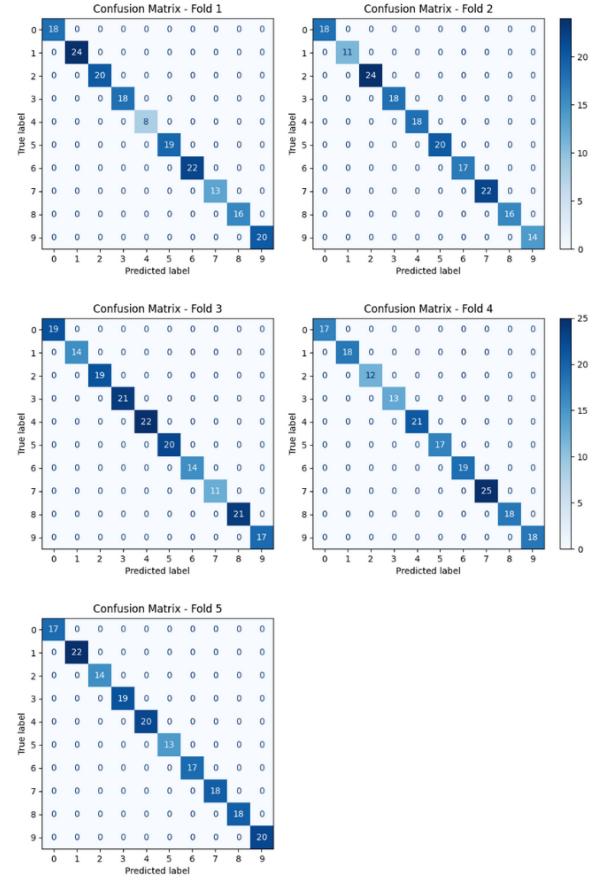
Daarnaast zijn per fold de resultaten van de retentie-analyse weergegeven. Deze geven inzicht in het effect van verschillende confidence-thresholds op het percentage voorspelling dat automatisch verwerkt kan worden zonder reconstructie. Zowel het startmodel, het complexere model als het eindmodel zijn gevisualiseerd.



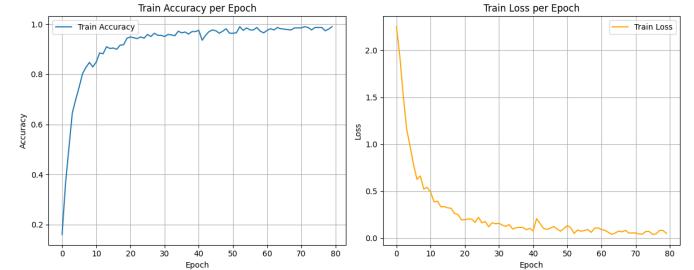
Figuur 2C: Confusion matrix per fold CNN startmodel



Figuur 3C: Accuracy en loss curve training CNN verhoogde complexiteit



Figuur 4C: Confusion matrix per fold CNN verhoogde complexiteit



Figuur 5C: Accuracy en loss curve training CNN met train-en validatieset.

BIJLAGE D ITERATIEF PROCES GNN

A. Iteratie 1: denoising VGAE (startmodel)

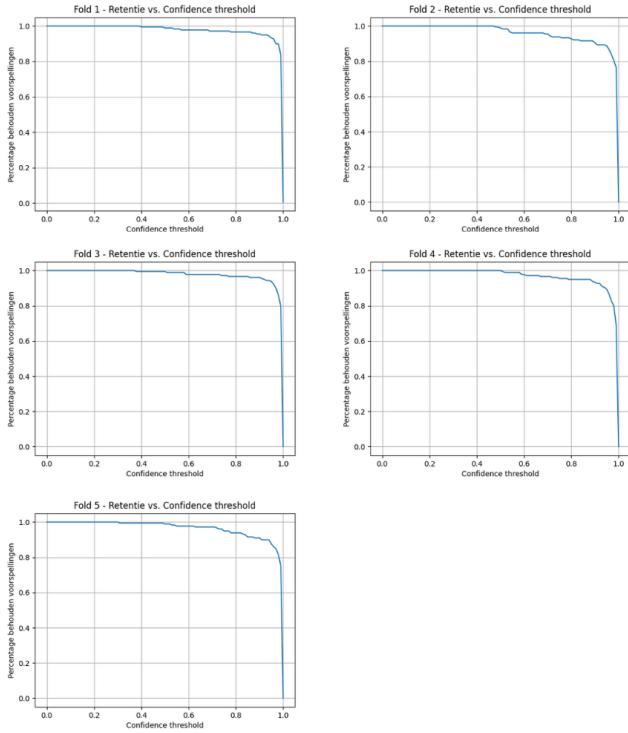
doel: Het doel van deze eerste iteratie is het opzetten van een eenvoudige maar functioneel startmodel. Dit startmodel dient als startpunt om vanuit verder te itereren.

Beschrijving: Het model is gebaseerd op de standaardimplementatie van de VGAE uit PyTorch Geometric [58], die aansluit bij de architectuur geïntroduceerd door Kipf & Welleing [52]. De encoder bestaat uit twee GCN-lagen en de decoder maakt gebruik van een inproduct om knooppalen te vergelijken. Om het model te trainen als denoising autoencoder

worden tijdens training willekeurig 25% van de edges verwijderd. Hierdoor leert het model om de structuur van de graaf te reconstrueren op basis van corrupte input. Voor validatie is dezelfde corruptiegraad gebruikt, maar hierbij zijn de grafen eenmalig gecorrumpeerd en blijven deze blijven tijdens de training hetzelfde.

Onderbouwing: Deze architectuur is gekozen omdat het een standaardaanpak is voor link prediction met grafen en daarmee een goed startpunt is voor volgende verbeteringen.

Resultaten: De loss curve (Figuur 1D (a)) laat zien dat het model snel convergeert en daarna langzaam blijft leren. Uit de



Figuur 6C: Percentage behouden voorspellingen bij oplopende confidence-thresholds (per fold).

validatie F1-score (Figuur 1D (b)) is te zien dat de recall hoog is en de precision heel laag. In de reconstructies (Figuur 1D (c)) is dit duidelijk terug te zien. Het model voorspelt heel veel extra verbindingen. Het lijkt bijna op een triviale oplossing waarbij het model vaak dezelfde edges voorspelt ongeacht de graaf. Dezelfde negatieve edges komen namelijk vaak terug.

B. Iteratie 2: GIN-lagen gebruiken

Doel: Het vergroten van het onderscheidend vermogen van de encoder, met als doel het beter herkennen van subtiële verschillen in cijferstructuur.

Beschrijving: De GCN-lagen zijn vervangen door GINConv-lagen. De encoder bestaat nu uit een GINConv-laag voor feature extractie en twee aparte lagen voor mu en logstd. GINConv is krachtiger dan de standaard GCN-lagen [53].

Onderbouwing: GINConv kan beter onderscheid maken tussen verschillende lokale structuren. Bij cijferherkenning is dit belangrijk, omdat veel cijfers maar een paar knopen verschil hebben. Zeker als er verbindingen missen.

Resultaten: De F1-scores (Figuur 2D (b)) zijn duidelijk verbeterd ten opzichte van het stratmodel. Bij drop rates van 0.25 en 0.3 scoort het model nu duidelijk hoger. De precision bij alle drop rates zijn ook duidelijk verbeterd. Dit laat zien dat de encoder met GIN-lagen robuuster is geworden voor structuurverlies in de inputgrafen.

De reconstructievoorbeelden (Figuur 2D (c)) bevestigen deze verbetering. Bij hogere drop rates blijft de algemene vorm van de cijfers beter behouden dan bij het startmodel. Echter,

het model reconstrueert grote gaten zoals bij drop rate 0.3 nog niet. Alleen kleine gaten worden correct hersteld.

Het gebruik van GIN-lagen zorgt ervoor dat het model beter in staat is om de structuur van de graaf te herkennen en maakt het model minder gevoelig voor ruis of ontbrekende verbindingen.

C. Iteratie 3: tweede GIN-laag toevoegen

Doel: Het vergroten van het contextbereik van de encoder, zodat knooprepresentaties meer context bevatten.

Beschrijving: In deze iteratie is een extra GINConv-laag toegevoegd aan de encoder. Hierdoor krijgt elke knoop niet alleen informatie van directe buren (1-hop), maar ook van 2-hop omgevingen. De projectielagen voor mu en logstd zijn ongewijzigd gebleven.

Onderbouwing: Cijfers zijn opgebouwd uit meerdere lokale structuren die vaak op afstand van elkaar liggen. Volgens Xu et al. bevat een GNN met k lagen informatie uit k -hop buren. Door een tweede GIN-laag toe te voegen, vergroot het model dus zijn contextbereik tot 2-hop, wat belangrijk is voor het herkennen van de volledige cijferstructuur [53].

Resultaten: Uit de loss-curve (Figuur 3D (a)) is te zien dat het model nog steeds snel convergeert. De F1-scores op de validatieset (Figuur 3D (c)) laten een kleine verbetering zien bij alle verschillende drop rates. De visuele reconstructies (Figuur 3D (c)) laten ook een verbetering zien. Ook wat grotere gaten worden gereconstrueerd, zoals bij de 7 met een drop rate van 0.3.

Het toevoegen van een extra GIN-laag geeft het model net wat meer context, wat de prestaties verbetert.

D. Iteratie 4: derde GIN-laag toevoegen

Doel: Onderzoeken of het vergroten van de diepte van het model verdere prestatieverbetering oplevert.

Beschrijving: De encoder is verder uitgebreid met een derde GINConv-laag, zodat het contextbereik wordt vergroot tot 3-hop. Dit zorgt er voor dat het model ook complexere structuren in de graaf mee kan nemen in de knooprepresentaties.

Onderbouwing: Door het toevoegen van een derde GIN-laag wordt het contextbereik van het model vergroot tot 3-hop, waardoor elke knoop informatie kan verzamelen uit een groter deel van de graaf. In de literatuur wordt aangeraden om voldoende lagen te gebruiken om globale structuur te kunnen vastleggen [53].

Resultaten: De validatie F1-scores (Figuur 4D (b)) zijn duidelijk verbeterd. Met name bij de precision is er een zichtbare verbetering te zien. De voorbeeld reconstructies (Figuur 4D (c)) laten dit ook zien. Er zijn minder extra foute verbindingen gemaakt, wat de reconstructie duidelijk houdt. Echter, het reconstrueren van grotere gaten blijft een lastig punt dat blijft terug komen.

E. Iteratie 5: bilineaire decoder

Doel: De decoder uitbreiden zodat het model kan leren welke dimensies van de embeddings relevant zijn voor het voorspellen van verbindingen.

Beschrijving: De decoder is vervangen door een bilineaire variant. In plaats van de standaard dot-decoder van VGAE.

