AI voor beginners (2)

neurale netwerken met Linux en Python

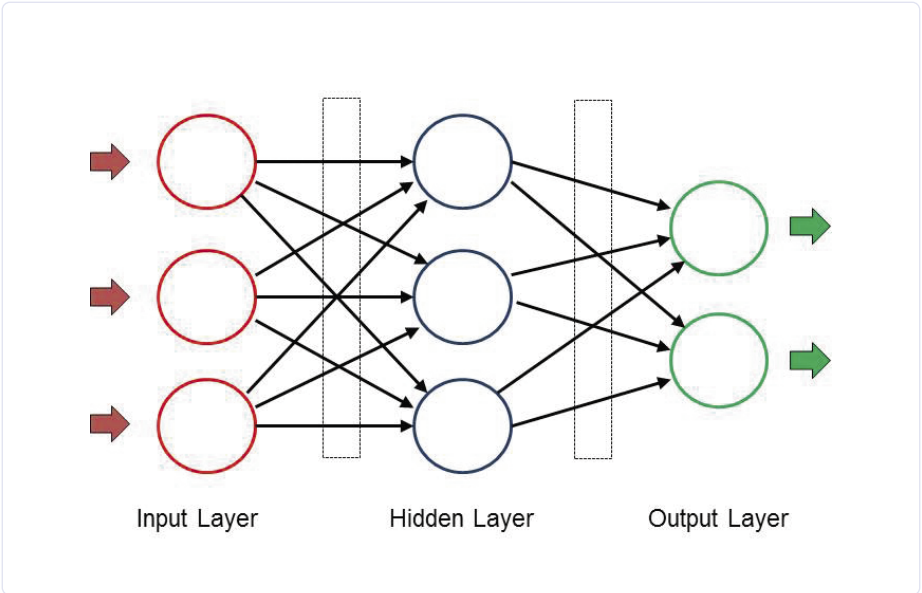
Walter Trojan (Duitsland)

In het eerste artikel van deze miniserie [1] zijn we ingegaan op de hardware van de Maixduino en hoe we die met de Arduino IDE in C++ kunnen programmeren. We hebben de prestaties van de processor gedemonstreerd met een AI-model voor objectherkenning. In dit artikel bespreken we hoe kunstmatige intelligentie werkt bij Deep Learning. Als u serieus met AI aan de slag wilt, zijn Linux en Python essentieel. Maar met het juiste gereedschap is dat geen zwarte magie.

Verwacht niet dat u na het lezen van dit artikel alles weet over Deep Learning. Ook mijn kennis over dit onderwerp is niet volledig. Maar ik wil u in ieder geval een overzicht geven van de structuren en methoden die worden gebruikt om een universeel toepasbaar programma in de vorm van een neuraal netwerk (NN) uit te rusten met gespecialiseerde intelligentie.

Opbouw van een neuraal netwerk

Een neuraal netwerk (NN) bestaat uit meerdere lagen, die elk meerdere knooppunten hebben. In **figuur 1** zijn de knooppunten van elke laag verticaal gerangschikt. In theorie kan er een willekeurig aantal knooppunten per laag en een willekeurig aantal lagen in het netwerk zijn. De architectuur wordt bepaald door de uit te voeren taak; de omvang is natuurlijk ook afhankelijk van de capaciteit van het gebruikte computerplatform. Als het NN bijvoorbeeld objecten moet classificeren die het als beelden krijgt aangeboden, dan komen de pixels aan in de eerste laag. Die heeft dan evenveel knooppunten als er pixels in de afbeelding zijn. Als een afbeelding met een lage resolutie uit slechts 28x28 pixels bestaat, dan zijn 784 knooppunten nodig voor een weergave in grijswaarden. Voor het verwerken van kleurenbeelden zijn al drie keer zoveel knooppunten nodig. Bij het registreren van het beeld krijgt elk invoerknooppunt de grijswaarde van 'zijn' pixel.

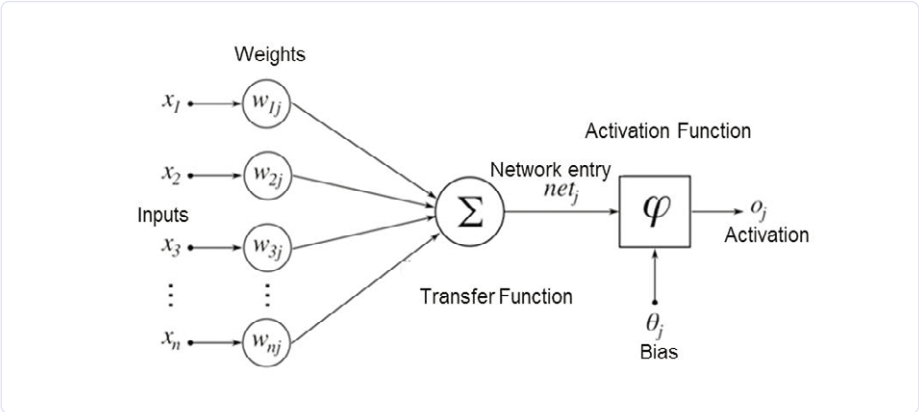


Figuur 1. Structuur van een neuraal netwerk.

