FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Interferentiedetectie in sub-GHz IoT Netwerken op Constrained Devices

Francis Begyn
Studentennummer: 01303049

Promotoren: prof. dr. ir. Eli De Poorter, dr. Adnan Shahid
Begeleider: ir. Jaron Fontaine

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

Academiejaar 2019-2020

UNIVERSITEIT
GENT

I Copyright

II Voorwoord

Mijn promotor prof. dr. ir. Eli De Poorter en co-promotor dr. Adnan Shahid wil ik graag bedanken voor de mogelijkheid om mijn masterproef af te leggen in Gent, voor de begeleiding en de constructieve feedback van de resultaten van deze masterproef.

Ik zou ook graag mijn begeleider Jaron Fontaine willen bedanken voor de goede en flexibele begeleiding. Ik heb enorm veel bijgeleerd op praktisch en theoretisch vlak. Altijd beantwoorden jullie mijn vragen vol enthousiasme en toewijding.

Graag zou ik ook alle thesisstudenten en doctoraatstudenten willen bedanken voor de toffe sfeer in het labo zelf. Ten slotte zou ik graag mijn familie en vrienden willen bedanken voor alle steun tijdens deze uitdagende periode om deze masterproef tot een zo goed mogelijk einde te brengen.

Contents

Lijst van tabellen

Lijst van figuren

III Gebruikte afkortingen

ASK Amplitude Shift Keying. 18

ATC Automatische Technologie Classificatie. 2, 18

BPSK Binary Phase Shift Keying. 18

CNN Convolutional Neural Network. IV, IX, 3, 10, 11, 13, 15–17, 26, 28, 33, 34, 55

CSS Chrip Spread Spectrum. 21

D-BPSK Differential Binary Phase-Shift Keying. 20

DNN Deep Neural Network. 10, 11, 17

FFT Fast Fourier Transform. 12

FPS Frames Per Second. 10

GFSK Gaussian Frequency-Shift Keying. 20

GP-GPU General Purpose Graphical Processing Unit. 15

IoT Internet of Things. III, VI, 1–6, 9, 18, 22, 55

ISM Industrial, Scientific and Medical. 9, 16, 18, 20

LoS Line of Sight. 18

LPWAN Low Power Wide Area Network. IV, IX, 2, 53, 56

M2M Machine-to-Machine. 4

MAC Medium Access Control. 11

ML Machine Learning. III, 5, 10, 11, 17, 22, 23, 53, 55, 56

MST Multi Stage Training. 11

Q-PSK Quadrature Phase Shift Keying. 18

RFID Radio Frequency Identification. 4

RSS Relative Signal Strength. 12

RSSI Received Signal Strength Indication. VI, 11, 12, 22, 26, 28, 33–38, 42–44, 49

SDR Software Defined Radio. V, 9–11, 21–24, 44, 55

SF Spreading Factor. 21

SL Supervised Learning. 16

SNR Signal-to-Noise Ratio. 13, 16, 17

SSD Single Shot Detector. 10

SVM Support Vector Machine. 11

WII Wireless Interference Identification. 13

Interference detection in sub-GHz IoT networks on constrained devices

Francis Begyn, prof. dr. ir. Eli De Poorter, dr. Adnan Shadid & ir. Jaron Fontaine

***Abstract*—The upcoming technology known as Internet of Things (IoT) will put a lot of pressure on the current Low-Power Wide Area Network (LPWAN) technologies. As more and more devices start connecting to the cloud, the need for better management and a way to select good technologies depending on the environment has grown. Current research in "smart networks" utilise specialised and expensive hardware. This makes the selection and management of these LPWANs highly unavailable. This paper proposes to use low-end embedded devices and off-the-shelf Software Defined Radio (SDR) USB dongles. It seeks to answer 2 questions: 1.) is it possible to use low-end hardware for interferencedetection 2.) how well does the low-end hardware perform. 3 different complexity levels of Convolutional Neural Network (CNN) are tested on 2 different platforms: Dell XPS 13 9360, Raspberry Pi 3B and Raspberry Pi 4. The proposed CNNs have good performance on the Dell XPS 13 9360, and they have acceptable performance on the Raspberry Pi 3B and Raspberry Pi 4. While the performance is acceptable, there are limitations that are to be kept in mind: 1.) while the performance is acceptable, we can also see that the pre-processing has an impact on the performance and 2.) each application should be evaluated if the performance lays within the constrains of the application. This shows that the constrained devices can offer a realistic option for practical applications.**

I. Introduction

The current trend in IoT is that more and more devices will connect to the cloud through the means of LPWANs. To handle this growth, proper management of these networks will be required.

Multiple technologies are available for IoT use. These technologies offer solutions to the problems that current technologies have when connecting to a large number of devices or in challenging environments [1], [2].

Miller *et al.* and Bitar *et al.* have shown that the increasing demand of IoT technology will require flexible networks that can adjust their configuration depending on the environment [3], [4]. To create these flexible networks Machine Learning (ML) will play an important role. With the use of ML, it opens avenues that allow us to create dynamic, learning and growing networks that can adjust to a rapidly changing environment. The use of Received Signal Strength Indication (RSSI) and IQ-values in combination with ML techniques have been researched and proven to offer enough data to make distinctions between different modulations and technologies [3]–[7]. The use of RSSI has been shown to work excellent by Zacharias *et al.* to recognise certain technologies, even in challenging environments like office floors [6].

Aside from the use of CNNs, Supervised Learning (SL) has also shown to be a good ML method to classify technologies by Grimaldi *et al.* . Their research also shows that the selection of the data type and Signal-to-Noise Ratio (SNR) values, have a large impact on the performance of the CNN[8].

Embedded platforms are up and coming in the world of ML. Recent steps forward in the computational platforms that these devices have, make them a viable option for ML. This doesn't mean that they don't come with a trade off, there is still a performance hit when using these devices. The real benefit comes from the power savings that these devices can offer. Tests have shown that it is possible to reach 65% of the performance of the ML on non-constrained devices, with only 1.5% of the power usage [9].

II. Implementation

A variety of SDR dongles and constrained devices have been considered. Eventually the following selection was made.

A. SDR dongle

The dongles (Table I) all have an 8-bit resolution. All dongles except the HackRF One are based upon the DVB-RT chipset, RTL2832U, so it is expected that the properties of the dongles too closely resemble each other and only be dependant on the implementation of the chip in the design.

TABLE I: Properties of SDR dongles

Device	Samplerate	Bandwidth	Frequency range	Price
RTL-SDRv3	2MSPS	2.4MHz	500kHz-1766MHz	21.95 USD
NE NESDR SMArt SDR		2.4MHz	25MHz - 1750MHz	23.95 USD
NE NESDR Mini SDR and DVB-T			25MHz - 1750MHz	18.95 USD
HackRF One	20 MSPS	20 MHz	1MHz - 6000MHz	284.95 USD

The choice for this paper is the RTL-SDRv3. Mainly due to large community that can offer support, the popularity of the device, large availability and specifications.

B. constrained device

TABLE II: Specifications of *constrained devices* [10], [11]

Device	CPU	Clock frequency	GPU	Memory	Price
Raspberry Pi 3B	quad-core ARM Cortex A53	1.2GHz	None	1GB LPDDR2	40 EUR
Raspberry Pi 4	quad-core ARM Cortex A72	1.5GHz	None	4GB LPDDR4	40 EUR
Odriod XU4	Cortex A15	2GHz	None	2GB LPDDR3	70 EUR
NVidia Jetson Nano	quad-core ARM A57	1.43GHz	128-core Maxwell	4GB LPDDR4	99 USD
Dell XPS 13 9360	Intel(R) Core(TM) i7-7560U	2.4GHz	Intel eGPU	16GB LPDDR4	1000 EUR

In this paper the Dell XPS 13 9360, Raspberry Pi 3B and Raspberry Pi 4 will be used. These because of the availability and specifications. All these platforms are supported by Tensorflow/Keras, which makes them ideal for our use case. Each of these devices come from a different price class, which offers us insight into how the price would impact the performance.

C. Convolutional Neural Network

Since 2 types on input are measured, IQ and RSSI, multiple networks will be trained to see how the performance will be on the selected platforms. For each input type, 2 models will be trained with different levels of trainable parameters. Trainable parameters are a large factor in how much computational power is needed to run the network.

- simple network: a simple network means that it will have a low level of trainable parameters.
- complex network: a complex network is a network with a "high" level of trainable parameters.

Each of these networks is trained on the same datasets and on the same hardware. The training hardware is a server node with 8 CPUs, 51GB RAM and 2 NVidia 1080 GPUs. Before training the raw IQ-samples, are combined to compute the RSSI value: $RSSI = \sqrt{I^2 + Q^2}$

Since we need to keep in mind that these computations also need to be executed on a constrained device, we can approach the RSSI value by using the following formula: $RSSI^2 = I^2 + Q^2$.

The RSSI models reach high accuracies, but the difference between the simple RSSI model (Figure 1) and the complex model (Figure 2) is very noticeable (10%).

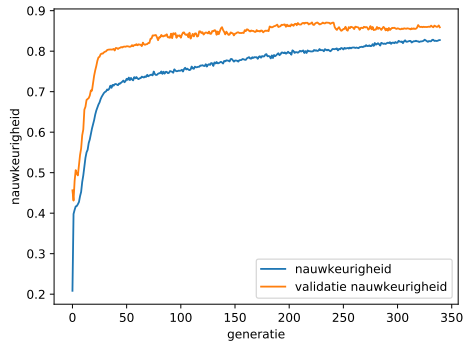


Fig. 1: Accuracy over generation - simple RSSI model

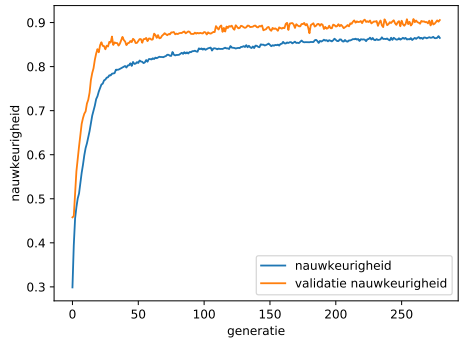


Fig. 2: Accuracy over generation - complex RSSI model

In contradiction to the RSSI model, the difference between the complex and simple IQ model is negligible (Figure 3 & 4). The IQ model has more data available to it and offers a more robust network.

It should be noted that the simple IQ model almost trained twice as long as the complex IQ model (figure 3 & 4) before early-stopping method stopped the training.

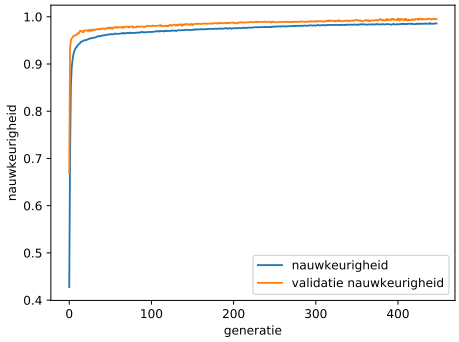


Fig. 3: Accuracy over generation - simple IQ model

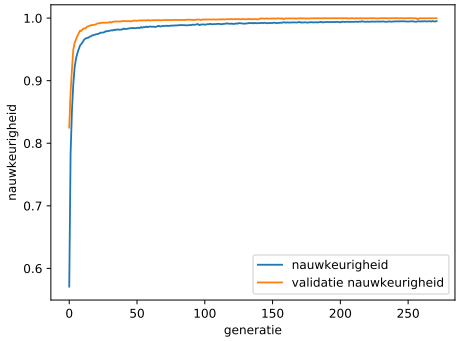


Fig. 4: Accuracy over generation - complex IQ model

III. Performance

The performance of these models on the chosen platforms are evaluated by iterating 100 times for each sample size. One sample is a measurement of 512 IQ-samples which results in a 2x512 input vector for the IQ network and a 1x512 input vector for the RSSI network.

A. Processing speed - samples amount

This section covers the impact of the amount of samples on the processing speed. The sample amount vary from 2 samples up to 15000 samples. The upper limit is the constrained of the amount of samples we can read from the RTL-SDR dongle in 1 iteration. The processing speed is defined as the amount of samples per second that can be processed.

The processing speed can be calculated from the timing data through the following formula:

$$average\ processing\ time = \frac{AVG(100\ measurements)}{samples\ per\ measurement}$$

$$average\ processing\ speed = (avg\ processing\ time)^{-1}$$

In Figure 5 and Figure 6 we can see that the RSSI models have a higher processing speed then the IQ models. But all

models have the same characteristic, the performance boost decreases exponentially with the amount of samples taken. When 1024 or more samples are taken, the processing speed does not increase drastically anymore.

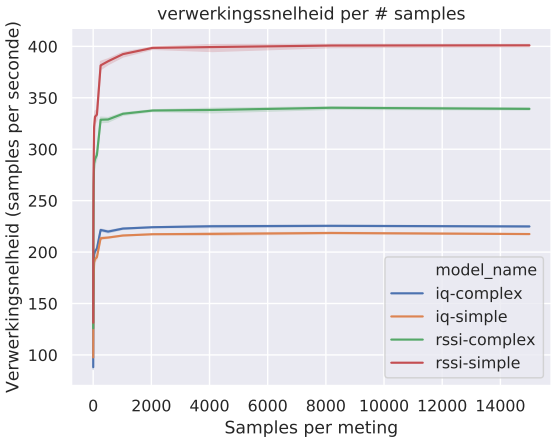


Fig. 5: Processing speed Raspberry Pi 3

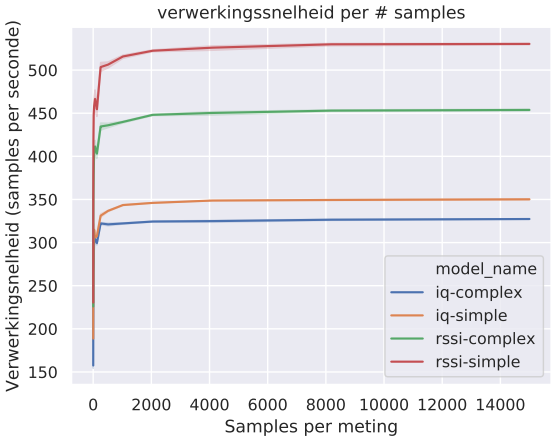


Fig. 6: Processing speed Raspberry Pi 4

B. *Processing speed - complexity*

The complexity (= the amount of trainable parameters) of the different models also have an impact on the computing power that is needed, and thus will also influence the performance of the models on the constrained devices.

The complexity of the model is not only dependant on the number of layers, but also on the layers used in the model. Max pooling layers in general lower the complexity and are used after the convolutional layer to decrease the filter output complexity from the convolutional layers.

In figure 7 and 8 it is noticeable that in general the RSSI models have a higher throughput then the IQ models. This means that the pre-processing that needs to happen for the RSSI models does not have a larger impact then the lower complexity inherent for the RSSI models (because of the smaller input size).

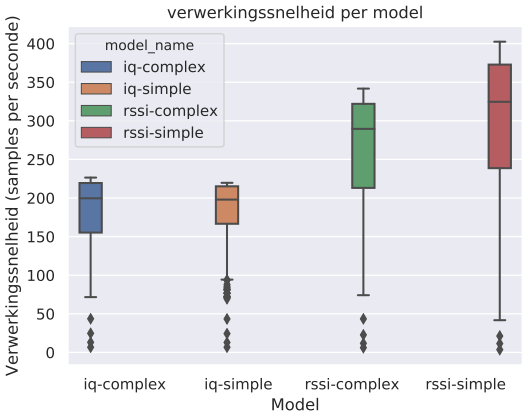


Fig. 7: Processing speed Raspberry Pi 3

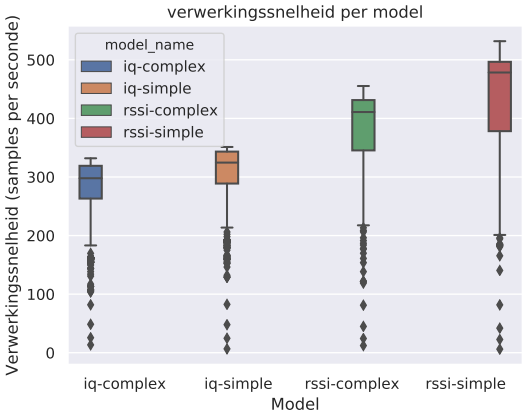


Fig. 8: Processing speed Raspberry Pi 4

C. *Resolution of the measurements*

From the previous section we can conclude that 1024 or more samples offer the best throughput. However one has to observe that with higher sample amounts (figure 9), longer processing times can be expected. Not only throughput has to be considered, the resolution of the measurements also matter.