Onderbouwing: De standaard dot-product decoder weegt alle dimensies even zwaar. Een bilineaire decoder leert welke dimensies belangrijk zijn voor het voorspellen van verbindingen. Hierdoor kan het model gerichter onderscheid maken tussen relevante en irrelevante knooppalen [54].

Resultaten: De F1-scores op de validatieset (Figuur 5D (b)) laten zien dat de beste threshold nu ligt tussen de 0.90 en 0.95 in plaats van 0.7 van de vorige iteraties. Dit betekent dat het model pas echt een verbinding voorspelt wanneer het model er relatief zeker van is. De bilineaire decoder lijkt dus beter te zijn in het onderscheiden van relevante en irrelevante verbindingen, dan de standaard dot decoder.

De visuele reconstructies (Figuur 5D (c)) zien er wel iets rommeliger uit. Er zijn meer extra foute verbindingen gemaakt, maar het model reconstrueert wel beter dan in iteratie 4

F. Iteratie 6: dropout

Doel: Regularisatie toevoegen om overfitting te minimaliseren en het model robuuster te maken.

Beschrijving: Dropout ($p=0.3$) is toegevoegd aan de MLP-lagen binnen elke GINConv.

Onderbouwing Dropout is een veelgebruikte techniek voor regularisatie. Er wordt aanbevolen om dropout toe te voegen binnen elke GINConv laag, om te voorkomen dat het model zich te veel richt op specifieke structuurelementen in de trainingsdata [53].

Resultaten: De F1-scores (Figuur 6D (b)) laten bij alle drop rates een verbetering zien in vergelijking met iteratie 5. Vooral bij de hogere drop outs is de F1-score duidelijk toegenomen. Ook is de precision bij deze hogere drop rates een stuk beter. De optimale thresholds blijven hoog (rond de 0.9), zoals in de vorige iteratie.

De voorbeeld reconstructies (Figuur 6D (c)) lijken wat consistentiever. Er lijken minder foute verbindingen gemaakt te zijn, vooral bij de wat hogere drop rates. Dit suggereert dat het toevoegen van dropout het model iets robuuster heeft gemaakt tegen ruis in de input.

G. Iteratie 7: batch normalisatie

Doel: Stabiliseren van de activaties tijdens training.

Beschrijving: Batch normalisatie is toegevoegd na iedere activatie in de GINConv-lagen. Dit zorgt ervoor dat de activatiewaarden binnen een batch vergelijkbaar blijven.

Onderbouwing Batch normalisatie draagt bij aan een stabielere training, vooral bij diepere modellen. Deze techniek is daarbij ook onderdeel van de aanbevolen GIN-implementatie in bestaande literatuur [53].

Resultaten: De F1-scores op de validatieset (Figuur 7D (b)) blijven voor het grootste gedeelte gelijk, alleen bij een drop rate van 0.4 is het model is zekerder geworden. Bij de reconstructie voorbeelden (Figuur 7D (c)) lijken wel wat meer negatieve verbindingen te hebben, maar reconstrueert verder wel voldoende. Ondanks dat er geen grote verbeteringen zijn verkregen, wordt er wel verder gewerkt met batch normalisatie, zoals aanbevolen in de literatuur.

H. Iteratie 8: jumping knowledge

Doel: Het behouden van informatie uit eerdere lagen om over-smoothing te vermijden.

Beschrijving: Met Jumping Knowledge (JK) (concat-modus) worden de outputs van alle GINConv-lagen samengevoegd voordat ze aan de projectielagen worden doorgegeven. op deze manier gaat informatie uit eerdere lagen niet verloren.

Onderbouwing: Zonder JK heeft het model enkel toegang tot de output van de laatste laag, wat bij diepere netwerken kan leiden tot verlies van lokale structuurinformatie. JK biedt een oplossing door alle lagen te combineren en zo bij te laten dragen aan de uiteindelijke representatie [53].

Resultaten: De F1-scores op de validatieset (Figuur 8D (b)) laten bij de lagere drop rates (0, 0.1 en 0.2) een lagere F1-score zien. Bij de wat hogere drop rates (0.25, 0.3 en 0.4) is de F1-score wel verbeterd, wat ook te zien is in de hogere recall.

De reconstructie voorbeelden laten zien dat het model echt reconstrueerd. Ook grotere gaten zoals bij de 8 bij drop rate 0.3 is dicht onderaan. De reconstructies zijn echter nog niet voldoende.

I. Iteratie 9: MLP-decoder

Doel: Onderzoeken of een niet-lineaire decoder helpt bij het maken van nauwkeurigere reconstructies, vooral in lastigere gevallen.

Beschrijving: De bilineaire decoder is vervangen door een MLP, waarbij de embeddings van twee knopen worden gecombineerd en in een klein netwerk worden gestopt. De MLP decoder zal vervolgens een linksscore voorspellen.

Onderbouwing: Een MLP-decoder kan beter omgaan met complexere structuren dan een lineaire decoder. Door de embeddings van twee knopen te combineren en daar een MLP op toe te passen, leert het model zelf welke patronen belangrijk zijn. Zo kan het model bepalen of er een verbinding tussen twee knopen hoort te zijn [55].

Resultaten: De F1-scores (Figuur 9D (b)) laten bij bijna alle drop rates een verbetering zien ten opzichte van iteratie 8. Ook in de reconstructie voorbeelden (Figuur 9D (c)) is dit terug te zien. De reconstructies ogen over het algemeen iets netter, completer en met minder ruis. Toch zijn er wel nog relatief veel fouten aanwezig, vooral bij de hogere drop rates.

J. Iteratie 10: contrastive learning

Doel: Versterken van de latente ruimte door middel van contrastive learning, zodat het model leert welke grafen bij elkaar horen.

Beschrijving: Aan de loss is een extra contrastieve loss term toegevoegd. Bij elke graaf worden twee corrupte versies gemaakt (views). Deze views worden geëncodeerd, waarbij er daarna een NT-Xent loss wordt toegepast op de globale embedding (verkregen via global mean pooling).

Onderbouwing: Contrastive learning dwingt het model om globale structuur te herkennen, ook als er lokaal informatie ontbreekt. Door views van dezelfde graaf dichter bij elkaar

te brengen in de latente ruimte, leert het model robuustere representaties [56].

Bij het uitvoeren van de iteratie bleek dat het model sterk reageerde op aanpassingen bij de hyperparameters. Daarom zijn de vier hyperparameters die direct invloed hebben om de contrastive loss getuned. In totaal zijn er meer dan 100 combinaties getest:

- Temperature: bepaalt hoe scherp het contrastive loss-signal reageert op afstandsverschillen tussen embeddings. Lagere waarden maken het model gevoeliger voor kleine verschillen.
 - 0.1, 0.3, 0.5, 0.7
- Contrastive weight: het gewicht dat bepaalt hoe zwaar de contrastive loss meeweegt ten opzichte van de reconstructie- en KL-loss.
 - 0.1, 0.2, 0.3
- Drop rate view 1: het percentage randen dat wordt verwijderd bij het maken van de eerste augmentatie (view 1). Deze view wordt gebruikt voor zowel reconstructie als contrastive loss.
 - 0.1, 0.25, 0.3
- Drop rate view 2: het percentage randen dat wordt verwijderd bij het maken van de tweede augmentatie (view 2). Deze wordt alleen gebruikt voor de contrastive loss.
 - 0.1, 0.25, 0.3

Resultaten: Tijdens het testen van de 108 verschillende hyperparametercombinaties (zie Tabel IID) bleek dat contrastive learning in deze situatie niet goed werkt. De beste resultaten werden behaald bij lage drop rates voor zowel view 1 als view 2 (beide 0.1), waarbij het model alleen bij drop rates van 0 en 0.1 relatief hoge F1-scores haalde.

Bij hogere drop rates namen de prestaties af. Waarschijnlijk komt dit doordat het model tijdens training alleen views zag met lichte corruptie (drop rate 0.1), terwijl eerdere iteraties traande met 0.25. Hierdoor mist het model de robuustheid om zwaardere corruptie te reconstrueren.

Deze resultaten laten zien dat contrastive learning in deze opzet alleen werkt bij relatief schone input. Bij sterkere corruptie verslechteren de prestaties, waardoor besloten is om deze aanpak **niet** verder te volgen.

K. Iteratie 11: hard negative sampling

Doel: Verminderen van foute verbindingen door het model te trainen met moeilijkere negatieve voorbeelden.

Beschrijving: In plaats van negatieve voorbeelden willekeurig te selecteren (zoals gebruikelijk in VGAE-implementaties), worden nu verbindingen geselecteerd waarvan de embeddings op elkaar lijken, maar geen verbinding hebben. Na het encoderen van een gecorrumpeerde graaf berekent het model voor elke verbinding een score op basis van het inproduct van hun embeddings. Vervolgens worden de verbindingen met de hoogste score geselecteerd, als deze geen bestaande verbinding vormen. De 'moeilijke' negeatieve voorbeelden lijken structureel op echte verbindingen, maar zijn dat dus niet.

Onderbouwing: Bij standaard random negative sampling zijn de negatieve voorbeelden vaak te eenvoudig (bijvoorbeeld van links onder naar recht boven), waardoor het model weinig leert over subtile verschillen tussen echte en foutieve verbindingen. Door bewust 'moeilijke' negatieve voorbeelden te gebruiken, leert het model beter onderscheid maken [57].

Resultaten:

De F1-scores (Figuur 10D (b)) zijn over het algemeen iets lager dan in de vorige iteratie 9, met uitzondering van drop rate 0.25 en 0.4, waar een kleine verbetering zichtbaar is. De recall is echter opvallend hoger bij vooral de hogere drop rates, wat aangeeft dat het model meer echte verbindingen weet te reconstrueren.

In eerdere iteraties werd een threshold van 0.85 gekozen om een balans te vinden tussen voldoende gereconstrueerde randen en het vermijden van ruis. Uit deze resultaten blijkt dat een threshold van 0.95 nu beter werkt. Bij deze waarde worden voldoende randen hersteld zonder teveel gaten over te houden. Het gebruiken van een threshold van 0.95 bij het model van iteratie 9 zou resulteren in incompletere reconstructies, omdat de recall daar een stuk lager ligt.

De reconstructievoorbeelden (Figuur 10D (c)) bevestigen dit. Hoewel er nog foutieve verbindingen aanwezig zijn, is hun invloed op de algemene vorm van het cijfer kleiner. De globale structuur blijft beter behouden dan in eerdere iteraties. Bijvoorbeeld in iteratie 9, leidde foutieve verbindingen soms tot grote vervormingen (bijvoorbeeld bij het cijfer 7). Dat effect is hier duidelijk minder zichtbaar.

L. Iteratie 12:

Doel: Het model nauwkeuriger maken in het herkennen van echter verbindingen, door het model te trainen met een groter aantal negatieve voorbeelden.