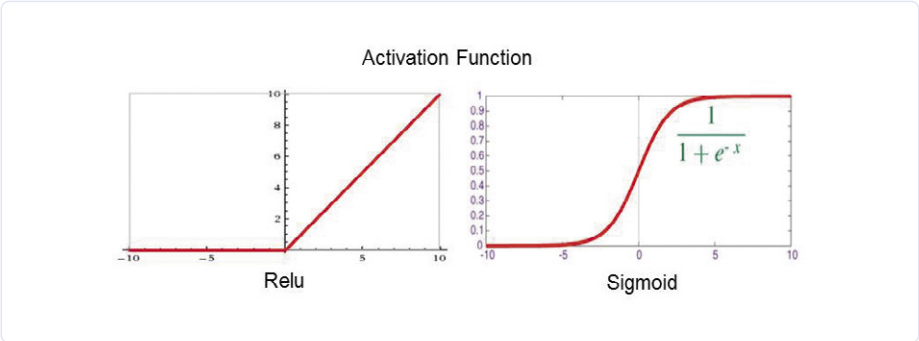
Het resultaat van de beeldanalyse wordt gepresenteerd in de uitvoerlaag, die voor elk resultaat een apart knooppunt heeft. Als het NN 1000 objecten moet herkennen, moet deze laag dus 1000 uitvoerknooppunten hebben. Elk knooppunt bevat een waarschijnlijkheid tussen 0 en 1,0 en geeft dus aan hoe zeker het netwerk is over het gevonden resultaat. Als bijvoorbeeld een huiskat wordt gevonden, kan het knooppunt 'Kat' een waarde van 0,85 hebben en het 'Tijger'-knooppunt een waarde van 0,1. Alle andere knooppunten zouden dan een nog lagere waarschijnlijkheid hebben en het resultaat zou duidelijk zijn.

Bij Deep Learning wordt het eigenlijke analysewerk gedaan door de verborgen lagen.

Afhankelijk van de taak kan een willekeurig aantal verborgen lagen worden gebruikt; in de praktijk zijn er zeker 100 tot 200 van dergelijke lagen in gebruik. Naast de hierboven getoonde architectuur zijn er complexere structuren te vinden met feedback of tussenliggende filters om de nauwkeurigheid te verhogen. Het aantal knooppunten in elke verborgen laag is ook willekeurig. In het bovenstaande voorbeeld voor objectdetectie kunnen drie verborgen lagen met elk 200 knooppunten al nuttige resultaten opleveren. Zoals in het menselijk brein intelligentie wordt bereikt door het koppelen van neuronen via synapsen, zo zijn ook de knooppunten van het NN met elkaar verbonden. Elk knooppunt van een laag is verbonden met alle knooppunten van de volgende lagen. Elke verbinding (weergegeven als een pijl in de figuur) bevat een waarde die gewicht wordt genoemd. Deze gewichten tussen de lagen worden opgeslagen in matrices. Daarom worden voor Deep Learning bij voorkeur programmeertalen gebruikt, waarmee matrixbewerkingen eenvoudig en snel kunnen worden uitgevoerd. En hoe bereikt een NN het gewenste resultaat? Elk knooppunt ontvangt zijn ingangssignalen van alle knooppunten van de laag ervoor, aangeduid met een *x* in **figuur 2.** Deze waarden worden vermenigvuldigd met de gewichtsfactoren *w* en bij elkaar opgeteld, zodat de netwerk-invoer gelijk is aan:



Figuur 2. Berekeningen in een knooppunt (https://commons.wikimedia.org/wiki/Artificial_neural_network).



Figuur 3. Veelgebruikte activeringsfuncties.

net = x1*w1 + x2*w2 + X3*w3 ...

De berekende waarde van net wordt vermenigvuldigd met een activeringsfunctie en naar de uitgang (en dus naar alle knooppunten van de volgende laag) gestuurd. Een knooppuntspecifieke drempelwaarde bepaalt of het knooppunt 'vuurt.'

De in **figuur 3** weergegeven activeringsfuncties worden aan de lagen toegewezen en zorgen ervoor dat de uitgangswaarden binnen het gewenste bereik blijven. Zo onderdrukt de functie Relu alle negatieve waarden en begrenst Sigmoid de waarde tussen 0 en 1. Door deze maatregelen werken NN's in een overzichtelijk numeriek bereik en kunnen ze niet worden gedomineerd door 'uitschieters.'