The measurements should happen often enough, so a sense of the ”current” situation can be done, but at the same time the measurement should be long enough to capture useful data and transmissions.

A compromise between the processing time and the sampled time needs to be found. A sample amount of 1024 has a large throughput (325-475 samples per second), while still being able to have an acceptable resolution (500ms measurement every 3.1 s).

IV. Conclusion

Constrained devices are catching up with consumer level HW, while the performance difference is still noticeable, the performance of the constrained devices is at an acceptable level for some practical applications. It is however required for each application, the performance is tested against the constrains of the application, since the performance can vary greatly depending on the application.

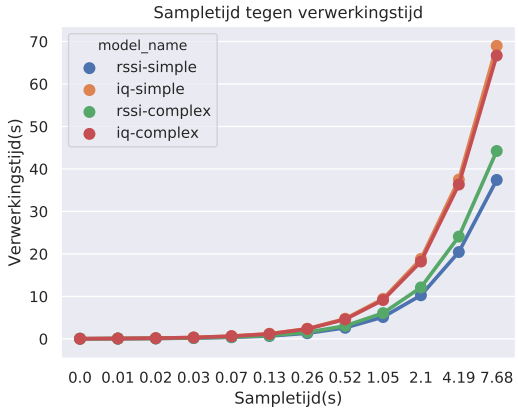


Fig. 9: Sampled time vs processing time

The selected devices have shown to be capable of running the different CNN networks at acceptable speeds. It can be noted that while there is a small difference between the IQ and RSSI models, the accuracy difference is more noticeable. Depending on the requirements of the application: speed or accuracy, it is possible to fine-tune the CNN models for it and select off-the-shelf components for it.

V. References

[1] J. Tan and S. G. Koo, "A survey of technologies in internet of things," *Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2014*, pp. 269–274, 2014. DOI: 10.1109/DCOSS.2014.45.

[2] C. Goursaud and J. M. Gorce, "Dedicated networks for IoT: PHY / MAC state of the art and challenges," *EAI Endorsed Transactions on Internet of Things*, vol. 1, no. 1, p. 150 597, 2015. DOI: 10.4108/eai.26-10-2015.150597.

[3] R. Miller, W. Xu, P. Kamat, and W. Trappe, "Service discovery and device identification in cognitive radio networks," *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, pp. 670–677, 2007. DOI: 10.1109/SAHCN.2007.4292880.

[4] N. Bitar, S. Muhammad, and H. H. Refai, "Wireless technology identification using deep convolutional neural networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2017-Octob, 2018, pp. 1–6, ISBN: 9781538635315. DOI: 10.1109/PIMRC.2017.8292183.

[5] H. Li, "Multi-agent Q-learning of channel selection in multi-user cognitive radio systems: A two by two case," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, no. October, pp. 1893–1898, 2009, ISSN: 1062922X. DOI: 10.1109/ICSMC.2009.5346172.

[6] S. Zacharias, T. Newe, S. O'Keeffe, and E. Lewis, "2.4 GHz IEEE 802.15.4 channel interference classification algorithm running live on a sensor node," *Proceedings of IEEE Sensors*, pp. 1–4, 2012. DOI: 10.1109/ICSENS.2012.6411279.

[7] M. Schmidt, D. Block, and U. Meier, "Wireless Interference Identification with Convolutional Neural Networks," 2017.

[8] S. Grimaldi, A. Mahmood, M. Gidlund, and S. Member, "Real-time Interference Identification via Supervised Learning : A Coexistence Framework for Massive IoT Networks," pp. 1–14,

[9] S. Oh, M. Kim, and M. Lee, "Investigation on Performance and Energy Efficiency of CNN-based Object Detection on Embedded Device," p. 4, 2017. DOI: 10.1109/CAIPT.2017.8320657.

[10] *NVidia Jetson Nano product page, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/*, Accessed: 2019-08-15.

[11] *Dell XPS 13 product page, https://www.dell.com/nl-be/shop/dell-laptops/xps-13/spd/xps-13-9380-laptop/cnx38004*, Accessed: 2019-08-15.

IV Samenvatting

De opkomende technologie die bekend staat als Internet of Things (IoT) zal veel druk uitoefenen op de Low-Power Wide Area Network technologieën. Naarmate meer en meer apparaten verbinding maken met de cloud, is daar behoefte aan beter beheer en een manier om goede technologieën te selecteren, afhankelijk van de omgeving. Huidig onderzoek in "slimme netwerken" maakt gebruik van gespecialiseerde en dure hardware. Dit maakt de selectie en het beheer van deze LPWANs niet beschikbaar voor alledaags gebruik.

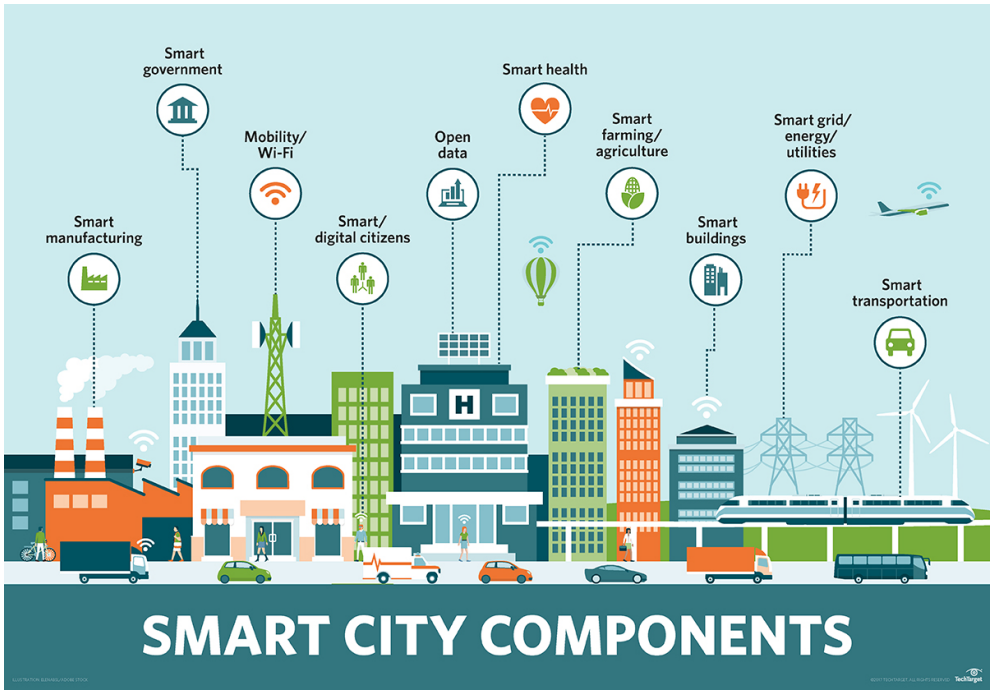
Er wordt voorgesteld om low-end embedded apparaten en standaard Software Defined Radio (SDR) USB-dongles te gebruiken. Om dit aan te pakken worden twee vragen gesteld: 1.) is het mogelijk om low-end hardware te gebruiken voor interferentiedetectie 2.) hoe goed werkt de low-end hardware hiervoor?

Drie verschillende complexiteitsniveaus van Convolutional Neural Networks (CNN) worden getest op drie verschillende platforms: Dell XPS 13 9360, Raspberry Pi 4 en Raspberry Pi 3B. De voorgestelde CNNs hebben zeer acceptabele prestaties op de Dell XPS 13 9360. Op de Raspberry Pi platformen is de performantie goed genoeg voor bepaalde praktische toepassingen, echter zijn er nog enkele limitaties.

Er kan dus besloten worden dat voor bepaalde praktische toepassingen de beschikbare hardware (zoals bv. de Raspberry Pi 3B en 4) geschikt zijn voor gebruik. Hierbij moet men echter rekening houden met de mogelijke beperkingen van deze hardware. Opkomende apparaten met toegepaste hardware voor Machine Learning (zoals bv. Jetson Nano) kunnen de performantie sterk bevorderen.

1 Inleiding

Door de alom groeiende vraag naar data van de hedendaagse maatschappij is het nodig om het aantal datapunten te vergroten. Om dit te bekomen kan gebruik gemaakt worden van bestaande technologieën, maar ook van de opkomende technologieën. Om aan deze vraag tegemoet te komen zullen het Internet of Things (IoT), het interconnecteren van objecten op grote schaal, en smart cities, homes en things (het gebruik van IoT technologieën om objecten met elkaar te communiceren) een belangrijke rol spelen in de toekomst.



Figuur 1: Smart city componenten: Het "slim" maken van objecten en open data creëren de vraag naar meer sensoren en apparaten [1]

Veel van deze technologieën zijn draadloos en maken dus gebruik van radiogolven voor hun communicatie. Deze technologieën maken gebruik van verschillende frequenties en worden onderverdeeld in verschillende radiospectra.

Naarmate dat deze technologieën groeien en vaker gebruikt worden om de IoT filosofie toe te passen, zullen deze radiospectra steeds meer gebruikt worden. Dit zal echter als gevolg hebben dat de performantie van de communicatie hieronder zal lijden. Overvolle spectra leiden namelijk tot interferentieverschijnselen zoals verstoorde communicatie, verzwakte signalen en miscommunicatie [2]–[4].

Het beheer van deze netwerken speelt dus een belangrijke rol om er voor te zorgen dat de juiste technologie gebruikt wordt en deze correct gebruikt wordt. Voor dit beheer zijn monitoringmiddelen uiterst belangrijk en niet weg te denken.

Het monitoren is belangrijk voor de handhaving van de regelgeving en de performantie van de technologieën. De huidige trend van IoT ontwikkeling zorgt ervoor dat het radiospectrum steeds drukker bezet wordt. Het wordt alsmaar belangrijker om het radiospectrum op grotere schaal te monitoren en dit makkelijker te maken.

Het spectrum monitoren op zo een grote schaal brengt een aantal uitdagingen met zich mee. Ten eerste is de kost van de apparatuur en de sensors die hiervoor gebruikt worden momenteel heel hoog. Daarnaast is het complex om met deze apparatuur te werken. Op grote schaal werken brengt ook privacy bezorgdheden met zich mee. Dit allemaal maakt het beleid rond deze LPWAN technologieën moeilijker. Daarnaast moet de technologie zich ook kunnen aanpassen aan de snel evoluerende innovaties in dit veld.

Deze thesis heeft als doel het ontwikkelen van een eenvoudig te gebruiken apparaat, dat in staat is om lokaal interferentie te detecteren en technologieën te classificeren. Dit apparaat is een *constrained device* (een apparaat met beperkte rekenkracht, opslag of communicatiemogelijkheden). Op deze manier wordt het monitoren toegankelijker gemaakt.

De doelstellingen zijn:

- Het implementeren van Automatische Technologie Classificatie (ATC), het automatisch classificeren van technologieën, voor IoT netwerken op een *constrained device*. Dit door het opbouwen van een gevalideerd

CNN model dat classificatie van IoT netwerken doet en kan draaien op een *constrained device*.

- Het verlagen van de kostprijs van monitoringmiddelen in vergelijking met huidige oplossingen, dit door het gebruik van *constrained devices* en Software Defined Radios (SDR) adapters.

Hoofdstuk 2 behandelt de bestaande literatuur die aanwezig is over het herkennen van draadloze radiotechnologieën en het gebruik van CNN op embedded platformen.

In hoofdstuk 3 wordt behandeld welke componenten een apparaat nodig heeft om de herkenning van het sub-GHz spectrum uit te voeren.

In hoofdstuk 4 wordt besproken welke implementatie mogelijk is, waarom voor deze implementatie gekozen wordt en hoe dit in realiteit omgezet wordt.

In hoofdstuk 5 wordt de performantie van de implementatie besproken op de verschillende platformen die beschikbaar zijn.

Hoofdstuk 6 behandelt verder onderzoek dat mogelijk is ter verbetering van de implementatie die in de thesis voorgesteld wordt.

2 Literatuurstudie

In dit deel van de thesis wordt onderzoek gedaan naar bestaande technologieën en oplossingen die er zijn. Hiervoor wordt eerst gekeken naar de beschikbare en opkomende IoT netwerken. Daarna kijken we naar de bestaande methoden voor de classificatie en technologieherkenning. Uiteindelijk worden de mogelijkheden bekeken om Machine Learning op *constrained devices* te doen.

2.1 Internet of Things netwerken en Low-Power Wide Area Networks

J. Tan *et al.* geven een overzicht van bestaande technologieën en hun eigenschappen voor het moderne IoT. Zo worden ook de mogelijkheden van QR codes, Radio Frequency Identification (RFID) en sensoren bekeken en uitgelicht. Daarna werpen ze een blik op de technologieën gebruikt in netwerken. Hierbij wordt aangehaald dat kennis van het 802.15.4 protocol nuttig kan zijn, gezien veel componenten van dit protocol terugkeren in andere technologieën. Hierna worden Zigbee, Z-Wave en 6LoWPAN besproken[5].

Door de opkomende noden, die zowel IoT en Machine-to-Machine (M2M) hebben, is het nodig om te kijken welke technologieën gebruikt kunnen worden om aan deze noden te voldoen. Hierbij wordt vooral gekeken naar de noden van IoT gezien dat M2M een uitbreiding hiervan is. Voor IoT is het nodig dat er een groot aantal apparaten mee kunnen verbonden worden, dat er een beperkte infrastructuur nodig is en dat een lange batterijduur mogelijk is. Een continue verbinding heeft dus niet de grootste prioriteit. De communicatie voor IoT toepassingen is namelijk heel sporadisch (1 keer per week tot 1 keer per jaar). Deze berichten moeten wel betrouwbaar afgeleverd worden.

Om deze netwerken te bouwen is er de keuze: (i) de bestaande telefoon infrastructuur uitbreiden en uitbouwen, zodat de bestaande 4G (en toekom-

stige 5G) netwerken gebruikt kunnen worden en (ii) netwerken opbouwen met nieuwe technologieën. Het gebruik van de bestaande infrastructuur lijkt een oplossing, echter heeft deze infrastructuur wel een aantal beperkingen. Deze kan namelijk niet omgaan met het groot aantal apparaten dat IoT met zich mee zou brengen. Daarnaast is deze technologie veel te statisch, omdat deze zich te veel focust op datadoorvoer in plaats van betrouwbaarheid. Het uitbouwen van een nieuw mobiel netwerk met oog op de noden van IoT is dus een goede optie [6].

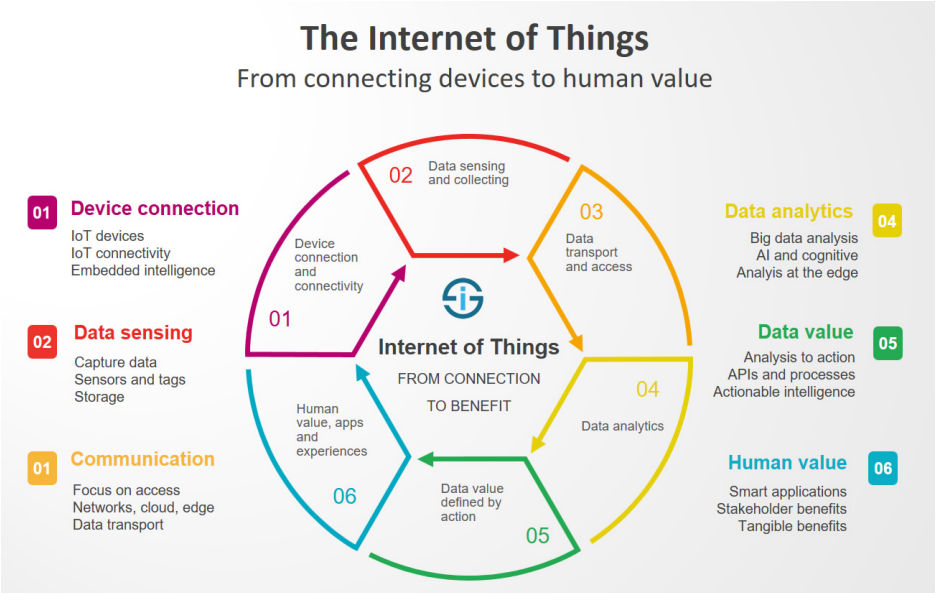
In figuur 2 zijn de stappen te zien die nodig zijn om van data over te gaan naar bruikbare informatie. Deze cyclus wordt opgesplitst in zes stappen. De eerste drie stappen zijn:

- Connectiviteit van apparaten: men moet evalueren welke apparaten en technologie men gebruikt voor het connecteren van de data.
- Data verzameling: in deze stap moet men de vraag stellen welke informatie men wil verzamelen en op welke manier men dit zal doen.
- Communicatie van het apparaat: hier evalueert men hoe vaak men de data van het apparaat zal ophalen en hoe men deze data opslaat.