Beschrijving: Experimenteren met de verhouding tussen negatieve en positieve voorbeelden. In iteratie 11 werd eerder een 1:1 verhouding toegepast. In deze iteratie zijn meerdere hogere verhoudingen getest, van 2:1 tot en met 10:1. Omdat het aantal negatieve voorbeelden hierdoor veel groter wordt en zal gaan overheersen, is de lossfunctie aangepast. Zonder aanpassing zouden de negatieve voorbeelden gaan domineren, wat ervoor kan zorgen dat het model leert om vooral geen verbindingen te voorspellen. Door een gewogen lossfunctie toe te passen, waarbij positieve voorbeelden zwaarder meewegen, blijft hun invloed voldoende in het leerproces.

Onderbouwing Een groter aantal negatieve voorbeelden sluit beter aan bij de werkelijkheid, waarin de meeste knooppalen geen verbinding hebben. Door meer negatieve voorbeelden te gebruiken, sluit de training beter aan bij deze onbalans. Een gewogen loss voorkomt dat het model te terughoudend wordt.

Resultaten: De resultaten van de verschillende verhoudingen zijn weergegeven in Tabel ID. Zowel de 3:1 als de 6:1 verhouding gaven de hoogste F1-scores, maar er is uiteindelijk gekozen voor 3:1 omdat deze verhouding net iets beter scoorde bij hogere drop rates.

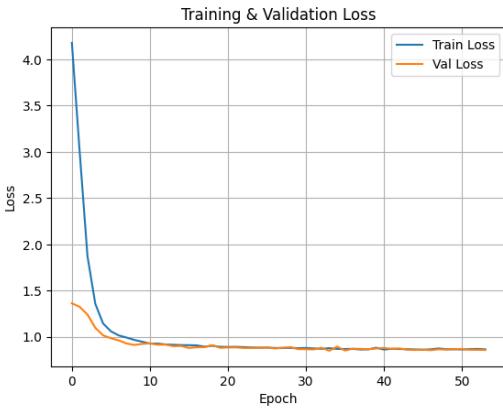
De F1-scores op de validatieset (Figuur 11D (b)) zijn ten opzichte van de vorige iteratie iets verbeterd. In de recon-

structievoorbeelden (Figuur 11D (c)) is te zien dat er iets minder foutieve verbindingen zijn voorspelt. Vooral bij het cijfer 8 ziet de reconstructie er iets netter uit. Er blijven nog wel negatieve verbindingen aanwezig, maar deze versturen de algehele structuur van het cijfer minimaal.

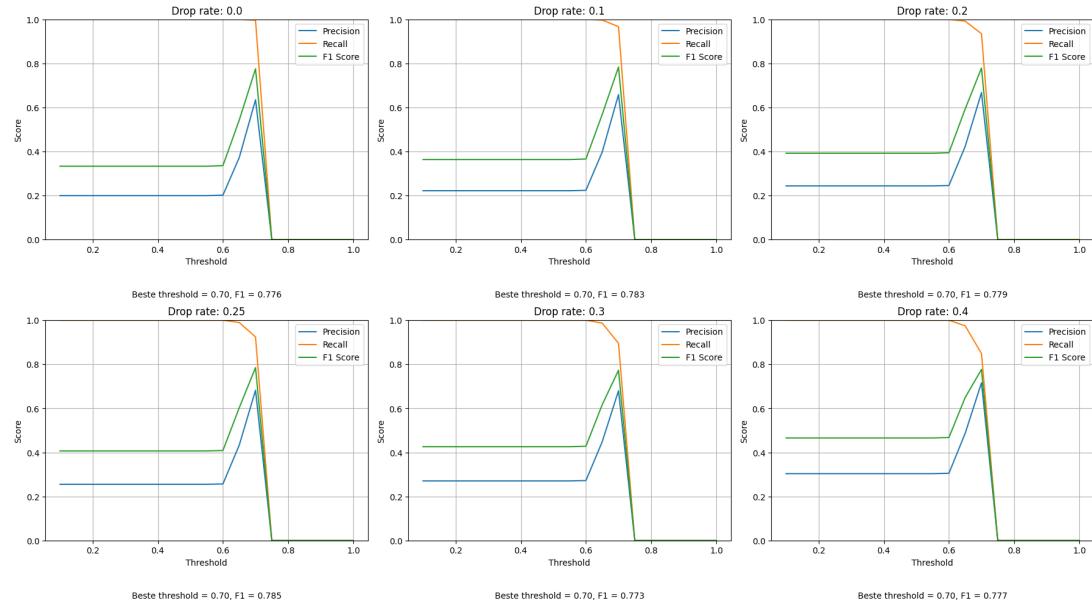
Ratio	F1-score					
	0.0	0.1	0.2	0.25	0.3	0.4
2:1	0,908	0,884	0,859	0,848	0,833	0,809
3:1	0,918	0,887	0,865	0,853	0,840	0,820
4:1	0,903	0,879	0,862	0,851	0,839	0,824
5:1	0,900	0,871	0,852	0,845	0,832	0,816
6:1	0,914	0,890	0,869	0,854	0,836	0,813
7:1	0,896	0,871	0,852	0,842	0,837	0,807
8:1	0,917	0,894	0,869	0,858	0,858	0,812
9:1	0,915	0,886	0,859	0,852	0,835	0,809
10:1	0,924	0,887	0,864	0,847	0,833	0,809

Tabel ID: F1-scores voor verschillende verhoudingen van negatieve voorbeelden

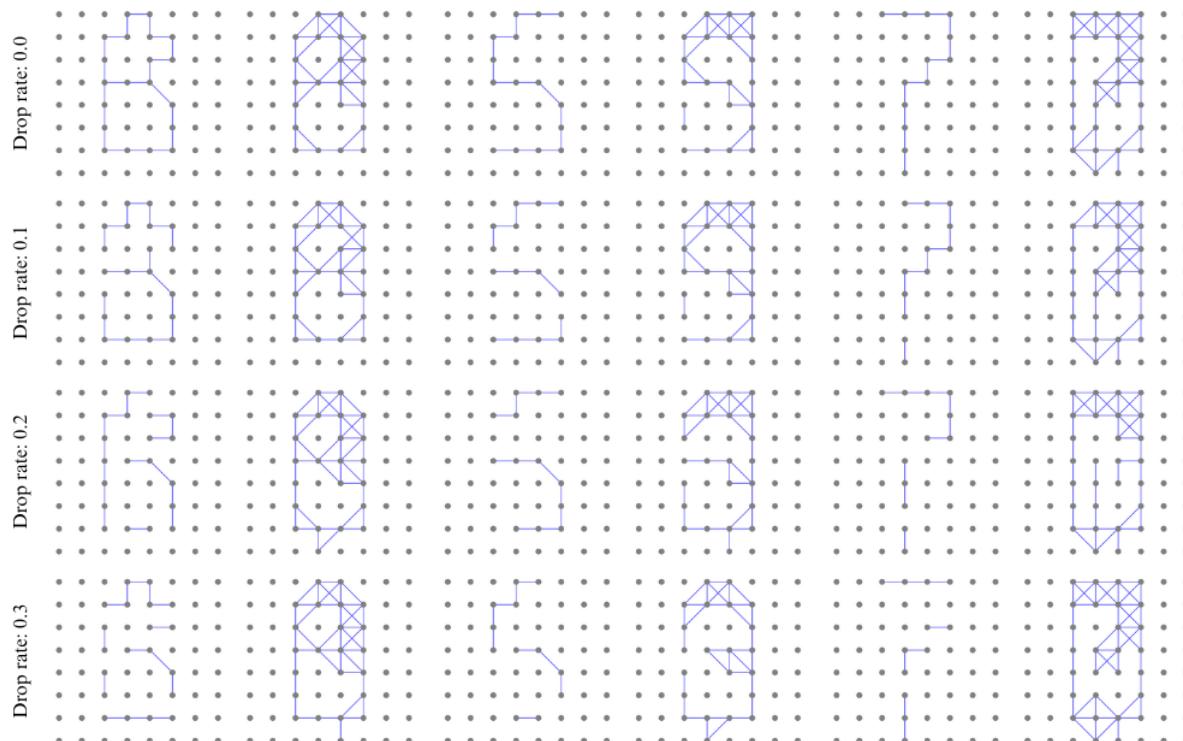
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

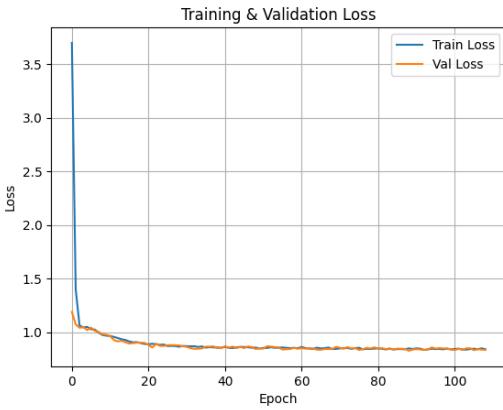


(c) Reconstructie voorbeelden over verschillende drop rates

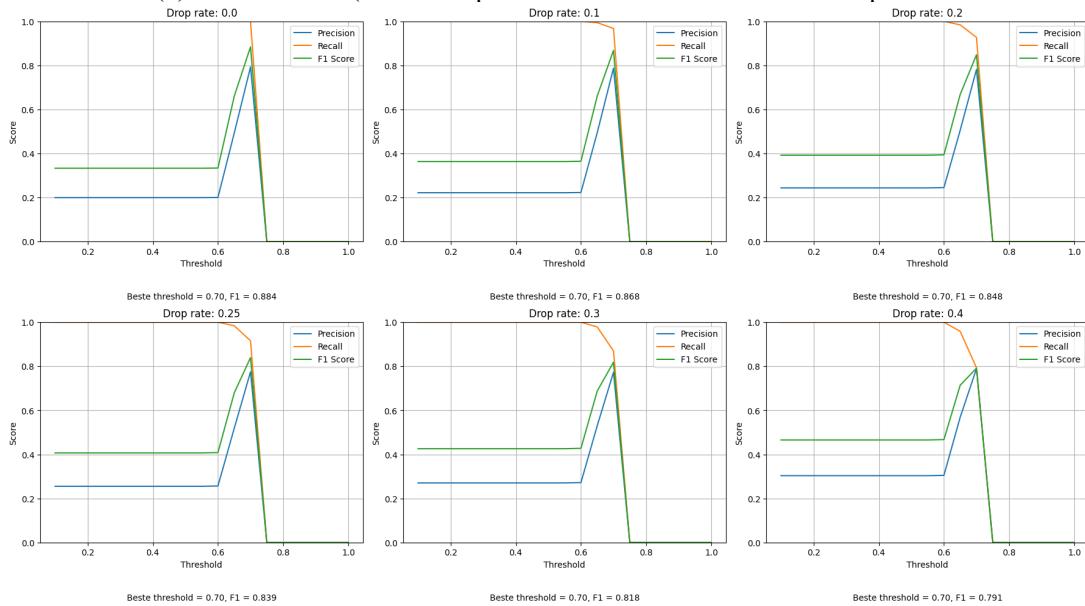


Figuur 1D: Visualisaties iteratie 1: denoising VGAE (startmodel)

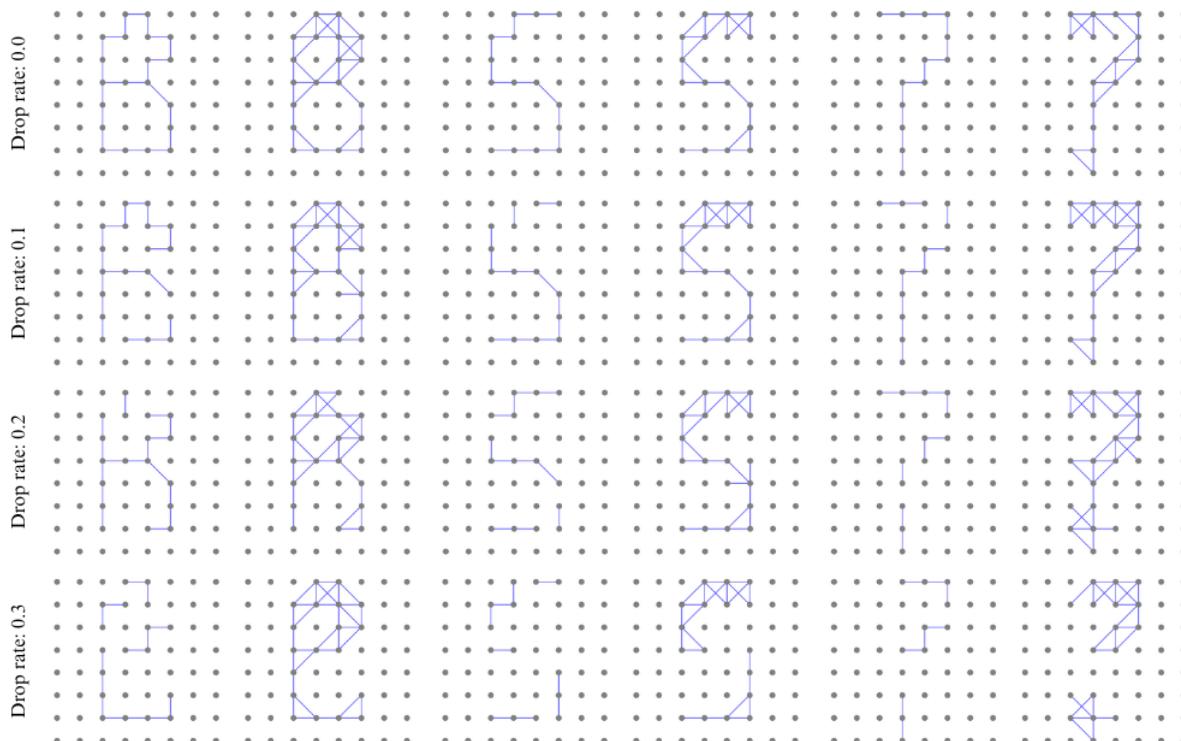
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

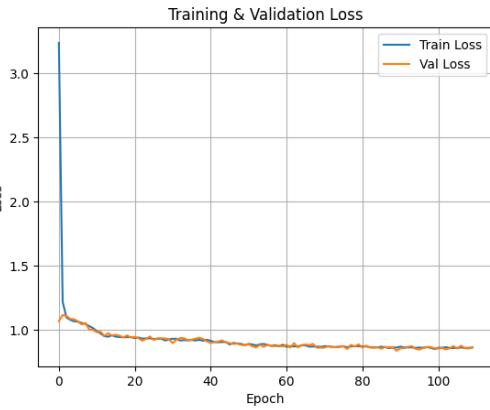


(c) Reconstructie voorbeelden over verschillende drop rates

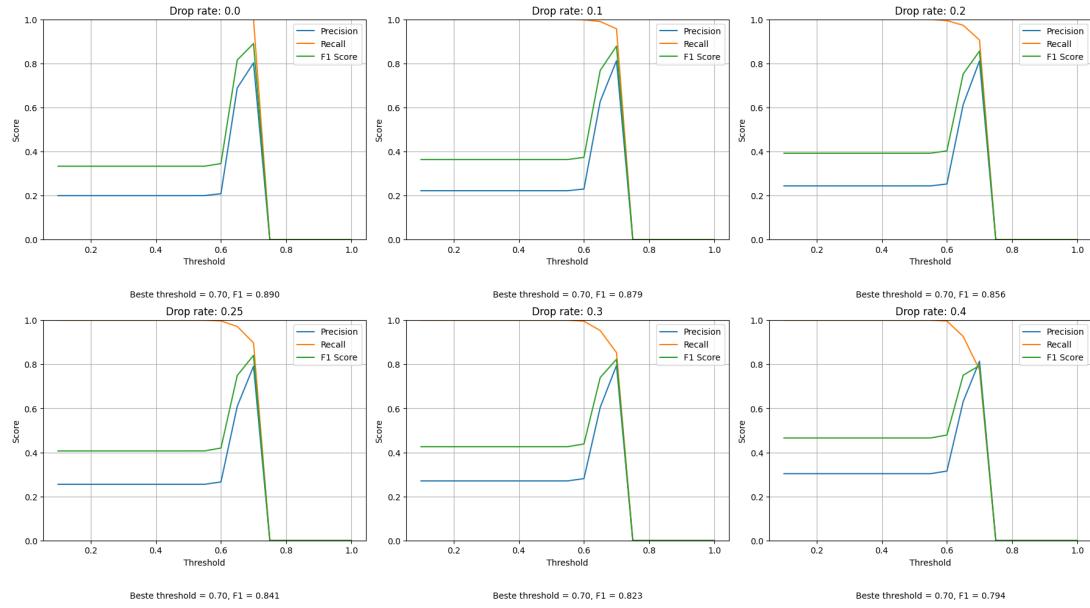


Figuur 2D: Visualisaties iteratie 2: GIN-lagen gebruiken

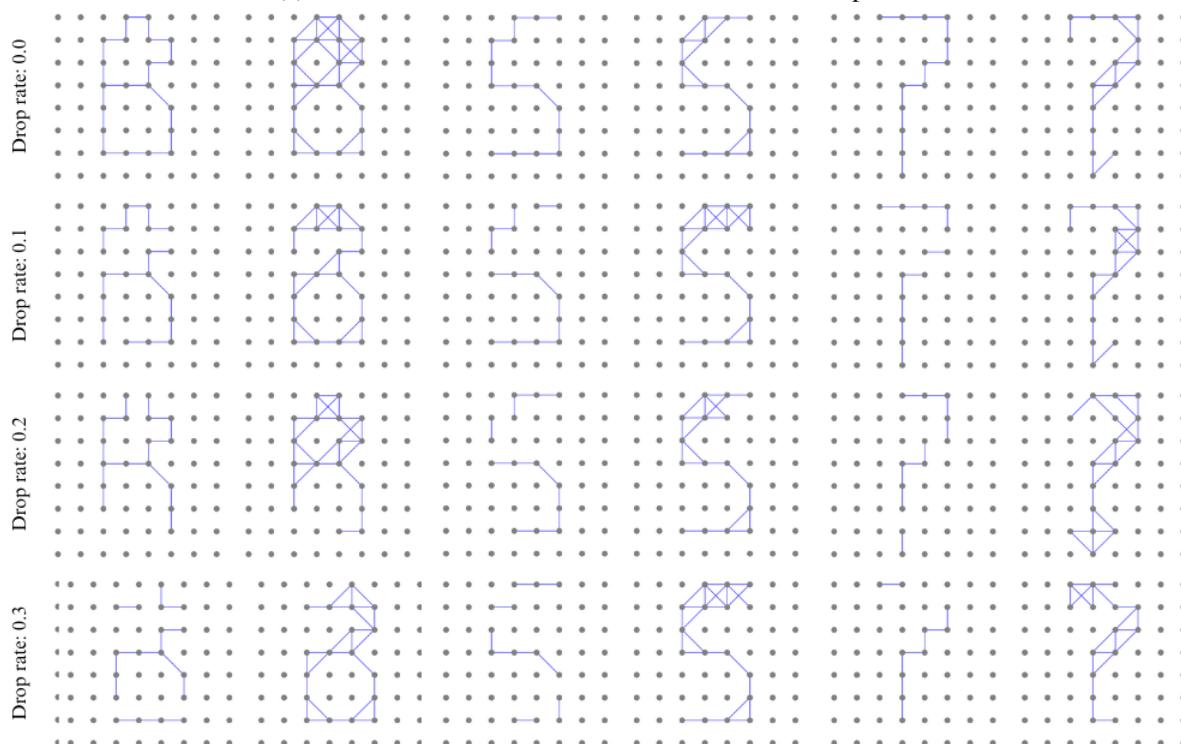
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