In elk van deze drie stappen kan men gebruik maken van IoT technologieën. De volgende 3 stappen gaan voornamelijk over de verwerking van de verzamelde data en hoe men nuttig gebruik kan maken van de uiteindelijk verkregen informatie [7]. In deze stappen kan men bv. ook gebruik maken van ML technieken.

- Data analyse: de verzamelde data wordt geanalyseerd door middel van bv. statistiek of ML. Door deze stap wordt data in informatie veranderd.
- Waarde door actie: op basis van de verkregen informatie neemt men actie.

- Menselijke waarde: de acties leiden tot mobile apps en ervaring. Die ervaring zal op zich leiden tot meer vraag voor connectiviteit van apparaten.



Figuur 2: IoT van technologie naar menselijke waarde [7]

Door de vraag naar meer automatisatie in de woningen van particulieren, komen er ook meer sub-GHz netwerken in het straatbeeld. Sub-GHz netwerken worden voornamelijk gebruikt in particuliere autmatisatie vanwege de voglende voordelen: (i) energiezuinig, (ii) lage kosten en (iii) minder gevoelig aan interferentie. Doordat deze energiezuinig werken kunnen veel van deze netwerken gemakkelijk op batterijspanning langdurig werken. Dit biedt het voordeel dat men geen grote verbouwingswerken moet uitvoeren om een energiebron dichtbij te voorzien. De lage kosten, in vergelijking met traditionele automatisatie, maken de instapdrempel aanzienelijk lager. Door gebruik te maken van een minder bezet spectrum is de kans op interferentie veel kleiner [8].

Een volledig geautomatiseerd huis zal deze technologieën gebruiken voor het aansturen van de lichten, rolluiken, poorten, stopcontacten en gordijnen, Maar bv. ook voor het regelen van de verwarming en luchtcirculatiesystemen.

Daarnaast zullen ook veiligheidsapparaten (zoals rookmelders, brandmelders, diefstalpreventie, ...) gebruik maken van deze technologieën.

Mogelijke kandidaten voor deze technologieën zijn:

- Zigbee
- 6LoWPAN
- Z-Wave
- INSTEON
- Wavenis
- LoRaWAN
- SigFox

Van deze kandidaten zijn Zigbee en 6LoWPAN ontwikkeld om in een algemene context te werken, terwijl Z-Wave, INSTEON en Wavenis ontwikkeld zijn met specifiek oog voor automatisatie in huizen.

2.1.1 Zigbee

Zigbee is opgebouwd uit 4 lagen: PHY, MX, Network en APP waarboven Zigbee ook security maatregels ondersteunt en de mogelijkheid voor tree of mesh routing. In de sub GHz-band wordt BPSK gebruikt. Het onderliggend PHY/MAC protocol IEEE802.15.4 ondersteunt 2 manieren voor MAC [9]:

- Bacon-enabled : centrale node die coordineert door middel van 3 tijdverdelingen
- Beaconless: plain CSMA/CA

2.1.2 Z-wave

Z-wave is specifiek ontwikkeld om betrouwbaar een bericht van zender naar receiver te krijgen. Het is opgebouwd uit 5 lagen: PHY, MAC, transfer, routing en applicatie. De MAC laag biedt een CA mechanisme aan. Er is ook de mogelijkheid tot hertransmissies op basis van ACK pakketten [9].

2.1.3 INSTEON

INSTEON is een mesh netwerk dat bestaat uit RF apparaten en "power line" adapters. Elk apparaat in INSTEON is een peer, wat betekent dat ze de rol kunnen opnemen als zender, ontvanger of doorgever. Communicatie met apparaten niet in het bereik van de ontvanger liggen, worden als multihop verzonden [9].

2.1.4 Waveniss

Waveniss legt de PHY en netwerk laag van apparaten vast en biedt de applicatie laag toegang via APIs. De MAC sublaag biedt zowel de mogelijkheid voor synchone als niet-synchrone apparaten. Voor synchrone netwerken geldt dan dat de apparaten werken op basis van CSMA/TD. In niet-synchrone netwerken, wordt CSMA/CA gebruikt. Elk apparaat heeft ook de mogelijkheid om gebruik te maken van QoS [9].

2.1.5 6LoWPAN

6LoWPAN is de implementatie van IPv6 protocol stacks voor het gebruik in LPWAN netwerken. Hoewel het netwerk een mesh topologie heeft, zijn er 2 opties voor de routing: (i) mesh under en (ii) route over. Mesh under maakt gebruik van IEEE802.15.4 adressen (en werkt dus onder de IP laag). In route over, is elke radio hop het equivalent van een IP hop [9].

2.2 Groei van het IoT

Naarmate dat de technologieën groeien wordt duidelijk dat de manuele methodes voor technologieherkenning niet schaalbaar zijn. De netwerken en het aantal apparaten worden zodanig groot dat een manuele blackbox analyse van het netwerk uitvoeren onhoudbaar is wanneer er zich een probleem voor doet.

Daarbij wordt ook het combineren van verschillende databronnen belangrijk. In deze stap is het ook van belang om ervoor te zorgen dat de data zuiver en accuraat is. Deze combinatie is belangrijk omdat de locatie van observatie alsook andere parameters (naast de link specifieke parameters) een invloed hebben op de dataverzameling.

Daarnaast mag men de impact van een dynamische omgeving niet onderschatten. Zo hebben omgevingen met bewegende objecten een impact op de signaal propegatie, maar ook de omgevingseigenschappen (vochtigheid en temperatuur) kunnen een impact hebben op de performantie van een computersysteem.

Het is aangetoond dat het gebruik van off the shelf componenten haalbaar is [10].

2.3 Automatische Technologie Classificatie en bestaande methoden

Door de groei van draadloze technologieën en het toenemend aantal IoT apparaten die in de Industrial, Scientific and Medical (ISM) bandbreedte functioneren, zal de vraag naar intelligente, flexibele systemen toenemen. Dit voornamelijk door de beperkte bandbreedte van de ISM band, die voor een drukke radio omgeving zorgt. Deze systemen zullen ook een cruciale rol spelen. Deze flexibele systemen zijn in staat om hun configuraties aan te passen aan de omgeving, doordat deze de omgeving waarnemen door het gebruik van SDR. De interferentiedetectie is noodzakelijk voor de volgende doeleinden:

- Ter verbetering van de performantie van IoT netwerken door het slimmer maken van transmittors. Slimme transmittors zouden namelijk hun parameters kunnen aanpassen op basis van de omgeving.
- Ter verbetering van de veiligheid van IoT netwerken: aanvallen moeten gedetecteerd worden om deze te kunnen weren [11]–[13].

2.4 ML op *constrained devices*

Vele onderzoeken naar coëxistentie van sub GHz-netwerken, vereisen geavanceerde signaalverwerkingsapparatuur en krachtige radio's voor het verzamelen van de gegevens. Deze thesis behandelt het idee om op basis van *constrained devices*, goedkope SDR dongles en een getraind model de classificatie te doen. Het uiteindelijk doel is om een goedkoop apparaat te maken, dat in staat is om zijn radio omgeving te begrijpen en te identificeren wanneer deze omgeving onbekend is.

Het idee om met *constrained devices* te werken heeft al eerder exploratie gehad. Dit dan voornamelijk onder de vorm dat er *constrained devices* gebruikt worden als sensoren, waarna de opgemeten data doorgestuurd wordt naar een centraal punt. Hierin wordt gekeken om de data te beperken die doorgestuurd wordt vanaf het *constrained device* [14].

R. Pehgghg *et al.* hebben onderzoek gedaan naar het draaien van objectherkenningssoftware op basis van CNN. Hiervoor wordt een aangepaste versie van Single Shot Detector (SSD) gebruikt, waar de backbone van elke laag MobilenetV1 is, in plaats van het VGG framework. De implementatie op een NanoPi2 levert een performantie van 1.13 Frames Per Second (FPS) op. Dit is niet heel snel, omdat de gebruikte hardware enkel Cortex-A9 s heeft, die enkel gebruikt worden voor general computing [15].

ML kan een methode zijn voor het classificeren en identificeren van draadloze technologieën in de sub GHz-bandbreedte. ML methoden hebben al succes geboekt in de 2.4GHz bandbreedte op vlak van identificatie en classificatie. Vooral het gebruik van Deep Neural Network (DNN) en Convo-

lutional Neural Network (CNN) lijkt succesvol. De ML methodes baseren zich op bepaalde eigenschappen van de draadloze technologieën. Er zijn implementaties die gebruik maken van eigenschappen van de Medium Access Control (MAC) laag, de I/Q-waarden of de Recieved Signal Strenght Indication. Hoewel de MAC laag interessante eigenschappen heeft voor classificatie, is aangetoond dat men meer voordeel haalt door de PHY laag te gebruiken. Dit vooral doordat veel MAC implementaties beveiligd en hard coded in de controllers op firmware niveau zijn. Voordat men dus op MAC laag zou kunnen werken moet men de firmware reverse engineeren. De PHY laag daartegen, is vrij toegankelijk en makkelijk te meten met spectrumanalyzers of SDR [11], [12], [16]–[18].

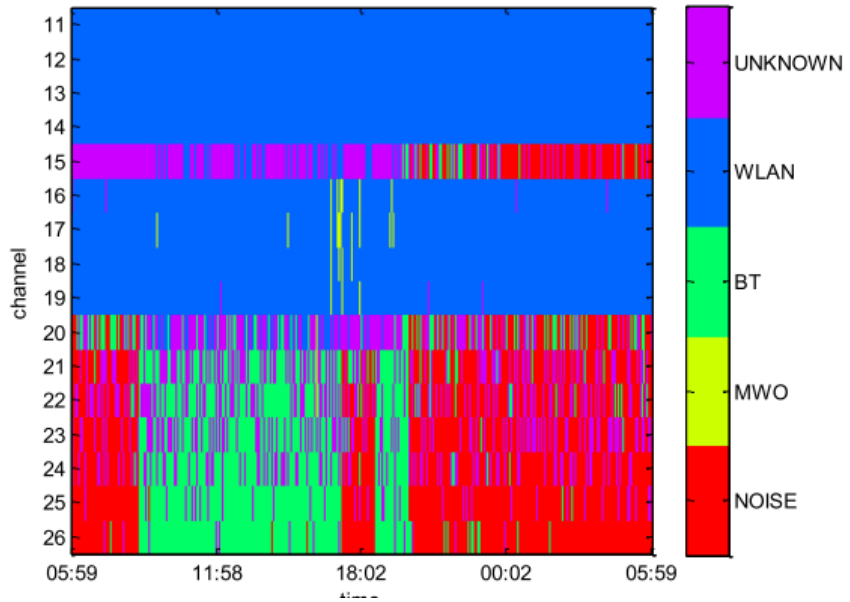
Youssef et al. hebben vier methodes voor RF classificatie bestudeert: (i) Support Vector Machine (SVM), (ii) DNN, (iii) CNN en (iv) Multi Stage Training (MST) A-LM. Van deze methodes lijkt MST A-LM de performantste methode. Kort gevolgd door CNN. MST A-LM en CNN verslaan de andere methoden, maar wanneer men kijkt naar een groot aantal onbekende zenders dan komt MST A-LM voorop [13].

S. Hu *et al.* tonen aan dat het mogelijk is om op basis van SVM een onderscheid te maken tussen verschillende MAC klassen. Hiervoor heeft men gebruik gemaakt van een experiment waarbij men onderscheid moet maken tussen 4 verschillende MAC technologieën (TDMA, CSMA/CA, pure ALOHA, slotted ALOHA). Het experiment meet vermogen- en tijdeigenschappen en gebruikt deze als input voor de SVM. Verschillende MAC protocollen hebben verschillende performantie, voornamelijk door de eigenschappen ervan. Slotted ALOHA is makkelijker te herkennen dan de andere 3, doordat deze technologie veel sterkere en nauwkeurigere vermogenseigenschappen aantonen. Daarnaast zorgt hoge netwerkbelasting voor een betere herkenning, doordat er meer pakketbotsingen zijn, die op hun beurt meer informatie (zowel vermogen- als tijdeigenschappen) bezorgen [19].

In een onderzoek wordt voorgesteld om in plaats van een RSSI sweep van de volledige bandbreedte te doen, en bijgevolg de node offline te halen, kanaal

per kanaal te sweepen. Hierdoor blijft de node verbonden met het netwerk en blijven zend/ontvangstcapaciteiten intact. De classificatie gebeurt in drie stappen: sampling, feature extractie en classificatie. De sampling fase duurt een seconde. Op basis van testen hebben de auteurs vastgesteld dat -85dBm een goede threshold is. De samplerate is 8192Hz. Tijdens de feature selectie bepaalt men de frequentie component voor de verschillende technologieën. Hiervoor implementeert men een algoritme dat minder complex is, Fast Fourier Transform (FFT), en geen floating point vereist zoals Goertzel. De classificatie wordt bepaald door specifieke kenmerken van technologieën die af te leiden zijn uit de RSSI metingen. Ruis wordt herkend aan het feit dat als een kanaal niet gebruikt wordt (RSSI <-85dBm), er geen metingen zijn. De microgolfoven wordt herkend aan de periodiciteit van het signaal in vergelijking met de andere geteste technologieën. Voor andere technologieën is periodiciteit heel overeenkomstig. Om deze te onderscheiden wordt er dus gekeken naar de lengte van de signalen en de "rust" tussen transmissies. Alles dat buiten de herkende technologieën valt wordt als "onbekend" geclassificeerd. Het algoritme is getest in zowel een anechoische kamer als in een realistische testomgeving (kantoorruimte). In de anechoische kamer is het algoritme heel dicht bij ideaal, en zelfs in de kantooromgeving is het algoritme performant. Zoals in figuur 3 te zien is, kan het algoritme de grootste bron van interferentie weergeven [17].

S. Rajab *et al.* hebben verschillende soorten 802.11 netwerken proberen onderscheiden op basis van naïeve Bayes en k-nearest neighbour classificatie. De gebruikte data werd verzameld uit zowel homogene als heterogene 800.11b/g/n netwerken. De homogene netwerken zijn opgebouwd uit een type van 802.11, heterogene netwerken zijn opgebouwd uit twee of drie 802.11 types. De verzamelde data bestond uit de ontvangen signaalsterkte (Relative Signal Strength (RSS)). Deze dataset werd dan op basis van een op voorhand bepaalde threshold opgesplitst in twee subsets: actief kanaal en inactief kanaal. Voor zowel homogene als heterogene netwerken, is bevestigd dat beide technieken werken en identificatienauwkeurigheden van 85.9% tot



Figuur 3: 24u opname van het algoritme [17] in een kantoor omgeving

96.83% mogelijk zijn [20].