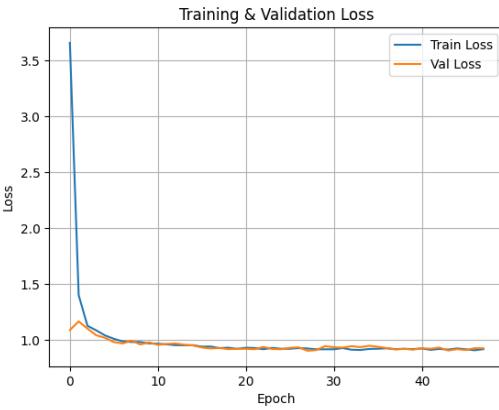


(c) Reconstructie voorbeelden over verschillende drop rates

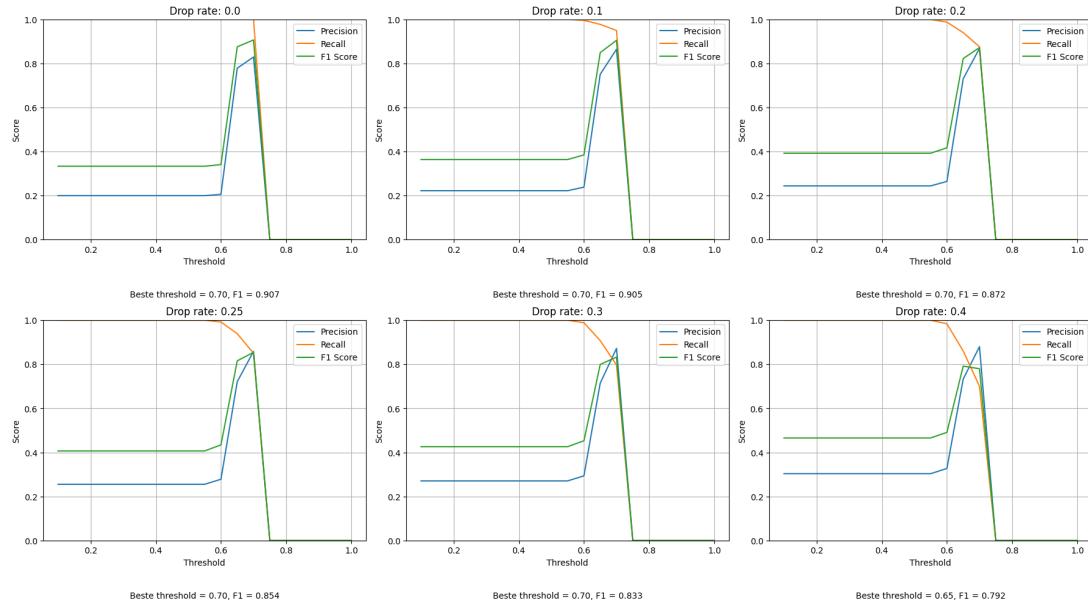


Figuur 3D: Visualisaties iteratie 3: tweede GIN-laag toevoegen

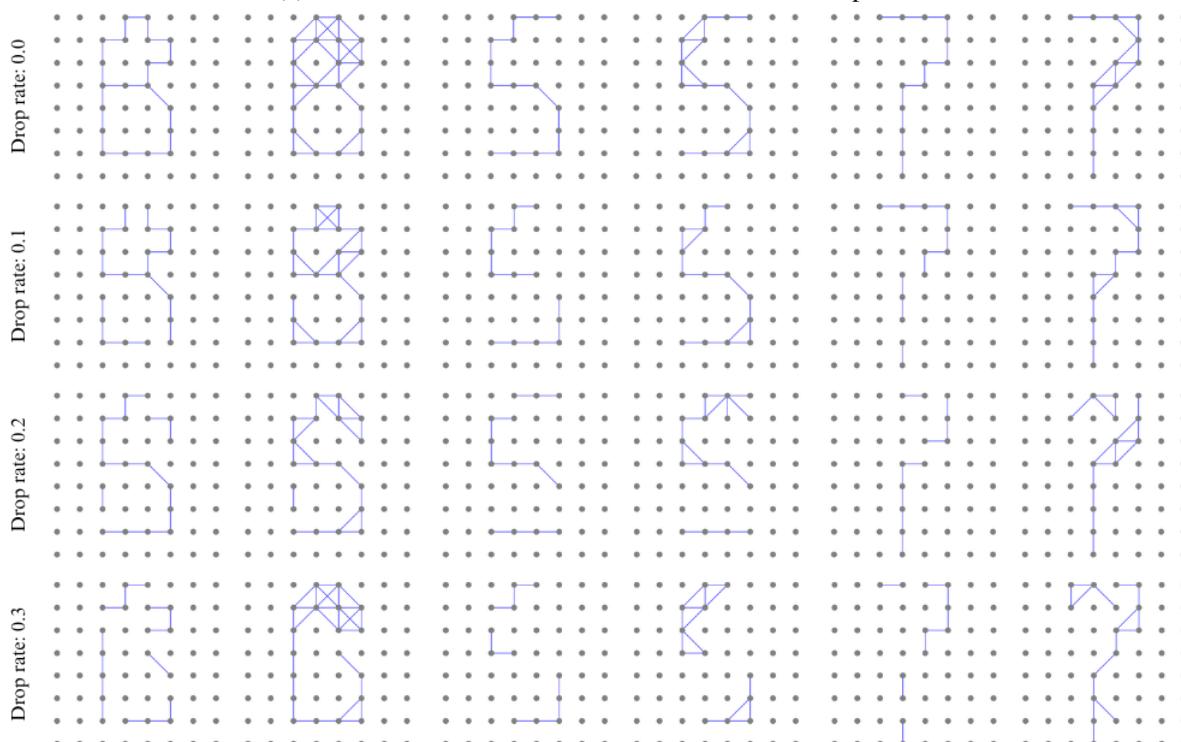
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

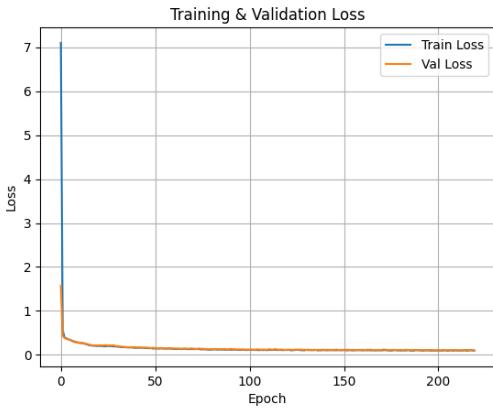


(c) Reconstructie voorbeelden over verschillende drop rates

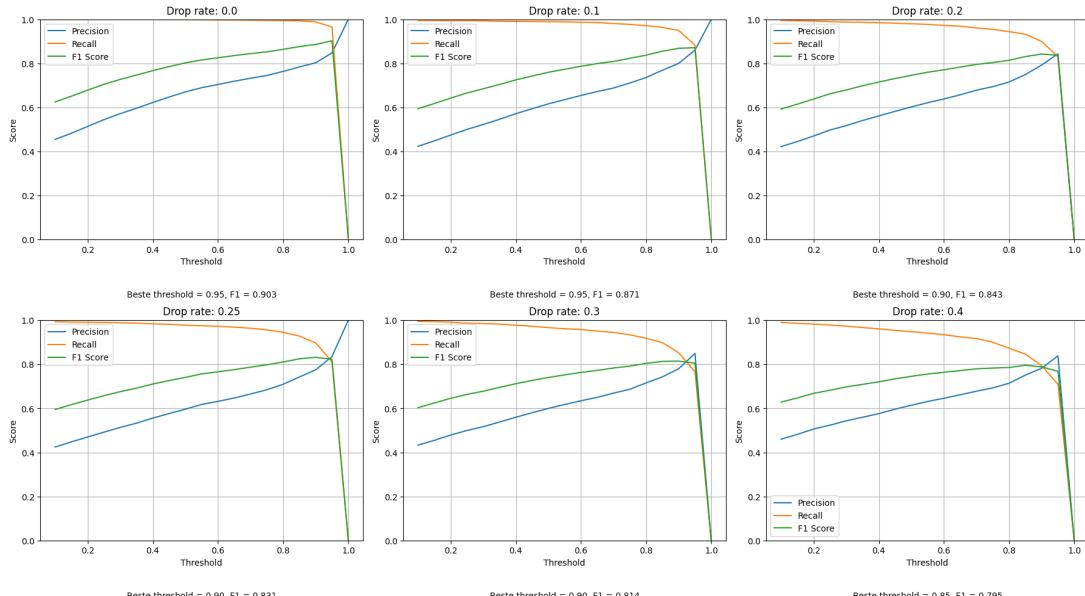


Figuur 4D: Visualisaties iteratie 4: derde GIN-laag toevoegen

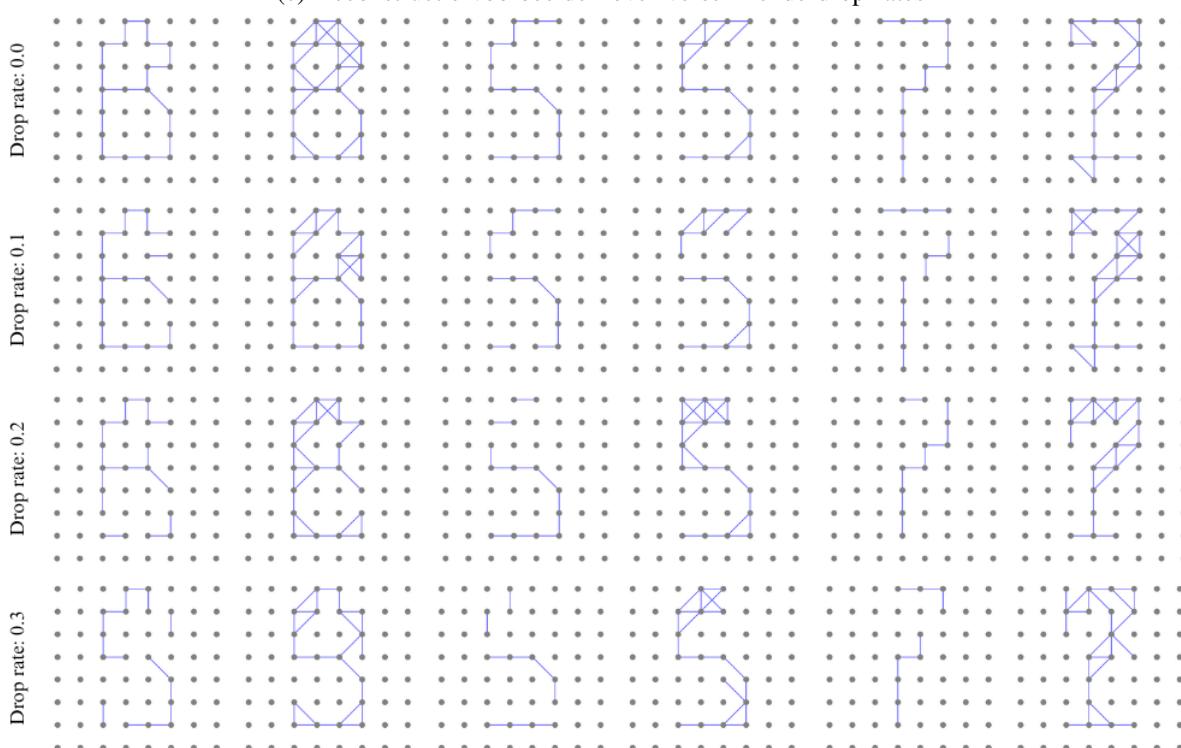
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

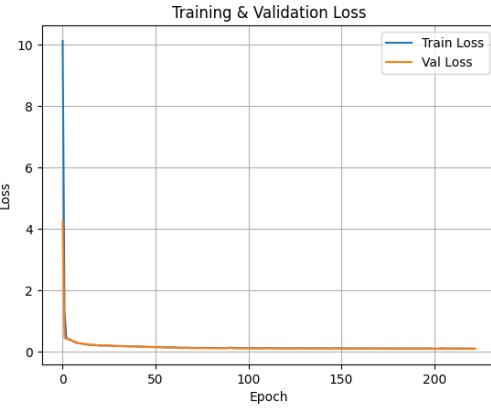


(c) Reconstructie voorbeelden over verschillende drop rates

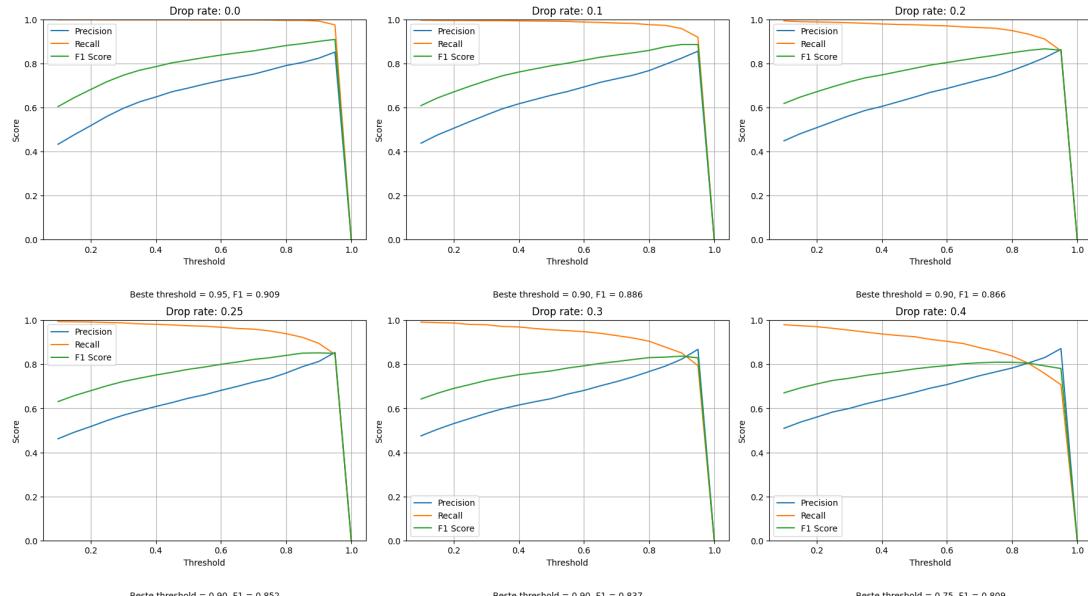


Figuur 5D: Visualisaties iteratie 5: bilineaire decoder

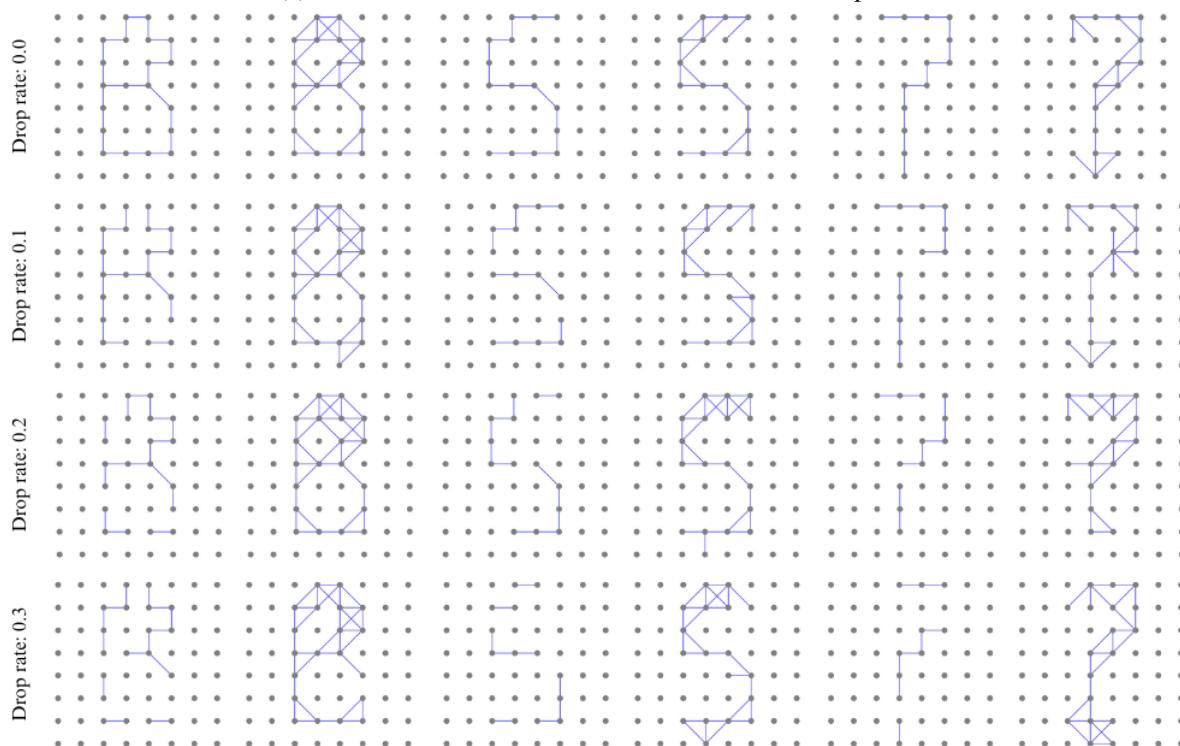
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)



(c) Reconstructie voorbeelden over verschillende drop rates

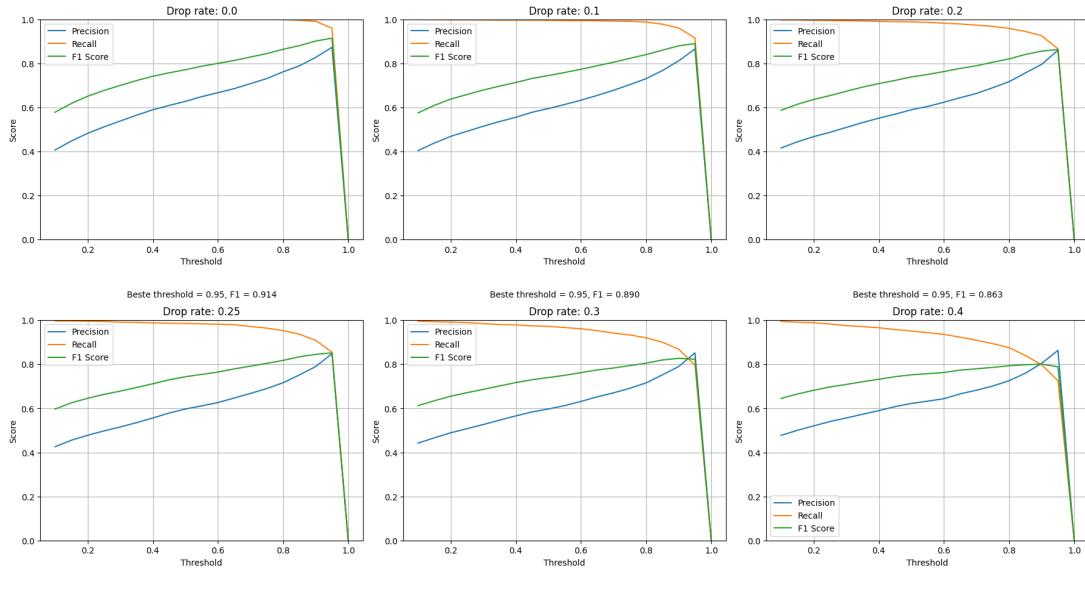


Figuur 6D: Visualisaties iteratie 6: dropout

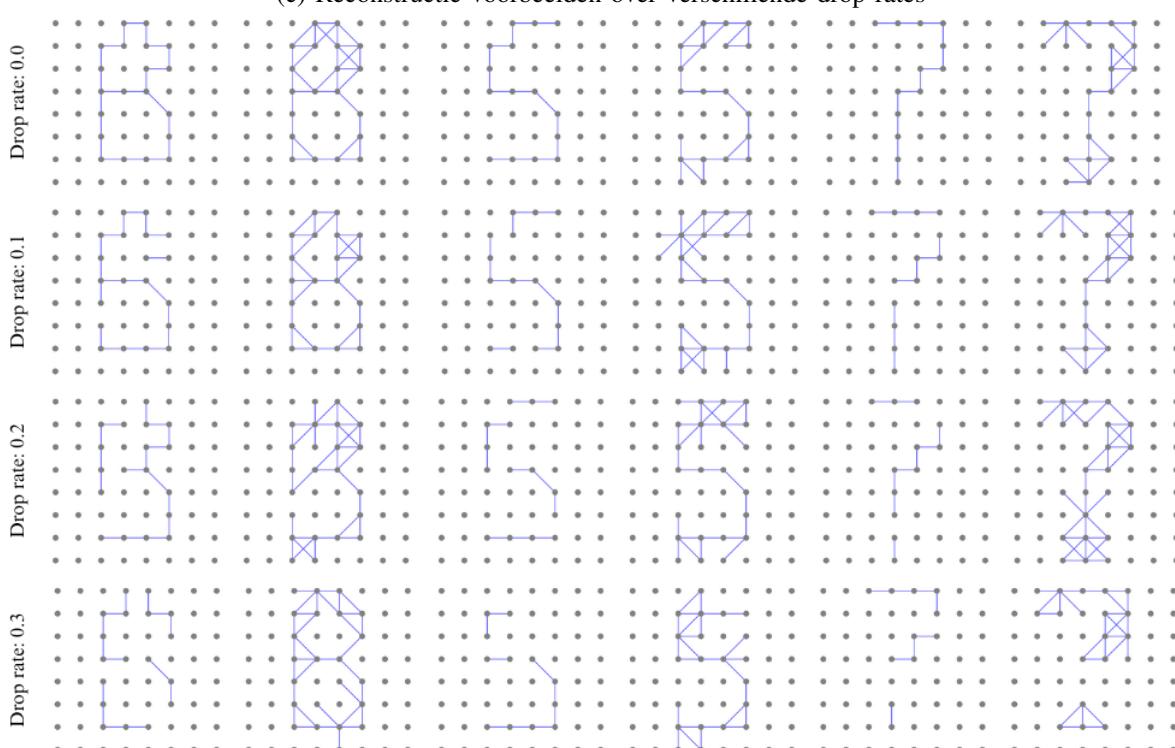
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