2.4.0.1 Convolutional Neural Network

Schmidt *et al.* doen een voorstel voor het gebruik van een CNN voor Wireless Interference Identification (WII). Er wordt een neuraal netwerk gebruikt op basis van het model van O. Shea *"Convolutional radio modulation recognition networks,"*. Hiervoor wordt een $128x2$ matrix gebruikt die 128 IQ-samples bezit, gemeten aan een samplerate van 1Msps. Het model toont voor lage Signal-to-Noise Ratio (SNR) waarden (<-5dB) een hoge nauwkeurigheid (95%). Er zijn 2 problemen waargenomen:

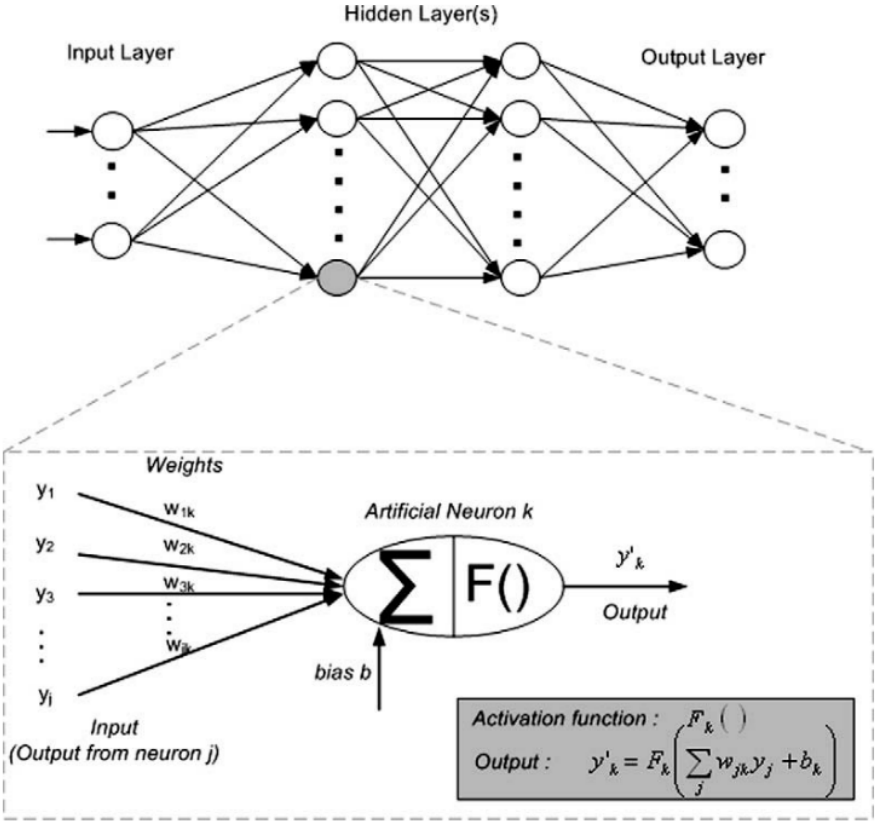
- Samples, die geclipt zijn omdat de bandbreedte tegen de rand van de waargenomen bandbreedte zit, hebben een lagere nauwkeurigheid dan de normale samples.
- Samples, waarvan de centerfrequentie overkomt met het eerste IEEE802.15.1 en vierde IEEE802.15.4 kanaal, hebben minimale nauwkeurigheid.

Ondanks deze 2 problemen, is het model performanter dan de state-of-the-art [18].

Artificiële neurale netwerken worden, net zoals biologische neurale netwerken, opgebouwd uit neuronen. In deze neurale netwerken maakt men een onderscheid tussen 3 types neuronen:

- input neuronen: ontvangen de data van buiten het netwerk
- hidden neuronen: de input/output van dit netwerk blijft binnen het netwerk (en dus verborgen/hidden)
- output neuronen: verzenden data naar buiten het netwerk.

De neuronen worden onderling verbonden door connecties en de structuur van deze connecties is afhankelijk van de soort laag. Elk van deze connecties krijgt een gewicht W_{nk}, die de invloed van de output van het neuron op het volgende neuron bepaalt. Elke neuron heeft ook een activatiefunctie die bepaalt wanneer en hoe de output van het neuron "actief" wordt.



Figuur 4: Overzicht van een artificieel neuraal netwerk [21]

De focus van dit onderzoek ligt op herkenning en voorspelling van signalen. Hiervoor beschrijft men twee scenario's waarvoor men neurale netwerken traint. Tijdens de bespreking van de performantie van deze netwerken wordt de vaststelling gedaan dat beide een slechtere performantie hebben op de validatie datasets dan op de trainingset. Dit komt voornamelijk omdat opnamen van radiosignalen afhankelijk zijn van veel parameters [21].

Embedded platformen zijn tot op de dag van vandaag nooit beschouwd als geschikte platformen om CNNs op te doen. Dit komt voornamelijk door de hoge computationele eisen die CNN opleggen. Echter met de opkomst van General Purpose Graphical Processing Unit (GP-GPU) in embedded platforms wordt het aantrekkelijk om embedded platforms te gebruiken om CNN te versnellen of breder in te zetten. Hoewel de embedded platforms met een GP-GPU nog niet volledig de dedicated platforms benaderen (behalen

65% van de performantie), zijn de embedded platforms wel aantrekkelijk. Dit komt vooral omdat deze de performantie bekomen met ongeveer 1,5% van het energieverbruik van het dedicated platform. Dus bij toepassingen waar de performantie daling niet cruciaal is, zijn embedded platforms wel degelijk een goede optie [22].

Een CNN kan ook getraind worden op basis van IQ data. Hiervoor wordt de 80MHz ISM band gesweeped met een resolutie van 285kHz. Er is ook gekozen om low-end spectrumanalyzers te simuleren, dit zodat de implementatie door de meeste scanners mogelijk is. Er werd ook geclassificeerd op basis van SNR. Het netwerk dat bestudeerd werd is opgebouwd uit WiFi 802.11n, ZigBee en Bluetooth. Verschillende cases werden bestudeerd. Homogene netwerken van een van de drie technieken of verschillende permutaties van heterogene netwerken. De input voor de CNN is een 2D matrix met grootte 280x280 bestaande uit I/Q samples van de netwerken. De input data is genormaliseerd en geschaald voordat deze in het CNN werd gestoken. Dit is belangrijk aangezien er geen onderscheid werd gemaakt op basis voor SNR, maar dat training en valideren allebei worden gedaan op een mengeling van SNR om de fluctuaties die in realiteit kunnen voorkomen te simuleren. Dit zorgt ervoor dat het model getraind werd om elke random input dataset aan te kunnen. De CNN die in de paper voorgesteld wordt, is opgebouwd uit vijf lagen: eerst vier lagen opgebouwd uit opeenvolgende convolutionele en pooling lagen, opgevolgd door een fully connected layer. Uit hun testen blijkt dat hun voorgesteld model 100% nauwkeurigheid haalt voor training en 96.2% voor validatie. In vergelijking met andere classificatiemethoden, scoort het model consistent hoger dan de andere methoden. Ook bleek dat een groter CNN niet tot hogere nauwkeurigheid leidde [12], [20].

Grimaldi *et al.* tonen aan dat op basis van Supervised Learning (SL) het mogelijk is om 4 verschillende technologieën te classificeren in werkelijke heterogene omgevingen met een nauwkeurigheid van 90% tot 97%. Onderzoek toont aan dat SNR en de keuze van datatype een grote rol speelt bij de performantie van het CNN. Zo worden performantie verbeteringen van rond

20% aangetoond in [23], [24].

O' Shea *et al.* erkennen dat zowel FCC als Europese equivalente organisaties strenger zullen toezien op radio transmissies. Dit komt vooral door de groeiende vraag naar draadloze technologieën [25]. De auteurs passen de successen van CNN en DNN in zowel audio- als videotoepassingen toe en willen dit gebruiken om radiomodulaties te herkennen. Ze vergelijken twee methoden voor modulatieherkenning: herkenning gebaseerd op integrated cyclic moment en op basis van convolutionele feature herkenning. De gebruikte dataset is gegenereerd door het gebruik van de Radio Channel blocks in GNURadio en is opgebouwd uit elf modulatietechnieken. In veel gevallen wordt het gebruik van synthetische data in ML veroordeeld, maar radio transmissies geven een unieke kans hiervoor, aangezien deze signalen toch altijd gesynthetiseerd worden. De gebruikte dataset is beschikbaar op https://www.deepsig.io/datasets. Hun onderzoek toont aan dat het gebruik van CNNs voor lage SNR metingen performanter is dan bestaande technologieën. Hierdoor zou men het effectief meetbereik van bestaande netwerken kunnen vergroten. Ook de trainingstijd en classificatietijd van de CNN worden geëvalueerd. En hoewel de trainingstijd iets langer is dan veel technieken, blijkt dat de classificatietijd vrij laag ligt in hun implementatie, enkel Gaussiaanse NB en decision tree modellen zijn sneller [25].

3 Componenten van het apparaat

Deze thesis heeft twee doelstellingen: (i) het implementeren van ATC, het automatisch classificeren van technologieën, voor IoT netwerken op een *constrained device*, (ii) het verlagen van de kostprijs van monitoring middelen in vergelijking met huidige oplossingen.

Hier worden de mogelijkheden voor het classificeren van interferentie in sub-GHz bandbreedtes op low-end hardware (*constrained device*) bekeken. Hiervoor zijn er drie functionele componenten noodzakelijk: het meetapparaat, de automatische classificatie en het *constrained device*.

3.1 Netwerken

Deze sectie zal de verschillende netwerktechnologieën bespreken die bestudeerd worden in dit onderzoek. Er worden drie netwerk technologieën onderzocht:

- Zigbee (IEEE802.15.4)
- SigFox
- LoRaWAN

Elk van deze technologieën worden apart bestudeerd en er wordt gekeken of het mogelijk is om deze technologieën te herkennen in homogene radio omgevingen alsook in heterogene omgevingen.

3.1.1 Zigbee

Zigbee is een op IEEE802.15.4 gebaseerd Personal Area Network. Het werkt in kleine gebieden 100m Line of Sight (LoS). Het netwerk biedt verschillende modulatietechnieken in de 868 MHz ISM: Binary Phase Shift Keying (BPSK), Amplitude Shift Keying (ASK) en Quadrature Phase Shift Keying (Q-PSK). De datasnelheden zijn voor de modulatiechnieken respectievelijk: 20 kbps, 100 kbps en 250 kbps [26].

Figuur 5: Zigbee logo

3.1.2 SigFox

SigFox is een ultra-narrow band sub-GHz netwerk ontwikkeld door het bedrijf SigFox. Het bedrijf startte met de uitbating van het netwerk in 2009. De uitbating van he netwerk gebeurt strikt onder toezicht van het bedrijf, hiervoor werkt het samen met netwerkoperatoren die actief zijn in de regio's waarin men het netwerk wil uitbaten.

Figuur 6: SigFox logo

Het netwerk werkt in Europa binnen de ISM band op 868 MHz. Afhankelijk of men uplink of downlink dataverkeer nodig heeft, worden er verschillende modulatie technieken gebruikt. Voor downlink verkeer wordt er Gaussian Frequency-Shift Keying (GFSK) gebruikt en is de beschikbare bandbreedte 500 Hz (er zijn 400 kanalen beschikbaar in de 200kHz band van 868.034 - 868.226 MHz). De mogelijke datasnelheid is 600 [27].

Voor uplink verkeer wordt gebruik gemaakt van Differential Binary Phase-Shift Keying (D-BPSK) die in dezelfde ISM band als uplink werkt (868 MHz). De beschikbare bandbreedte reikt van 869.429 - 869.621 MHz. Voor de uplink zijn er 2 mogelijk baud rate mogelijk, hiervoor kan men kiezen tussen 100 of 600 [27].

3.1.3 LoRaWAN

LoRaWAN is ontwikkeld door Cycleo waarna het door Semtech en de LoRa Alliance (opgericht door Semtech) verspreid en beheerd wordt. Het uitbaten van het netwerk kan door iedereen gedaan worden. Zo zijn er verschillende crowd sourced netwerken (zoals The Things Network) als commerciële netwerken (Proximus LoRaWAN).

Figuur 7: LoRaWAN logo

Het netwerk maakt gebruik van de ISM bandbreedte 864.1 - 868.5 MHz. Er zijn twee mogelijke bandbreedte mogelijk voor de transmissie: 125kHz of

250 kHz. Daarnaast gebruikt het netwerk Chrip Spread Spectrum (CSS). De beschikbare datasnelheid is afhankelijk van de Spreading Factor (SF). Door de SF aan te passen kan men een compromis sluiten tussen datasnelheid en reikwijdte van het netwerk, zo biedt SF12 250 bps aan met het grootste bereik (15 km) en SF7 5.4 kbps aan met het kleinste bereik (5 km) [28].

3.2 Meetapparatuur

Voor het meten en waarnemen van de radiosignalen dient gebruik gemaakt te worden van flexibele, goedkope meetapparatuur. Deze meetapparatuur kan bij voorkeur I/Q samples opmeten, maar RSSI samples zouden ook goede resultaten moeten opleveren. Voor dit doeleinde zijn de opkomende SDR dongles uitermate geschikt. Deze dongles bieden de opties om de waarnemende bandbreedte zelf te kiezen en bepaalde parameters flexibel in te stellen.

De verschillende mogelijkheden voor de SDR dongle worden weergeven in tabel 1.

Tabel 1: Kenmerken van SDR dongles

Device	Samplerate	Bandwidth	Frequency range	Price
RTL-SDRv3	2MSPS	2.4MHz	500kHz-1766MHz	21.95 USD
NE NESDR SMArt SDR		2.4MHz	25MHz - 1750MHz	23.95 USD
NE NESDR Mini SDR and DVB-T			25MHz - 1750MHz	18.95 USD
HackRF One	20 MSPS	20 MHz	1MHz - 6000MHz	284.95 USD

De RTL-SDRv3 maakt gebruik van de R820T2 en RTL2832U chips voor het ontvangen en verwerken van de radiosignalen. Alsook heeft deze een SMA connector voor het aansluiten van een externe antenne. Met deze externe antennes kan de ontvangst in het gewenst radiospectrum verbeterd worden. De NooElec NESDR SMArt SDR lijkt heel sterk van uiterlijk en specificaties op de RTL-SDRv3. Deze gebruiken namelijk exact dezelfde chipset en aansluitingen. De NE NESDR Mini SDR and DVB-T is een compactere versie van de NooElec SMArt SDR. Deze heeft minder mogelijkheden qua configuratie en heeft een CMX antenne connector.

De HackRF one is een geavanceerde SDR. Met een uitgebreide frequentieband en bandbreedte, ondersteuning voor verschillende antennetypes, interne versterkers en transmissie mogelijkheden is dit apparaat in staat om elk mogelijk scenario aan te kunnen. De HackRF One is dan ook de duurste optie op vlak van USB SDR dongles.

Als uiteindelijke keuze werd beslist om de RTL-SDRv3 aan te kopen en te gebruiken. Deze SDR dongle heeft globaal gezien goede specificaties voor onze doeleinden en geniet van een grote community ondersteuning. Daarnaast bestaan er Python modules die ons toegang geven tot deze SDR dongle. Om de RTL-SDRv3 beter af te stemmen op de ISM bandbreedte die wij zullen gebruiken, wordt deze aangevuld met een antenne afgestemd op de 860MHz.

3.3 Automatische classificatie

Deze component zal de gemeten invoer verwerken met als doel IoT netwerk te classificeren. Hiervoor zal het gebruik maken van ML methoden, specifiek neurale netwerken. Om dit te bekomen zullen er ML modellen opgebouwd en getraind worden op basis van IQ-samples of RSSI metingen.

Er zijn verschillende *frameworks* beschikbaar om ML te ontwikkelen in verschillende talen. Zo bestaat er *golearn, goml* en *neural-go* voor Golang en *rusty-machine* voor Rust. Voor elk van deze taal bestaan er ook *Tensorflow bindings* die het mogelijk maken om Tensorflow te gebruiken in deze talen [29], [30].

Echter wordt in het *data science* veld vooral gebruik gemaakt van Python in combinatie met Tensorflow/Keras. Dit komt voornamenlijk door het bestaan van Numpy, die het heel gemakkelijk maakt om matrix operaties uit te voeren op grote datasets. Daarnaast wordt door het gebruik van Jupyter notebooks interactieve databewerkingen heel makkelijk gemaakt.

3.4 Constrained device

Het ML model kan getraind worden op gespecialiseerde hardware om het trainen te versnellen. Het uiteindelijke doeleinde is om dit op *constrained device* te kunnen draaien. Deze hardware zal voornamelijk op de CPU werken, hoewel in de toekomst er hardware beschikbaar komt die deze modellen exponentieel zouden moeten versnellen.

Het *constrained device* moet de SDR dongle kunnen gebruiken en in staat zijn op het ML model te draaien. Hiervoor kan het gebruik maken van puur rekenkracht, of het kan gebruik maken van hardware offloading om deze applicaties te versnellen.

De opties voor een *constrained device* zijn weergeven in tabel 2.

Tabel 2: Specificaties van *constrained devices* [31], [32]

Device	CPU	Clock frequency	GPU	Memory	Price
Raspberry Pi 3B	quad-core ARM Cortex A53	1.2GHz	None	1GB LPDDR2	40 EUR
Raspberry Pi 4	quad-core ARM Cortex A72	1.5GHz	None	4GB LPDDR4	40 EUR
Odroid XU4	Cortex A15	2GHz	None	2GB LPDDR3	70 EUR
NVidia Jetson Nano	quad-core ARM A57	1.43GHz	128-core Maxwell	4GB LPDDR4	99 USD
Dell XPS 13 9360	Intel(R) Core(TM) i7-7560U	2.4GHz	Intel eGPU	16GB LPDDR4	1000 EUR

De Odroid XU4 is krachtiger dan de Raspberry Pi 3B+ en heeft ook dubbel zoveel geheugen ter beschikking. Echter biedt Tensorflow geen officiële ondersteuning voor dit platform. Er zijn dus ook veel minder bronnen te vinden met betrekking hoe we de vereisten hierop installeren, en of deze zelfs ondersteund zijn. De NVidia Jetson Nano was niet beschikbaar tijdens dit onderzoek. Deze lijkt echter een uiterst gepaste kandidaat voor ons doel.

Uiteindelijk is gekozen voor de raspberry Pi 3B+, dit voornamelijk door de officiële ondersteuning die Tensorflow biedt aan dit platform, maar ook vanwege de wijde beschikbaarheid van deze apparaten en de grote ondersteuning en gidsen die te vinden zijn voor dit platform. De NVidia Jetson Nano is een embedded platform dat door NVidia speciaal gemaakt wordt voor ML. De performantie hiervan zou hoger moeten liggen dan de andere 2 soorten. Omdat dit apparaat niet beschikbaar was, is hier geen onderzoek op gebeurd.

4 Implementatie

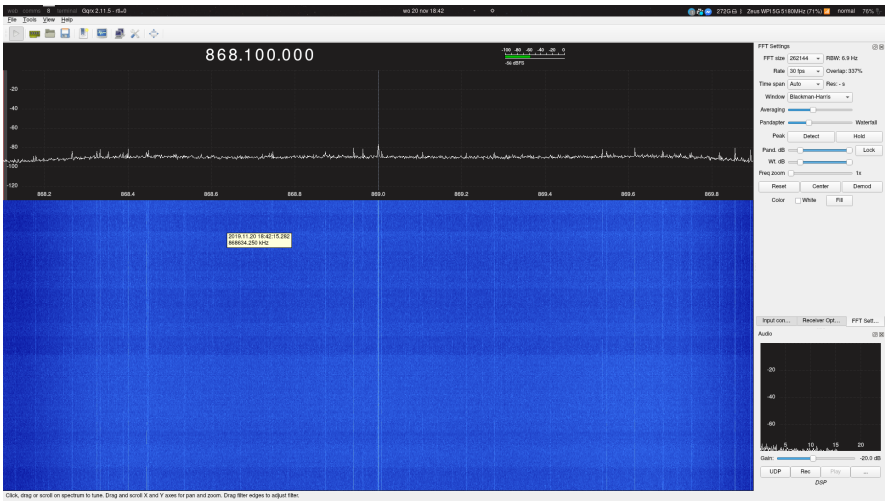
4.1 Dataverzameling

Voor het verzamelen van de dataset wordt gebruik gemaakt van de RTL-SDR dongle en een Dell XPS 13 9360. Via een python script wordt van elke technologie één minuut lang IQ-samples verzameld. Er worden meerdere opnames genomen van elke technologie.

De binnenkomende IQ-samples krijgen minimale preproccesing voor dat ze opgeslagen worden. De pure bytes, die door de SDR verzameld worden, worden omgezet naar complexe getallen. Hierna worden de samples opgeslagen in een .bin bestand.

Voor elke technologie wordt er op twee manieren een dataset opgebouwd. Voor de eerste manier verbinden we de transmitters van de technologie rechtstreeks met de RTL-SDR via een SME kabel. Op deze manier sluiten we externe storingsfactoren uit en krijgen we zuivere signalen binnen van de transmitter.

De tweede manier houdt in dat we signalen opvangen door het verzenden via antennes. Dit zijn dus metingen zoals ze in praktijk zouden gebeuren. Hierdoor kunnen we een dataset opbouwen die meer de realiteit voorstelt, maar wel onderhevig kan zijn aan externe storingsfactoren. Voor we dus starten met de metingen, gebruiken we de GQRX software om te verifiëren dat er geen storende elementen in het spectrum aanwezig zijn.



Figuur 8: GQRX software suite

4.2 Datasets

Er worden datasets opgesteld om de modellen op te trainen: een *eenvoudige* dataset waarbij alle technologieën apart opgemeten worden.

De gebruikte technologieën in de datasets zijn LoRaWAN (SF7 en SF12), SigFox en Zigbee. Hun specificaties worden weergeven in tabel 3.

Tabel 3: Gebruikte technologieën in de datasets

Technologie	Frequentieband	Bandbreedte	Modulatie
LoRaWAN	864.1-867.5MHz	125/250kHz	CSS
Sigfox	868.18-868.22MHz	100kHz	DBPSK
Zigbee	868.0-868.6MHz	600kHz	BPSK-Q-PSK-ASK

Voor elk sample in de meting classificeren we het sample, als een signaal van de technologie, vanaf dat 1 waarde van het sample boven de drempelwaarde ligt. Deze drempelwaarde wordt bepaalt door de noise floor te bepalen op basis van een meting van enkel ruis.

4.3 Training

Het model wordt getraind op de verzamelde dataset. Hiervoor wordt gebruik gemaakt van de Jupyterhub en het GPU-lab van UGent.

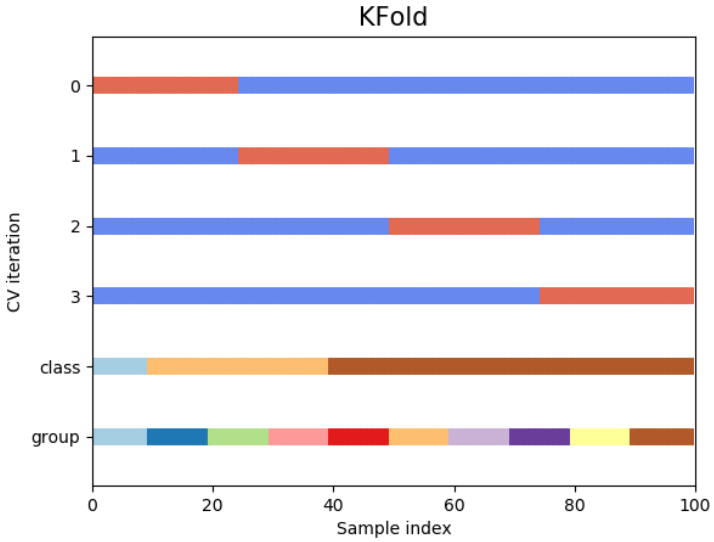
Voor het gebruik van IQ-samples en RSSI is er een Jupyter notebook opgesteld. Deze notebooks verwerken de input van de datasets om uiteindelijk omgevormd te worden tot een Nx2x512 of Nx1x512 vorm (voor IQ of RSSI respectievelijk). Deze matrices worden dan gebruikt als input voor een CNN model.

Het model wordt getraind om 512 meetpunten (= 1 sample) te verwerken. Hiervoor wordt de binnenkomende data opgesplitst in "buckets" van 512 meetpunten en genormaliseerd. De genormaliseerde data wordt gecategoriseerd en aan het model gegeven om te trainen.

Van elke technologie wordt een gelijk aandeel van samples geselecteerd, zodat de trainingset gebalanceerd is. De overige bestaande samples worden gebruikt als validatieset.

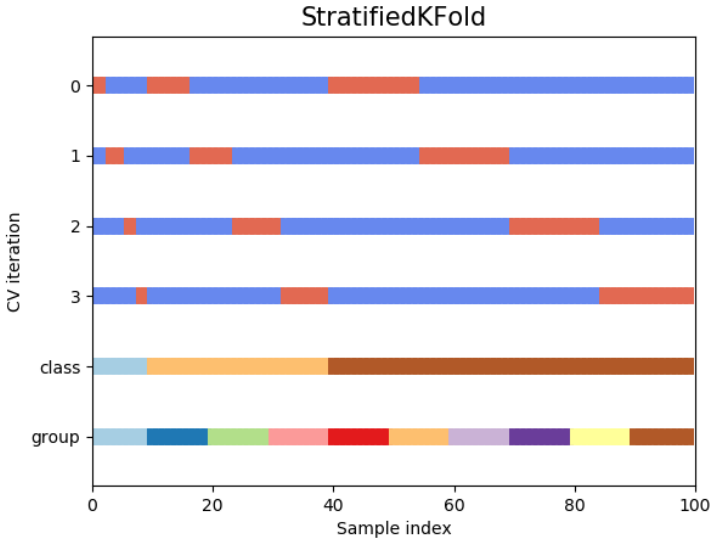
Voor het trainen wordt gebruik gemaakt van Kfold op de trainingset. Gezien deze trainingset al geshuffled is tijdens het samenstellen, is het niet nodig om stratified KFold training te gebruiken. KFold training helpt met het cross valideren en vermijdt dat men zou overfitten op dezelfde trainingdata elke keer men het netwerk traint.

KFold doet dit, zoals in figuur 9 te zien is, door bij elke trainingiteratie een andere subset te nemen van de trainingset als trainingdata en de overige samples te gebruiken als validatie tijdens de training.



Figuur 9: KFold training methode [33].

Stratified KFold (fig. 10) is een variatie op KFold die men kan gebruiken om rekening te houden met de categorieën van de dataset. Stratified past KFold toe op elke categorie die in de dataset zit, waardoor men elke categorie gelijk behandelt. Gezien er evenveel datapunten van elke categorie in de tainingset zijn en deze al geshuffled zijn, moet men niet noodzakelijk stratified KFold gebruiken [33].



Figuur 10: stratified KFold training methode [33].

4.4 Convolutional Neural Network model

Deze sectie bespreekt de implementering en training van het CNN. Voor de implementatie van het CNN wordt er gekeken naar twee mogelijke opties: 1.) CNN op basis van de IQ-samples ofwel 2.) CNN op basis van de RSSI waarden (samengesteld uit de IQ-samples).

Eerst worden de IQ CNNs overlopen, waarna men de RSSI CNNs overloopt. Telkens worden er twee modellen besproken:

Merkbaar is dat het aantal trainbare parameters voor het RSSI model altijd ongeveer de helft is van het IQ model. Dit heeft te maken met de input van deze modellen. Het IQ model heeft een 2x512 input vector, terwijl dit bij het RSSI model een 1x512 vector heeft.

- Complex model: een groot model met voor het IQ model 35,015 trainbare parameters en 17,919 voor het RSSI model.
- Eenvoudig model: een klein model met voor het IQ model 4,623 trainbare parameters en 2,719 voor het RSSI model.

De gebruikte termen hier zijn *relatief*, "complex" en "eenvoudig" zijn relatief ten opzichte van het aantal trainingssamples die beschikbaar zijn.

4.4.1 IQ gebaseerd model

De input van het IQ gebaseerd model is Nx2x512 matrix, deze matrix bezit N samples. Elk sample bestaat uit 512 meetpunten, elk meetpunt heeft twee waarden:

1. I waarde
2. Q waarde

De output van het IQ gebaseerd model is een 1x5 vector die de kans voorstelt dat het ingevoerde sample tot één van de volgende categorieën behoort (nummer komt overeen met de index van de output vector):

0. ruis
1. Zigbee
2. SigFox
3. LoRaWAN SF7
4. LoRaWAN SF12

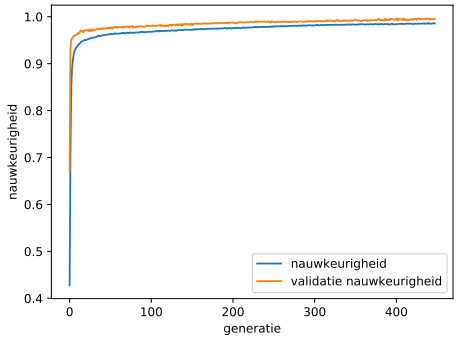
De tussenliggende lagen van het IQ model worden gevarieerd om te bepalen welke complexiteit nodig is om een goede performantie te bekomen.

Complexe modellen hebben meer parameters die een rol kunnen spelen in het bepalen van de categorieën. Echter nemen deze meer ruimte van het apparaat in beslag. Gezien in deze thesis voornamelijk gefocust wordt op het toepassen op *constrained devices*, moet er gekeken worden om de complexiteit zo laag mogelijk te houden, terwijl de impact op nauwkeurigheid minimaal moet blijven.

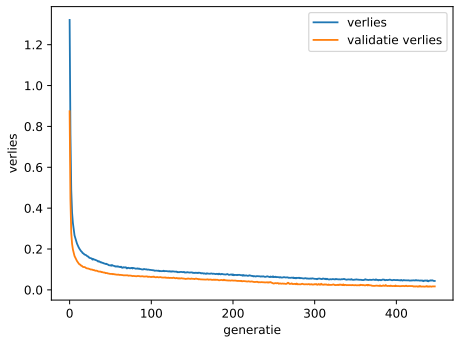
4.4.1.1 Eenvoudig model

Het eenvoudig model (fig.13) is opgbouwd uit drie keer een 2D convolutionele laag en direct daar achter een 2D max pooling laag. Na deze lagen wordt de volledige structuur geflattened waarna 2 densely connected layers (met een dropout van 30% tussen de 2 lagen) ervoor zorgen dat we 5 output waarden krijgen (1 waarde voor elke categorie).

In figuur 11 kunnen we zien dat dit model op de trainingsdataset een nauwkeurigheid van 99% behaalt.



(a)

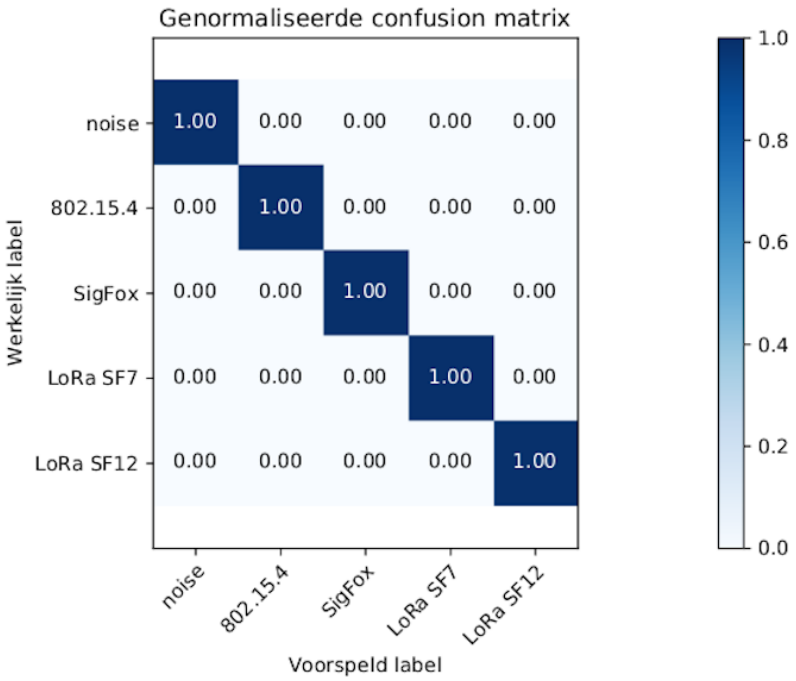


(b)

Figuur 11: training statistieken voor eenvoudig IQ model: a.) nauwkeurigheid b.) verlies

Validatie van het model op een validatiedataset levert een uitstekende nauwkeurigheid. Op metingen die op dezelfde plaats uitgevoerd zijn met

dezelfde meetopstelling worden er geen fouten gemaakt (fig. 12).

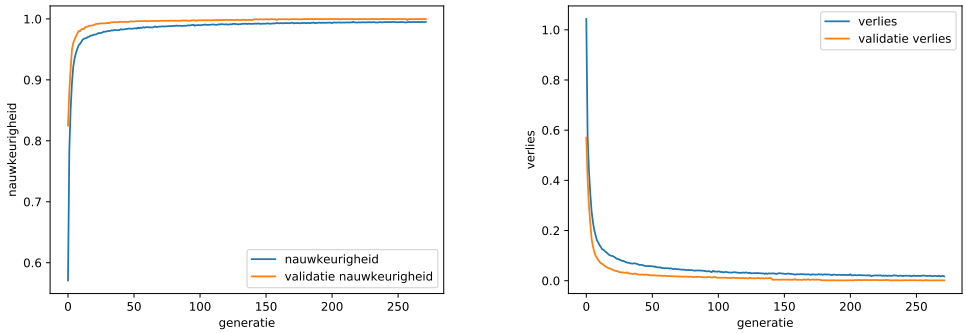


Figuur 12: Genormaliseerde confussion matrix voor eenvoudig IQ model

4.4.1.2 Complex model

Het complex model (fig.16) is gebaseerd op het eenvoudig model. Hierbij zijn de 2D max pooling lagen na de 2D convolutionele lagen weggelaten waardoor er meer parameters aanwezig zijn. De 2D maxpooling lagen verlaagden dus de complexiteit van het model.