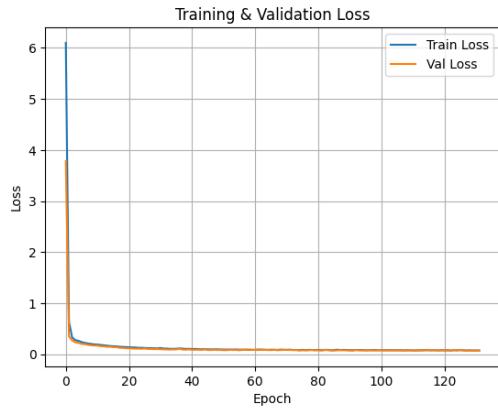


(c) Reconstructie voorbeelden over verschillende drop rates

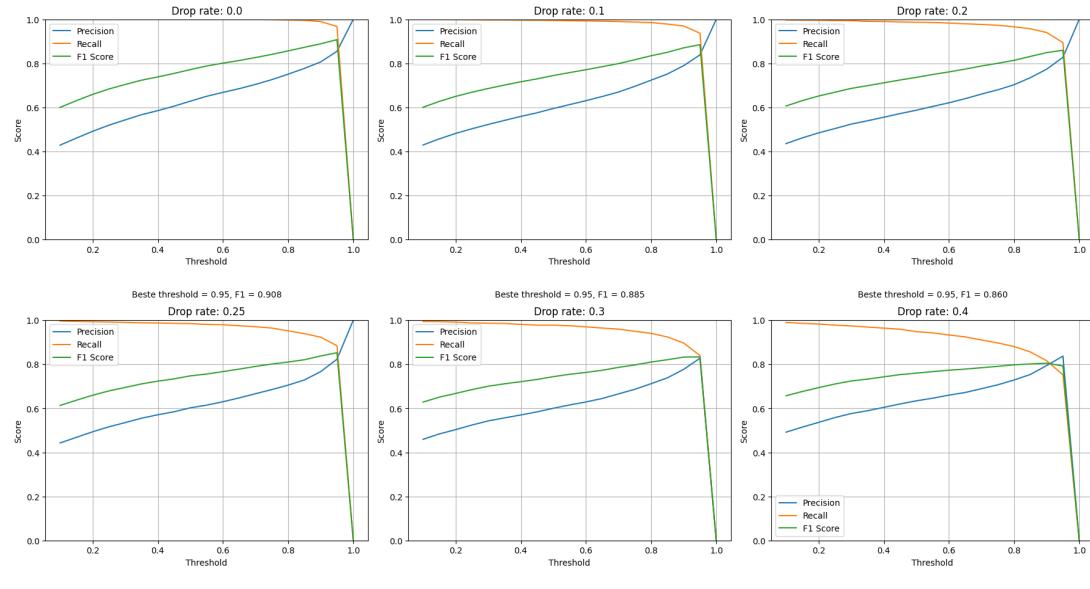


Figuur 7D: Visualisaties iteratie 7: batch normalisatie

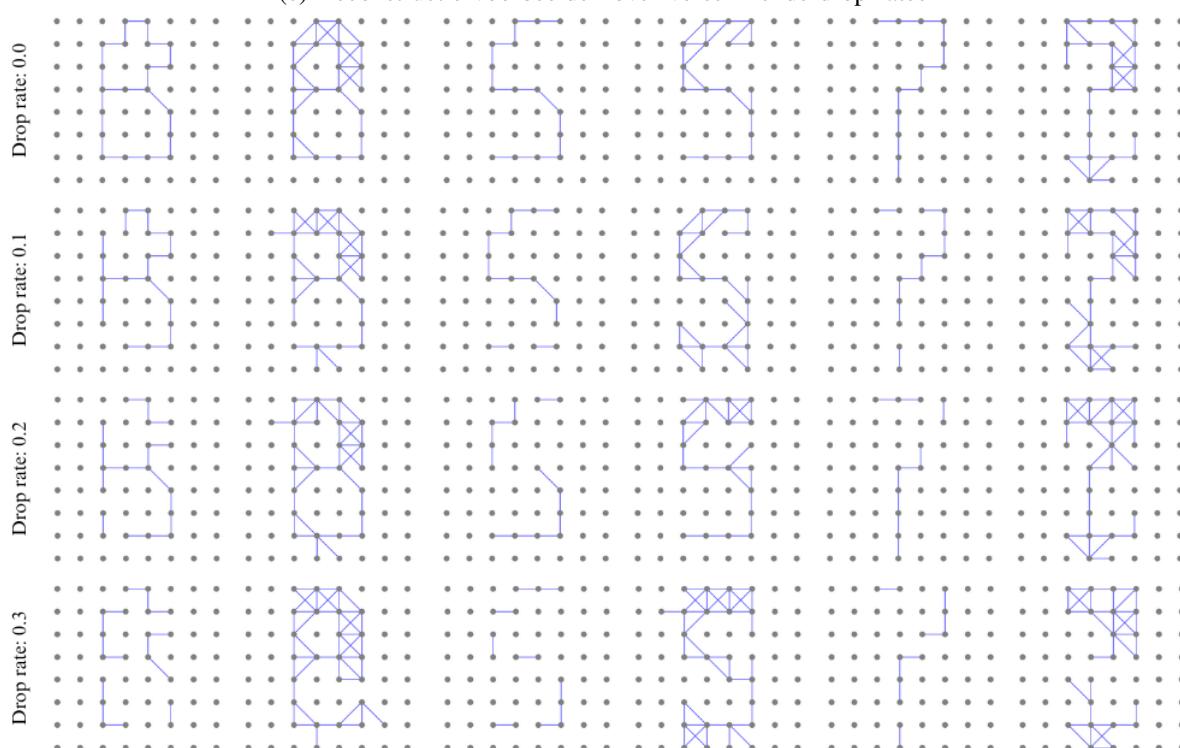
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

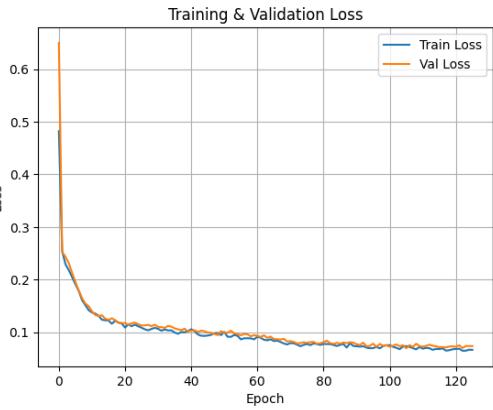


(c) Reconstructie voorbeelden over verschillende drop rates

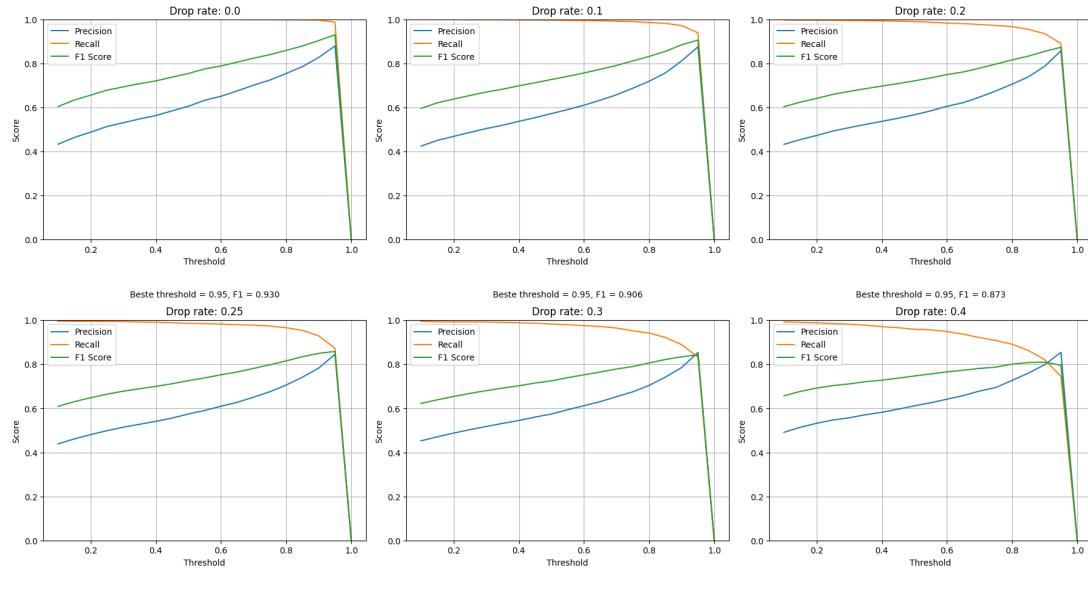


Figuur 8D: Visualisaties iteratie 8: jumping knowledge

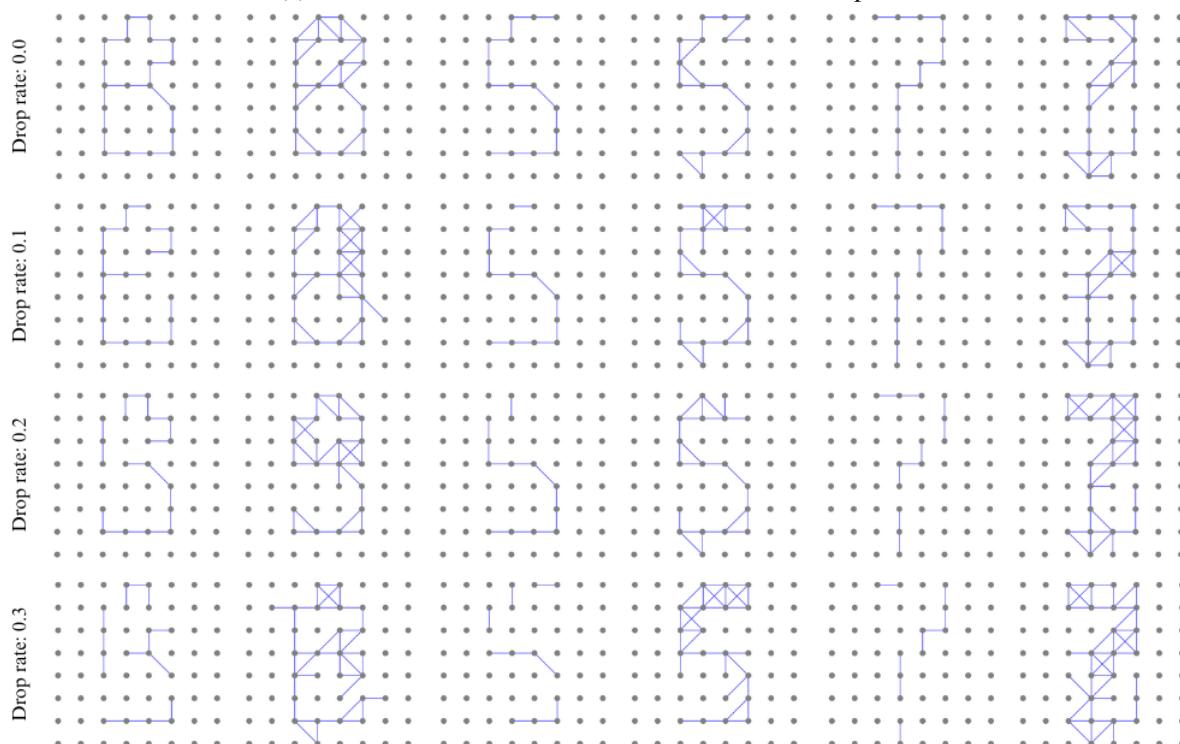
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)