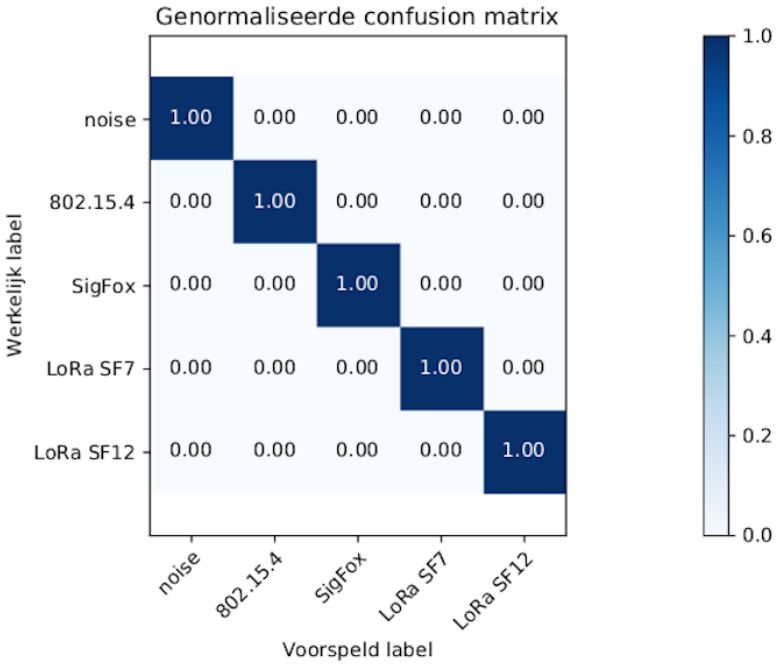
Zoals bij het eenvoudig IQ model zien we ook hier dat het model een hoge nauwkeurigheid van 99% kan behalen (fig. 14).



(a) (b)

Figuur 14: training statistieken voor complex IQ model: a.) nauwkeurigheid b.) verlies

Ook bij dit model is bij de validatie een heel goede nauwkeurigheid aanwezig (fig.15). Gezien dat het onderscheid tussen de twee IQ modellen dus miniem is, zal de performantie van de modellen een bepalende factor spelen in het kiezen welk model men moet gebruiken.



Figuur 15: Genormaliseerde confussion matrix voor complex IQ model

4.4.2 RSSI CNN modellen

De input voor het RSSI model is een Nx1x512 matrix dat N samples bezit. Een sample bestaat uit 1x512 vector dat 512 meetpunten bezit. Elk meetpunt is een RSSI waarde dat berekend is door de IQ-sample te gebruiken om de RSSI waarde te berekenen.

Voor de RSSI modellen moeten eerst de IQ-samples gecombineerd worden tot de RSSI waarde. In de onderstaande formule stelt I de I-waarde voor en Q de Q-waarde van het IQ-sample.

$$RSSI = \sqrt{I^2 + Q^2}$$

Gezien men toch rekenkundig omgaat met deze waarden, is het mogelijk om de volgende formule te gebruiken om de waarde te benaderen. Dit is

beter om de rekenkracht die nodig is zo klein mogelijk te houden voor de toepassing op een *constrained device*.

$$RSSI^2 = I^2 + Q^2$$

Na de waardes te combineren, kan men een 1x512 vector als input gebruiken voor het CNN model.

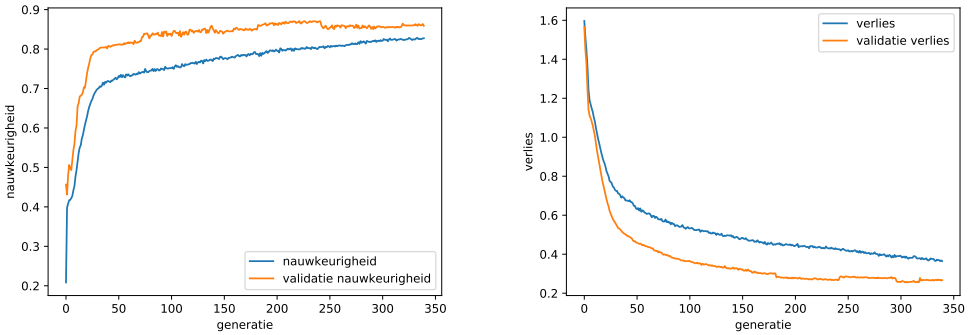
De output is dezelfde als bij de IQ CNN modellen. De output van het model stelt de kans voor dat de ingevoerde sample één van de vijf klassen is:

0. ruis
1. Zigbee
2. SigFox
3. LoRaWAN SF7
4. LoRaWAN SF12

4.4.2.1 Eenvoudig model

Het eenvoudig model (fig.19) is opgbouwd uit drie keer een 2D convolutionele laag en direct daar achter een 2D max pooling laag. Na deze lagen wordt de volledige structuur geflattened waarna 2 densely connected layers (met een dropout van 30% tussen de 2 lagen) ervoor zorgen dat we 5 output waarden krijgen (1 waarde voor elke categorie).

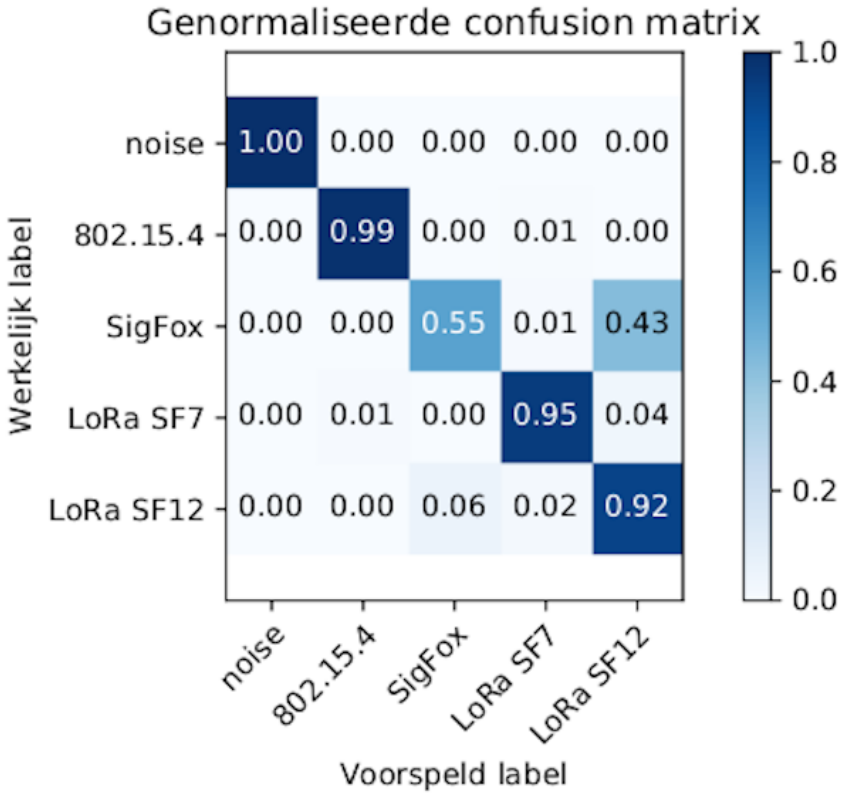
Uit figuur 20 is af te leiden dat het eenvoudig RSSI model een nauwkeurigheid van 83% behaalt. Dit komt voornamelijk doordat dit model een laag aantal trainbare parameters heeft en dat uit een RSSI signaal minder informatie te halen is dan uit IQ-samples.



(a) (b)

Figuur 17: training statistieken voor eenvoudig RSSI model: a.) nauwkeurigheid b.) verlies

Op de confussion matrix van het eenvoudig RSSI model (fig.18) is te zien dat er een beetje misclassificatie gebeurt tussen LoRaWAN SF7 en SF12. Dit is logisch gezien dit dezelfde technologie is, enkel de eigenschappen van de transmissies zijn aangepast. De grootste misclassificatie gebeurt echter tussen SigFox en LoraWAN SF12. Dit komt doordat het RSSI profiel van deze technologieën heel sterk op elkaar lijkt. Beide (SigFox en LoRaWAN SF12) hebben ongeveer dezelfde bandbreedte (100kHz en 125kHz).

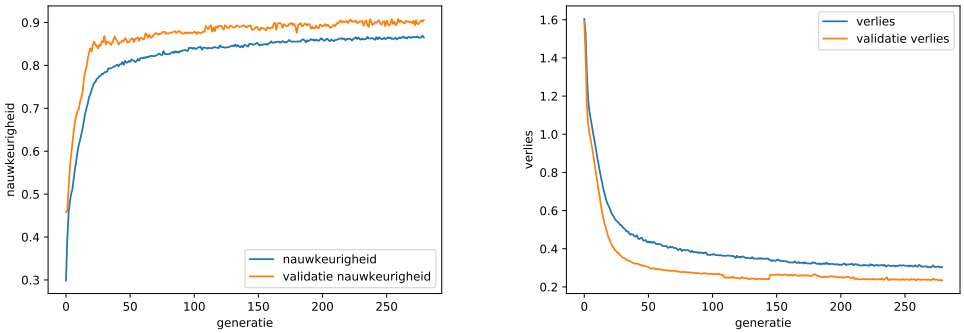


Figuur 18: Genormaliseerde confussion matrix voor eenvoudig RSSI model

4.4.2.2 Complex model

Het complexe model (fig.22) is gebaseerd op het eenvoudige model. Hierbij zijn de 2D max pooling lagen na de 2D convolutionele lagen weggelaten waardoor er meer parameters aanwezig zijn. De 2D maxpooling lagen verlaagden dus de complexiteit van het model.

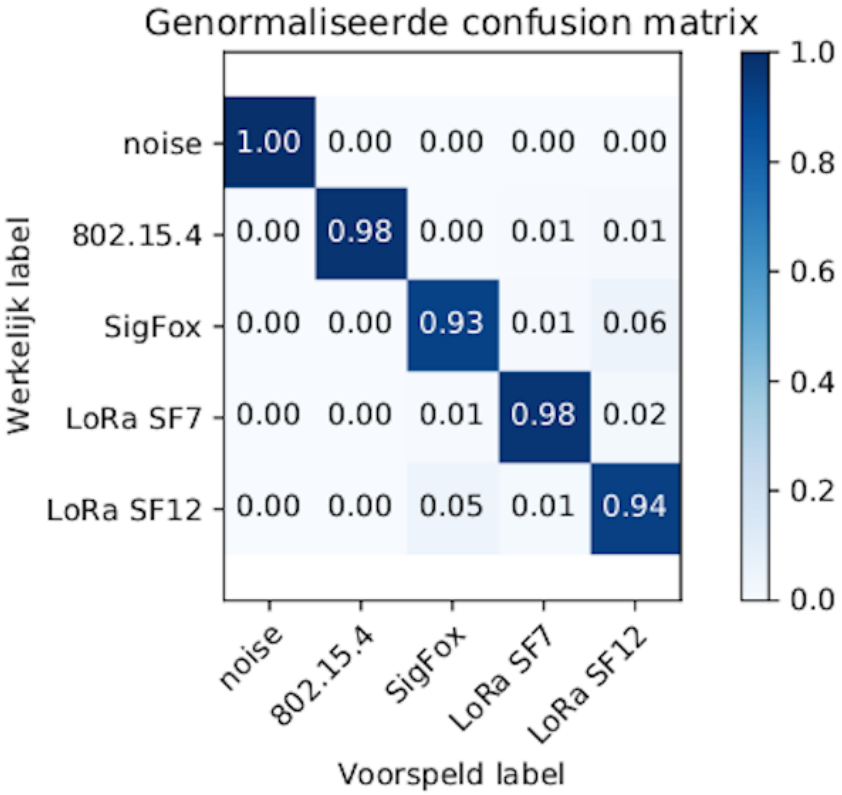
In figuur 20 is te zien dat het eenvoudig RSSI model een nauwkeurigheid van 86% behaalt.



(a) (b)

Figuur 20: training statistieken voor eenvoudig RSSI model: a.) nauwkeurigheid b.) verlies

Opvallend is dat het complex RSSI model wel in staat is om het onderscheid te maken tussen SigFox en LoRaWAN SF12 (fig. 21). De hogere complexiteit van het model heeft dus hier wel een voordeel (in tegenstelling tot de IQ modellen).



Figuur 21: Genormaliseerde confussion matrix voor complex RSSI model

4.5 Uitvoering van de modellen

Voor het uitvoeren van de modellen op de *constrained devices* wordt gebruik gemaakt van een Python script om de flexibiliteit te bewaren. Een andere mogelijkheid zou zijn om de modellen om te zetten naar een gecompileerde versie (waardoor deze in theorie sneller zouden zijn, hoewel dit niet verzekerd is).

Er worden twee versies uitgewerkt van het Python script.

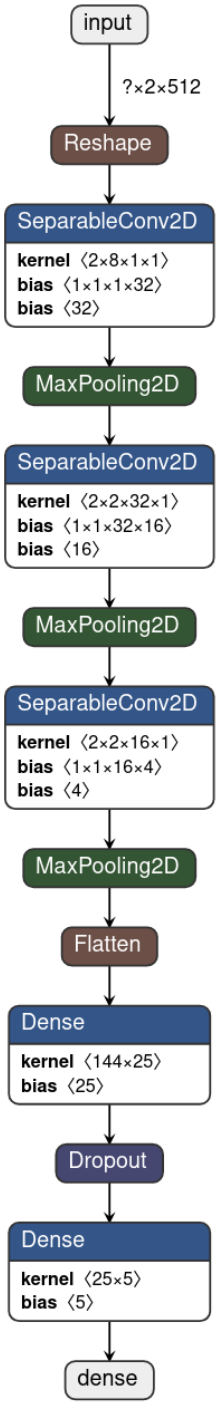
- Uitvoerings script: dit script voert enkel het model uit en publiceert de resultaten in een JSON formaat. Dit maakt het mogelijk om de resultaten te verzenden via web technologie.

- Benchmarking script: dit script voert niet enkel het model uit. Dit script exporteert ook de uitvoeringstijden en resultaten naar een CSV. Hierdoor is het mogelijk om nadien een statistische analyse uit te voeren op de behaalde resultaten.

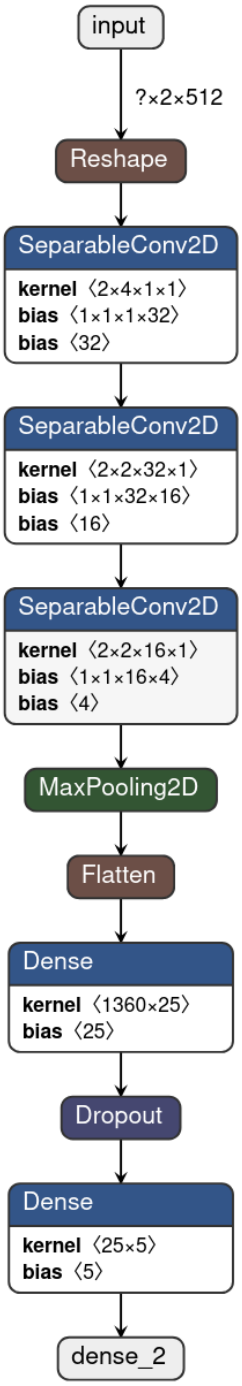
De modellen worden geïmporteerd in één bestand waarin zowel de architectuur als de gewichten van het model opgeslagen zit. Dit zorgt ervoor dat het model heel draagbaar is, mits de inruiling van de flexibiliteit dat men verschillende gewichten kan gebruiken per gewicht.

4.5.1 Raspberry Pi

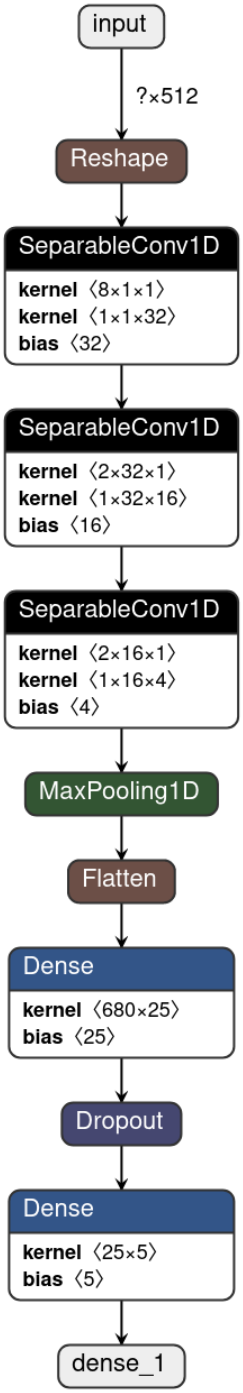
Sinds kort ondersteunt Tensorflow (vanaf versie 1.9) de Raspberry Pi zelf, Tensorflow en Keras kunnen eenvoudig geïnstalleerd worden op de Raspberry Pi. Om Tensorflow te installeren op Raspbian Stretch (Debian 9) kan men, nadat men de dependencies heeft geïnstalleerd, met sudo apt install libatlas-base-dev, pip3 install Tensorflow Keras gebruiken. Dit zorgt ervoor dat Tensorflow en Keras beschikbaar worden op het systeem, waardoor men gemakkelijk modellen en gewichten kan doorsturen naar de Raspberry Pi via de Keras save_model en load_model.



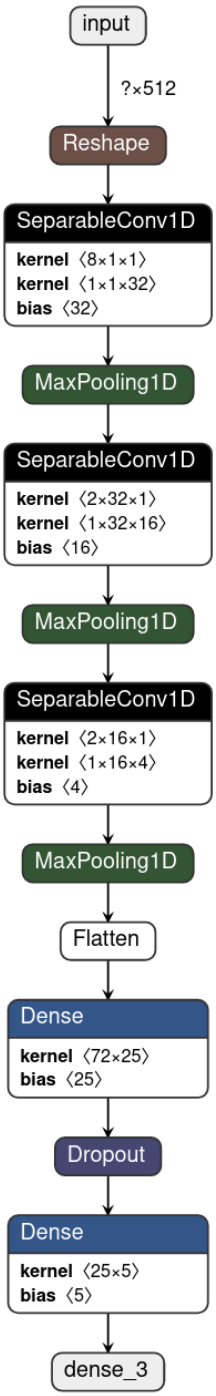
Figuur 13: Architectuur van het eenvoudig IQ model



Figuur 16: Architectuur van het complex IQ model



Figuur 19: Architectuur van het eenvoudig RSSI model



Figuur 22: Architectuur van het complex RSSI model

5 Impact van parameters

In dit hoofdstuk zal de impact van verschillende parameters van het identificatiescript op de verwerkingsnelheid besproken worden. Het doel hiervan is om het ideale punt van elke parameter te bepalen zodat men de hoogste verwerkingssnelheid kan bekomen.

Om de performantie van de constrained devices te meten, wordt de volgende methodiek gebruikt. Het model wordt als JSON geëxporteerd van de server waarop het getraind is, de gewichten van het model worden als H5 geëxporteerd. Deze twee bestanden worden dan op het geheugen van het testapparaat geplaatst, waarna ze ingeladen worden in de code die op het testapparaat draait.

5.1 Verwerkingssnelheid - aantal verwerkte samples

Deze sectie bespreekt de impact van het aantal verwerkte samples tegenover de sample verwerkingssnelheid. Een sample wordt gedefinieerd als een meting van 512 waarden (RSSI of IQ) dat een periode van 512μs meet.

Er wordt gekeken naar de verwerkingssnelheid wanneer men 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192 en 15000 samples per meting gebruikt. Voor elke meting wordt in het model een *batch size* (aantal samples die iteratie verwerkt worden) van 32 gebruikt. Een meting die 128 samples heeft zal dan in 4 iteraties verwerkt zijn. De bovengrens van 15000 ligt hier vast door het maximum aantal waarden dat men in één keer kan uitlezen van de RTL-SDR. Elk samples heeft een lengte van 16.384 ms.

De verwerkingsnelheid wordt vastgelegd als het aantal samples per seconde die verwerkt kunnen worden. Om de verwerkingssnelheid te berekenen, wordt elke meting 100 keer uitgevoerd. De eerste 5 metingen worden gezien als "warm up", hierdoor introduceren deze metingen buitenliggers (extreme waarden). In praktische toepassingen kan men deze niet negeren. De eerste keer zal de uitvoering langer duren vanwege de overhead van het inladen van het model en het opzetten van de initiële communicatie met de SDR.

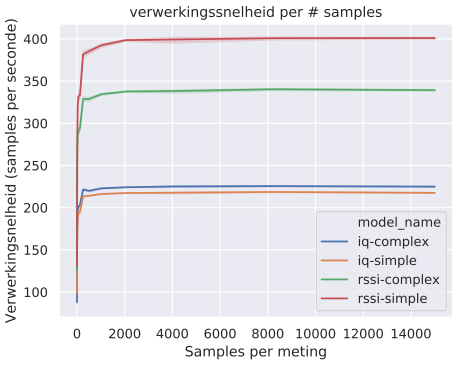
Van de overige 100 metingen wordt de gemiddelde uitvoeringstijd berekend, deze wordt dan gedeeld door het aantal verwerkte samples. Dit heeft ons de gemiddelde uitvoeringstijd per sample. Door deze te inverteren bekomen we de verwerkingsnelheid (samples per seconde):

$$gemiddelde\ verwerkingstijd = \frac{gemiddelde(100\ metingen)}{samples\ per\ meting}$$

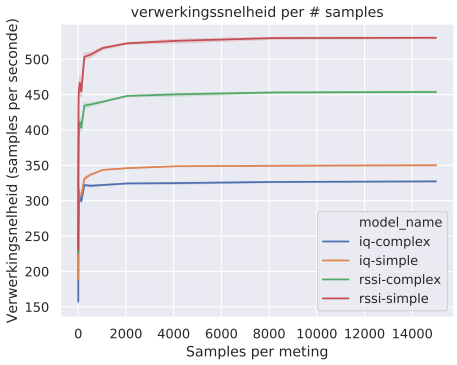
$$gemiddelde\ verwerkingsnelheid = (gemiddelde\ verwerkingstijd)^{-1}$$

Gebaseerd op de bovenstaande formule kunnen we de vergelijking maken voor elk model. Uit figuur 23 en figuur 24 is het mogelijk om af te leiden dat voor elk model dezelfde trend zich toont. De verwerkingssnelheid neemt sterk toe tussen de 2 tot 1024 samples per meting. Daarna, wanneer men meer dan 1024 samples per meting gebruikt, is de toename in de verwerkingsnelheid steeds kleiner.

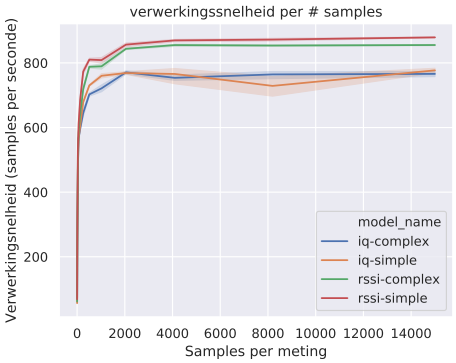
Optimaal gebruikt men dus 1024 of meer samples per meting om de hoogste verwerkingssnelheid te bekomen.



(a) Raspberry Pi 3

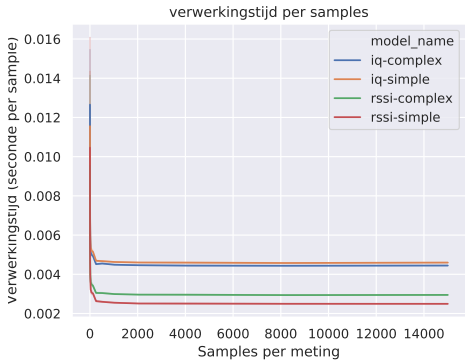


(b) Raspberry Pi 4

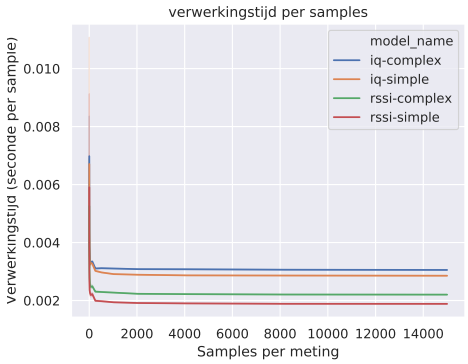


(c) Dell XPS 13

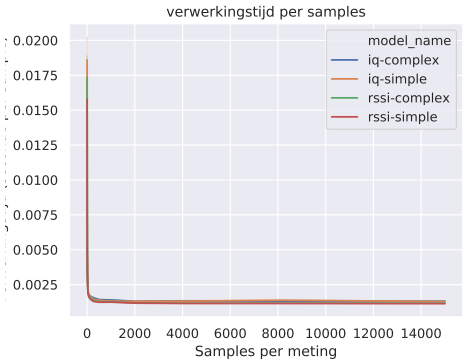
Figuur 23: Verwerkingssnelheid per samples



(a) Raspberry Pi 3



(b) Raspberry Pi 4



(c) Dell XPS 13

Figuur 24: Verwerkingstijd per samples

5.2 Verwerkingssnelheid - batch size

In deze sectie wordt de impact van de *batch size* van het model besproken. De *batch size* dicteert hoeveel samples het model tegelijkertijd kan verwerken. De standaard *batch size* van Keras modellen is 32.

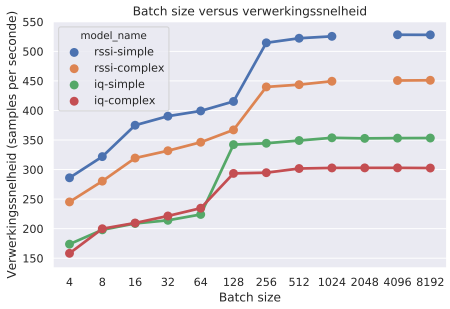
Dezelfde methodiek als bij 5.1 wordt hiervoor gebruikt, echter veranderen we nu enkel de *batch size* over de iteraties. In hoofdstuk 5.1 is afgeleid dat 1024 samples het keerpunt is voor snellere sample verwerking. Er worden voor elke iteratie 1024 samples gebruikt en de verschillende iteraties hebben respectievelijk de volgende *batch sizes:* 1, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 en 8912.

Op figuur 25 is te zien dat de verwerkingssnelheid ook toeneemt naarmate

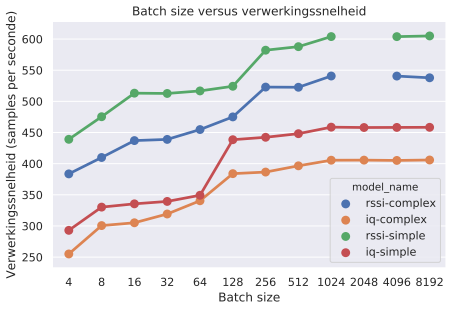
men de *batch size* kan matchen aan het aantal samples. Wanneer de *batch size* groter wordt dan het aantal samples, neemt de verwerkingssnelheid niet meer toe.

Er moet rekening gehouden worden met het feit dat als men de *batch size* vergroot, er meer geheugen nodig zal zijn. Zo kan men op subfiguur (a), (b) en (c) zien dat er voor bepaalde *batch sizes* geen datapunten beschikbaar zijn. Bij deze *batch sizes* nam, op alle apparaten, het programma meer geheugen in dan toegelaten.

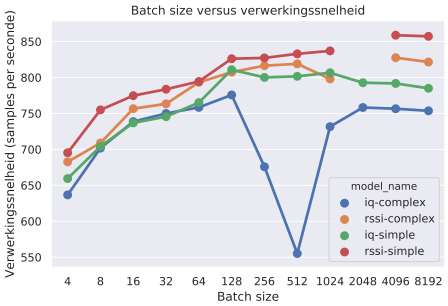
Hieruit kan men afleiden dat, zolang het geheugen van het apparaat het toelaat, men best de *batch size* matched aan het aantal samples. Dit levert de hoogste verwerkingssnelheid op.



(a) Raspberry Pi 3



(b) Raspberry Pi 4



(c) Dell XPS 13

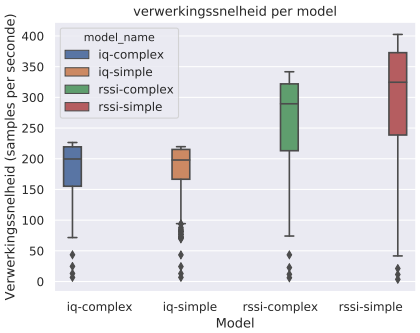
Figuur 25: Verwerkingssnelheid bij verschillende batch sizes

5.3 Verwerkingssnelheid - complexiteit

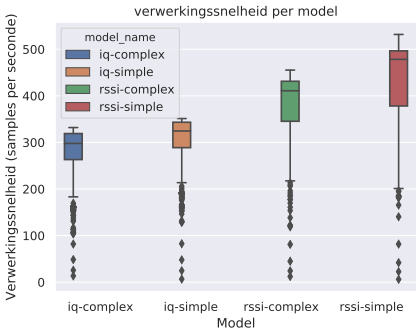
Uiteindelijk wordt de impact bekeken van de complexiteit van de verschillende modellen. Complexiteit wordt hier gedefinieerd als het aantal trainable parameters dat het model heeft. Dit kan berekend worden door Keras zelf wanneer men een *summary* van het model laat printen. Dit is ook mogelijk in de *identifier.py* code door gebruik te maken van de *summary* flag.

De complexiteit van het model wordt niet strikt noodzakelijk bepaald door het aantal lagen van het model, meer door de volgorde waarin deze lagen opgebouwd worden. Zo kunnen convolutionele lagen gevolgd door een max pooling laag de complexiteit van een model verminderen.

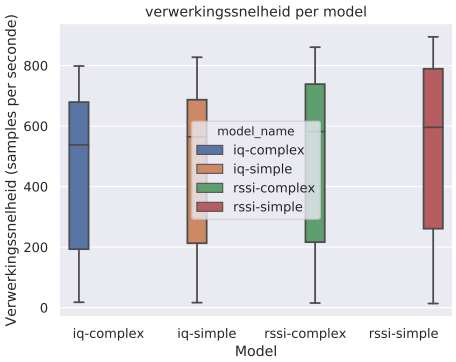
In figuur 26 is te zien dat globaal gezien de RSSI modellen sneller werken dan de IQ modellen. Hieruit valt af te leiden dat de voorverwerking (het berekenen van de RSSI waarden) die men voor elke meting moet doen, geen grote impact heeft. De lagere complexiteit van deze modellen (door de kleinere input grootte) weegt meer door, waardoor de modellen een hogere verwerkingssnelheid hebben.



(a) Raspberry Pi 3



(b) Raspberry Pi 4



(c) Dell XPS 13

Figuur 26: Verwerkingssnelheid per complexiteit van het model

5.4 Resolutie van de meting

Uit de voorgaande sectie kan men concluderen dat 1024 of meer samples per meting de beste verwerkingssnelheid oplevert. Echter moet men waarnemen dat met meer samples per meting, er ook langere periodes gewacht moet worden voor men weer een meting kan doen.

Wanneer men 8 samples per meting neemt, dan duurt de meting 4.096 ms. De verwerking van deze meting duurt dan in totaal 30.315 ms. Voor een meting van 128 samples, en dus een lengte van 65.536 ms, duurt de verwerking 416.532 ms.