(c) Reconstructie voorbeelden over verschillende drop rates

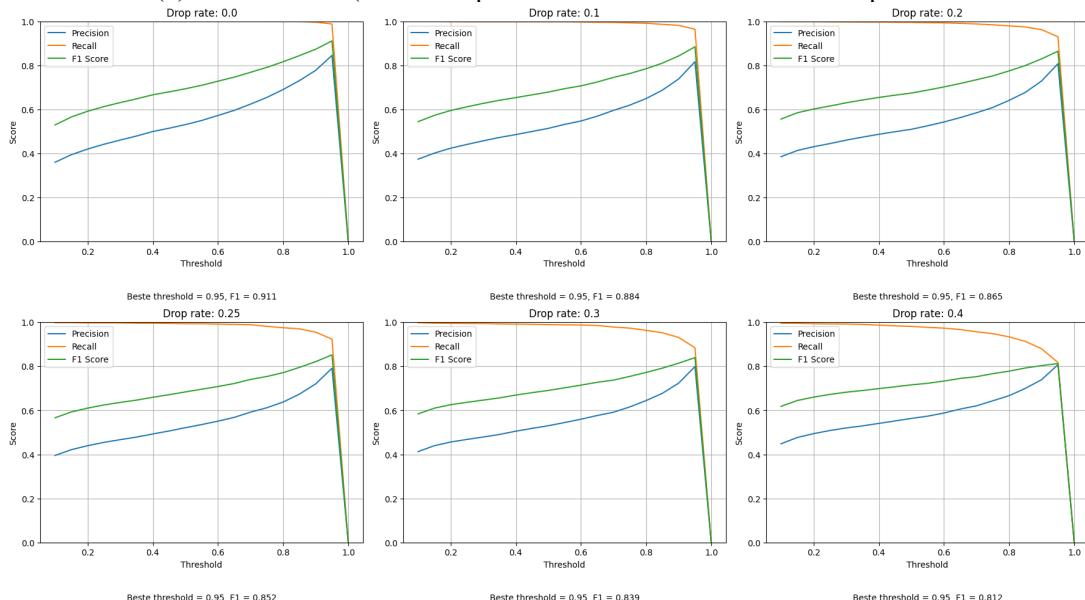


Figuur 9D: Visualisaties iteratie 9: MLP-decoder

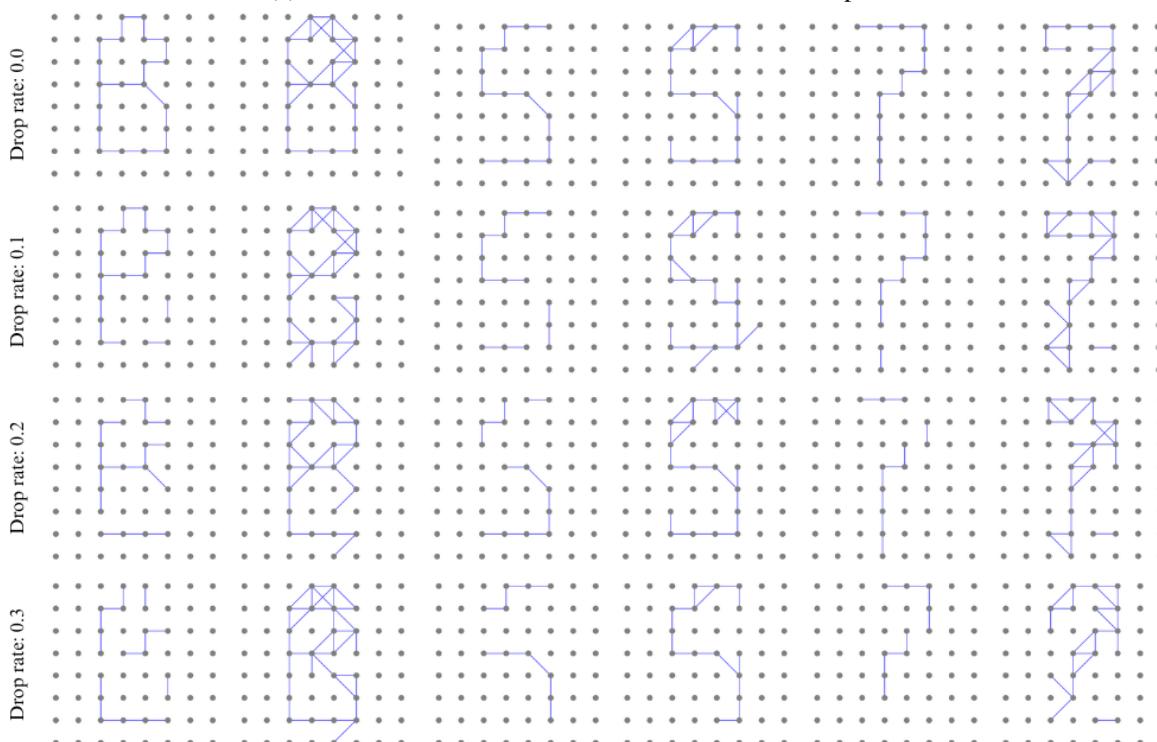
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)

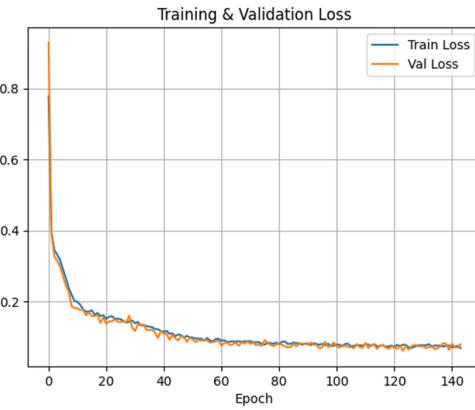


(c) Reconstructie voorbeelden over verschillende drop rates

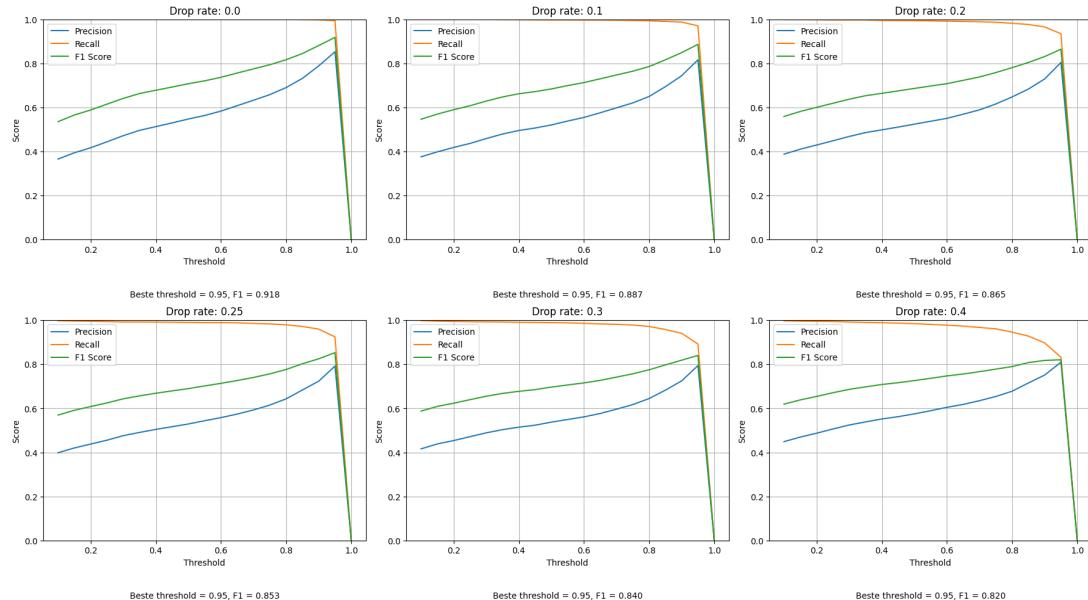


Figuur 10D: Visualisaties iteratie 11: hard negative sampling

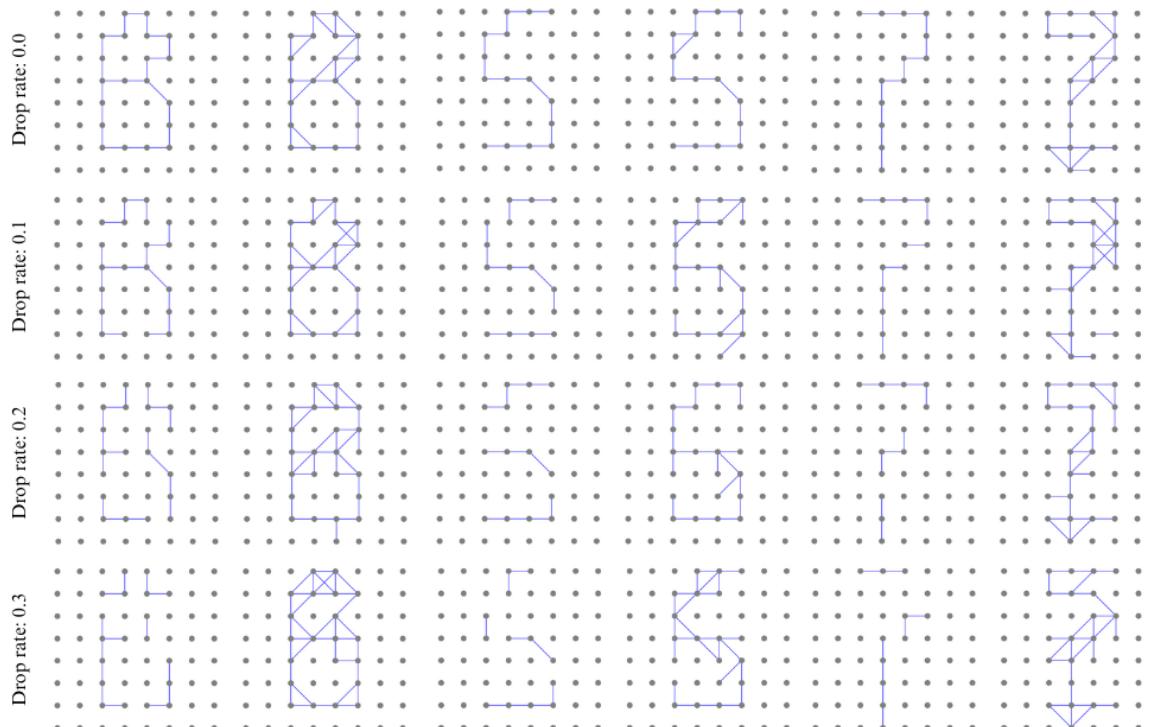
(a) Invoer (Loss curve van training)



(b) Reconstructie (F1-score op validatieset over verschillende drop rates)



(c) Reconstructie voorbeelden over verschillende drop rates



Figuur 11D: Visualisaties iteratie 12: hard negative sampling met 3:1 ratio