In tabel 4 kunnen we zien dat voor een groot aantal samples bv. 4096 samples, er dan uiteindelijk 18.196 s verwerkingstijd nodig is. Hierdoor kan

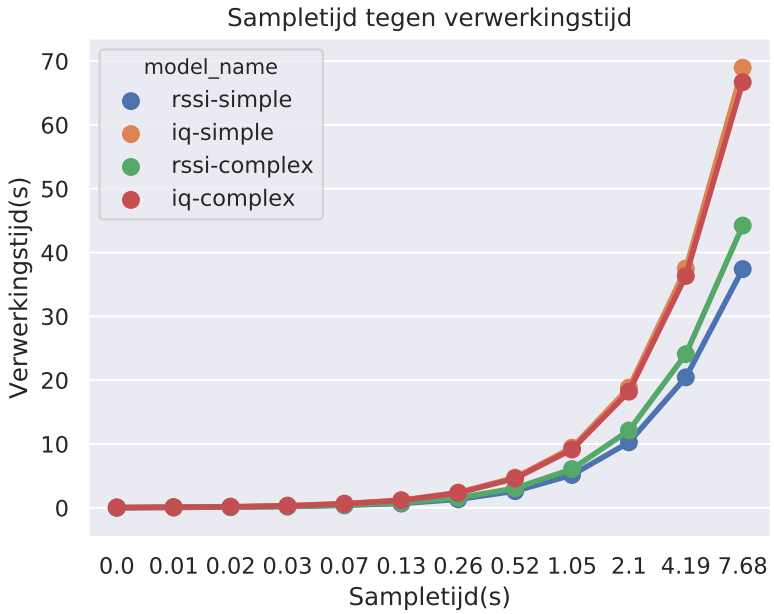
men dan maar elke 18.196 s een meting uitvoeren.

Tabel 4: Sampletijd versus verwerkingstijd

model naam	aantal samples	sampletijd	gem. verwerkingstijd	gem. totale verwerkingstijd
iq-simple	4	0.002048	0.036144	0.038192
iq-complex	4	0.002048	0.033602	0.035650
rssi-simple	4	0.002048	0.021301	0.023349
rssi-complex	4	0.002048	0.024559	0.026607
...	...	...	...	...
iq-simple	128	0.065536	0.592670	0.658206
iq-complex	128	0.065536	0.565471	0.631007
rssi-simple	128	0.065536	0.320871	0.386407
rssi-complex	128	0.065536	0.371235	0.436771
...	...	...	...	...
iq-simple	4096	2.097152	16.722148	18.819300
iq-complex	4096	2.097152	16.098638	18.195790
rssi-simple	4096	2.097152	8.1629663	10.260118
rssi-complex	4096	2.097152	10.016068	12.113220

We kunnen in figuur 27 zien dat de verwerkingstijd dus altijd verhoogt met de gemeten sampletijd.

Dit heeft als gevolg dat er gekeken moet worden naar een compromis waar het aantal samples hoog genoeg ligt zodat men een grote verwerkingssneilheid heeft, maar de resolutie goed genoeg is dat men geen kritieke momenten verliest tijdens de verwerking. Hiervoor ligt de waarde van 1024 samples per meting op een ideaal punt. De verwerkingssnelheid benadert de boven grens die mogelijk is en men kan elke 3.190 s een meting uitvoeren van 500 ms.



Figuur 27: Sampletijd tegenover verwerkingstijd

6 Verder onderzoek

Dit hoofdstuk bespreekt de verdere onderzoeksmogelijkheden die mogelijk zijn op basis van de problemen en het onderzoek in deze thesis.

6.1 LPWAN datasets

Het gebrek aan dataset voor ML voor LPWAN netwerken is een grote beperking in het onderzoek naar verdere stappen voor het dynamisch beheer van LPWAN netwerken. Voor beeldverwerking bestaan er vele standaard datasets die gebruikt kunnen worden om ML een basis aan te leren, waarna men het model kan bijsturen voor de specifieke applicatie. Deze standaard datasets ontbreken nog op vlak van "smart radios".

De ideale weg voorwaarts is het aanmaken van een centrale groepering van datasets waarop modellen gestandaardiseerd worden. Echter dient hiervoor een standaard opgesteld te worden voor de metingen. Door deze centralisatie zou het mogelijk worden om modellen te trainen op een gevarieerde set van metingen waardoor deze robuster en consistenter kunnen zijn.

6.2 Online learning

Met online learning wordt bedoeld dat het netwerk getraind wordt terwijl er data binnenkomt in plaats van de batch training met grote datasets. Dit zorgt ervoor dat het netwerk in staat zou zijn om bij te leren en zichzelf aan te passen aan de omgeving waarin het terecht komt.

6.3 Machine Learning on *constrained devices*

Door de ontwikkeling van nieuwe *constrained devices* die specifiek gericht zijn op het gebruik van ML (zoals de NVidia Jetson reeks) zijn deze apparaten uitermate geschikt voor verder onderzoek. Door het gebruik van hardware offloading is het mogelijk om betere en complexere ML modellen te gebruiken die mogelijks betere resultaten opleveren.

Daarnaast kan er gekeken worden om de interne instelling van het gebruikte framework, Tensorflow, nog aan te passen door bv. een lagere precisie te gebruiken voor de interne waarden van het netwerk. Ook mag men kijken om andere hogere niveau frameworks dan Keras te gebruiken (bv. PyTorch).

Ook andere programmeertalen zoals C,C++ of Rust bieden een mogelijke verbetering. Deze talen kunnen gecompileerd worden voor de *constrained devices* waardoor deze een kleinere overhead zullen hebben.

Verder moet de impact van de gemeten tijd op de verwerkingstijd bestudeerd worden. De waargenomen trend is dat een langere gemeten tijd de verwerkingstijd per sample reduceert.

7 Conclusie

In de hedendaagse maatschappij is er steeds meer vraag naar informatie. Hiervoor zal men niet alleen gebruik maken van bestaande netwerken maar ook van de opkomende technologieën. IoT en smart city netwerken spelen hierbij een belangrijke en prominente rol. Deze zullen de radiospectra steeds meer gebruiken. Overvolle spectra leiden tot interferentieverschijnselen zoals verstoorde communicatie, verzwakte signalen en miscommunicatie. Een belangrijk aspect is het beheer van deze IoT netwerken, de handhaving van de regelgeving en performantie.

Deze thesis heeft als doel om een eenvoudig te gebruiken apparaat te ontwikkelen dat in staat is om lokaal interferentie te detecteren en technologie te classificeren. Bovendien werd de kostprijs van dit apparaat zo laag mogelijk gehouden.

Het gebruik van ML technieken op *constrained devices* is een grote uitdaging. Door de beperkte rekenkracht aanwezig op deze apparaten moeten eenvoudigere modellen gebruikt worden. Hierdoor kan men niet altijd de gewenste nauwkeurigheid bereiken.
Door gebruik te maken van een constrained device in combinatie met een SDR, is het mogelijk om met een CNN aan interferentiedetectie te doen, zonder dat het nodig zou zijn dure hardware hiervoor aan te schaffen. Door de CNN op voorhand te trainen op krachtige hardware, kan men nadien dit getraind model op het constrained device laden en de classificatie hierop draaien.

De Raspberry Pi + SDR dongle komen neer op een kostprijs van 130 EUR. De Dell XPS 13 9360 + SDR dongle hebben een kostprijs van 1000 EUR. Wanneer men dit vergelijkt met de kostprijs van een toegewijde spectrum analyzer (2000 EUR) liggen deze prijzen lager. De voornaamste kost van een systeem dat gebruik gemaakt van een goedkoop embedded platform met een SDR dongle, ligt bij de nodige hardware om het ML model te trainen. Hiervoor is namelijk nog steeds specifieke hardware nodig.

De voorgestelde modellen behalen een hoge nauwkeurigheid (89% tot 97% afhankelijk van de complexiteit) en zijn inzetbaar op de Raspberry Pi voor gebruik.

Hedendaagse embedded platformen beginnen een reële optie te worden voor het gebruik van ML, vooral met de opkomst van applicatie specifieke platformen zoals de NVidia Jetson. Deze platformen zullen een innovatie brengen op vlak van het gebruik van ML in vele sectoren. De geteste platformen, Raspberry Pi 3B en Raspberry Pi 4, zijn tegenwoordig al in staat om de modellen te gebruiken.

De dagen dat men voor het gebruik van ML krachtige systemen nodig had met specifieke hardware zijn echter voorbij. De testen met de Dell XPS 13 9360, tonen aan dat een systeem uit de middenklasse tegenwoordig in staat is om de ML te gebruiken. Daarnaast zijn de Raspberry Pi apparaten ook in staat om de 350 samples per seconde te verwerken. Hierdoor zijn deze ook klaar voor gebruik in praktische toepassingen zoals het detecteren van technologieën en interferente met deze technologieën.

Hierbij is het echter wel belangrijk om de limitaties en grenzen van deze apparaten op te zoeken en te kijken of deze voldoen aan de noden van praktische toepassingen. Zo liggen de verwerkingssnelheden op deze *constrained devices* nog te laag om echte real-time monitoring voor LPWAN netwerken uit te voeren, maar kunnen ze wel al gebruikt worden om problemen te verhelpen of te detecteren.

8 Referenties

[1] (). Smart City Componentes. Bezocht op 2019-07-15, [Online]. Available: https://thepracticaldev.s3.amazonaws.com/i/yppgw7amxwo4bivmz6jz.jpg.

[2] N. Golmie and F. Mouveaux, "Interference in the 2.4 ghz ism band: Impact on the bluetooth access control performance," *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, DOI: 10.1109/icc.2001.936608. [Online]. Available: http://dx.doi.org/10.1109/ICC.2001.936608.

[3] C.-H. Yu, K. Doppler, C. Ribeiro, and O. Tirkkonen, "Performance impact of fading interference to device-to-device communication underlaying cellular networks," *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sep. 2009. DOI: 10.1109/pimrc.2009.5450264. [Online]. Available: http://dx.doi.org/10.1109/PIMRC.2009.5450264.

[4] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Networks*, vol. 11, no. 4, pp. 471–487, Jul. 2005, ISSN: 1572-8196. DOI: 10.1007/s11276-005-1769-9. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-1769-9.

[5] J. Tan and S. G. Koo, "A survey of technologies in internet of things," *Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2014*, pp. 269–274, 2014. DOI: 10.1109/DCOSS.2014.45.

[6] C. Goursaud and J. M. Gorce, "Dedicated networks for IoT: PHY / MAC state of the art and challenges," *EAI Endorsed Transactions on Internet of Things*, vol. 1, no. 1, p. 150 597, 2015. DOI: 10.4108/eai.26-10-2015.150597.

[7] (). IoT roadmap. Bezocht op 2019-06-20, [Online]. Available: https://40uu5c99f3a2ja7s7miveqgqu-wpengine.netdna-ssl.com/wp-content/uploads/2016/10/The-Internet-of-Things-from-connecting-devices-to-creating-value-large.jpg.

[8] Temboo, *5 Reasons to Use Sub-GHz for IoT Applications*. [Online]. Available: https://blog.temboo.com/using-sub-ghz-for-iot-applications/ (visited on).

[9] C. Gomez and J. Paradells, "Survey of home automation networks," *IEEE Communications Magazine*, no. June, pp. 92–101, 2010. [Online]. Available: http://www.ann.ece.ufl.edu/courses/eel6935%7B%5C_%7D11fal/papers/Survey%20of%20home%20automation%20networks.pdf.

[10] U. Wetzker, I. Splitt, M. Zimmerling, C. A. Boano, and K. Romer, "Troubleshooting Wireless Coexistence Problems in the Industrial Internet of Things," *Proceedings - 19th IEEE International Conference on Computational Science and Engineering, 14th IEEE International Conference on Embedded and Ubiquitous Computing and 15th International Symposium on Distributed Computing and Applications to Business, Engi*, pp. 98–109, 2017. DOI: 10.1109/CSE-EUC-DCABES.2016.167.

[11] R. Miller, W. Xu, P. Kamat, and W. Trappe, "Service discovery and device identification in cognitive radio networks," *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, pp. 670–677, 2007. DOI: 10.1109/SAHCN.2007.4292880.

[12] N. Bitar, S. Muhammad, and H. H. Refai, "Wireless technology identification using deep convolutional neural networks," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2017-Octob, pp. 1–6, 2018. DOI: 10.1109/PIMRC.2017.8292183.

[13] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. V. Valk, "Machine Learning Approach to RF Transmitter Identification," *IEEE Journal of Radio Frequency Identification*, vol. 2, no. 4, pp. 197–205, 2018, ISSN: 2469-7281. DOI: 10.1109/jrfid.2018.2880457.

[14] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018, ISSN: 2332-7731. DOI: 10.1109/tccn.2018.2835460.

[15] Y. Zhang, S. Bi, M. Dong, and Y. Liu, "The implementation of CNN-based object detector on ARM embedded platforms," *Proceedings - IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, IEEE 16th International Conference on Pervasive Intelligence and Computing, IEEE 4th International Conference on Big Data Intelligence and Computing and IEEE 3*, vol. 50, pp. 379–382, 2018. DOI: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00074.

[16] H. Li, "Multi-agent Q-learning of channel selection in multi-user cognitive radio systems: A two by two case," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, no. October, pp. 1893–1898, 2009, ISSN: 1062922X. DOI: 10.1109/ICSMC.2009.5346172.

[17] S. Zacharias, T. Newe, S. O'Keeffe, and E. Lewis, "2.4 GHz IEEE 802.15.4 channel interference classification algorithm running live on a sensor node," *Proceedings of IEEE Sensors*, pp. 1–4, 2012. DOI: 10.1109/ICSENS.2012.6411279.

[18] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," *Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017*, pp. 180–185, 2017. DOI: 10.1109/INDIN.2017.8104767.

[19] S. Hu, Y. D. Yao, and Z. Yang, "MAC protocol identification using support vector machines for cognitive radio networks," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 52–60, 2014, ISSN: 15361284. DOI: 10.1109/MWC.2014.6757897.

[20] S. A. Rajab, W. Balid, M. O. Al Kalaa, and H. H. Refai, "Energy detection and machine learning for the identification of wireless MAC technologies," *IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference*, pp. 1440–1446, 2015. DOI: 10.1109/IWCMC.2015.7289294.

[21] A. Katidiotis, K. Tsagkaris, and P. Demestichas, "Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems," *Computers and Electrical Engineering*, vol. 36, no. 3, pp. 518–535, 2010, ISSN: 00457906. DOI: 10.1016/j.compeleceng.2009.12.005. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2009.12.005.

[22] S. Oh, M. Kim, D. Kim, M. Jeong, and M. Lee, "Investigation on performance and energy efficiency of CNN-based object detection on embedded device," *Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology, CAIPT 2017*, vol. 2018-Janua, pp. 1–4, 2018. DOI: 10.1109/CAIPT.2017.8320657.

[23] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-End Learning from Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2818794.

[24] S. Grimaldi, A. Mahmood, and M. Gidlund, "Real-Time Interference Identification via Supervised Learning: Embedding Coexistence Awareness in IoT Devices," *IEEE Access*, vol. 7, pp. 835–850, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2885893.

[25] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," *IEEE Journal on Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018, ISSN: 19324553. DOI: 10.1109/JSTSP.2018.2797022.

[26] *Zigbee specification*, 2011. [Online]. Available: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/kjb79%7B%5C_%7Dajm232/pmeter/ZigBee%20Specification.pdf.

[27] *SigFox November 2019 specification*, 2019. [Online]. Available: https://storage.sbg1.cloud.ovh.net/v1/AUTH%7B%5C_%7D669d7dfced0b44518cb186841d7cb
prod%7B%5C_%7Dmedias/build/40599z1k361d4ht/Sigfox%20radio%20specifications%20v1.4%20November%202019.pdf.

[28] *LoRaWAN specification March 2018*, 2018. [Online]. Available: https://lora-alliance.org/sites/default/files/2018-04/lorawantm%7B%5C_%7Dspecification%7B%5C_%7D-v1.1.pdf.

[29] *Awesome Go*, 2019. [Online]. Available: https://awesome-go.com/%7B%5C#%7Dmachine-learning (visited on 11/20/2019).

[30] *Awesome Rust*, 2019. [Online]. Available: https://github.com/rust-unofficial/awesome-rust%7B%5C#%7Dmachine-learning (visited on 11/20/2019).

[31] (). NVidia Jetson Nano product page. Bezocht op 2019-08-15, [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/.

[32] (). Dell XPS 13 product page. Bezocht op 2019-08-15, [Online]. Available: https://www.dell.com/nl-be/shop/dell-laptops/xps-13/spd/xps-13-9380-laptop/CNX38004.

[33] *SKlearn: cross-validation behavior*, 2019. [Online]. Available: https://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/model%7B%5C_%7Dselection/plot%7B%5C_%7Dcv%7B%5C_%7Dindices.html%7B%5C#%7Dsphx-glr-auto-examples-model-selection-plot-cv-indices-py.