Training

Nadat de ingangsdata zijn ingelezen, voert het NN in alle lagen zijn berekeningen uit en presenteert de resultaten aan de uitgang. Dit proces wordt inferentie genoemd. En hoe komt daar de intelligentie in? Het netwerk moet, net als een kind, worden opgeleid en getraind. Bij een ongetraind NN worden de gewichten meestal gevuld met willekeurige getallen in een bereik van –1 tot +1, zodat het netwerk in het begin dom is en slecht willekeurige resultaten levert. Bij het trainen wordt het netwerk gevoed met gegevens en bekende gewenste uitgangswaarden. Als de gegeven zijn verwerkt, wordt het resultaat vergeleken met de gewenste uitkomst en wordt het verschil bepaald met behulp van een verliesfunctie. Dan volgt het leerproces, ook wel *backpropagation* genoemd. De gewichtsfactoren worden in kleine stappen aangepast om het verlies te minimaliseren, waarbij van de uitgang naar de ingang wordt gewerkt. Na vele (vaak miljoenen) trainingssessies met verschillende invoergegevens, hebben alle gewichtsfactoren een waarde die geschikt is voor de taak en kan het NN nieuwe, onbekende gegevens analyseren die het nog nooit eerder heeft gezien.

Een architectuur die bijzonder geschikt is voor beeld- en geluidsverwerking is het *Convolutional Neural Network*, dat ook door Maixduino wordt ondersteund. In zo'n CNN zijn de neuronen (in ieder geval in sommige lagen) tweedimensionaal gerangschikt, wat past bij tweedimensionale invoergegevens zoals een afbeelding. In vergelijking met het netwerk van figuur 1, waar de activiteit van een neuron afhankelijk is van alle neuronen van de vorige laag (via verschillende gewichtsfactoren), is de afhankelijkheid bij een CNN vereenvoudigd en lokaal begrensd. Hier is de activiteit van een neuron alleen afhankelijk van de waarden van (bijvoorbeeld) 3x3 neuronen die zich in de laag ervoor bevinden. De gewichtsfactoren zijn daarbij gelijk. Met zo'n netwerk kunnen kleinschalige structuren zoals lijnen, bogen, punten en andere patronen bijzonder goed worden herkend. In de volgende lagen worden meer complexe details en uiteindelijk hele gezichten gedetecteerd.

Wat betreft de programmering is een NN vergelijkbaar met een spreadsheet: een reeks cellen met voorgedefinieerde rekeninstructies, die bij de uitvoering gebruik maken van multidimensionale matrices met de gewichtsfactoren. Bij het classificeren van nieuwe invoergegevens worden alle lagen van de invoer naar de uitvoer doorgerekend; op de uitgangen verschijnt dan de waarschijnlijkheid van het resultaat dat aan het knooppunt

is toegewezen. Goed getrainde NN's kunnen waarden bereiken van ongeveer 0,9. Tijdens de training vindt na de inferentie backpropagation plaats van de uitgang naar de ingang om de gewichtsfactoren aan te passen.
Dit klinkt aanvankelijk ingewikkeld, maar er zijn tal van tools en bibliotheken beschikbaar voor de implementatie. Ik zal die het volgende deel van deze serie introduceren.

Linux en Python

In het eerste artikel van deze miniserie heb ik al vermeld dat er heel wat te leren valt als u zich met AI gaat bezigheouden. Een Linux-platform is de gemakkelijkste manier om met AI aan de slag te gaan, omdat daarvoor de meeste gratis tools van goede kwaliteit beschikbaar zijn. Bovendien biedt Linux hetzelfde comfort als Windows, maar is het anders verpakt (meestal beter). Mijn voorkeur gaat uit naar Ubuntu, dat ook beschikbaar is als LTS-versie (Long Term Support) en minimaal 4 jaar wordt ondersteund. Maar ook de andere distributies, zoals Debian, Mint enzovoort zijn heel geschikt, de keuze is een kwestie van smaak. Linux kan in een virtuele machine naast Windows worden geïnstalleerd, u hebt dus geen extra computer nodig.
En waarom Python? Het is een interpretatieve programmeertaal, dus sommigen zouden zeggen dat het potentieel traag is. Maar dat nadeel wordt gecompenseerd door talrijke voordelen: ten eerste werkt Python zonder de franje van haakjes en puntkomma's. De definitie van blokken wordt gewoon weergegeven door regels te laten inspringen. Er zijn krachtige datastructuren zoals lijsten, tupels, sets en dictionaries; bovendien heeft Python volledig geïntegreerde matrixberekeningen. Andere grote voordelen zijn de beschikbare AI-frameworks en bibliotheken, die door hun modulariteit geheel of gedeeltelijk kunnen worden geïntegreerd. Meestal zijn die geschreven in C++ en hebben dus niet het nadeel dat ze geïnterpreteerd moeten worden. Dat leidt tot snelle berekeningen. En ze kunnen eenvoudig, met slechts een paar instructies, worden geïnstalleerd.
Als u aan de slag wilt, installeer dan uw favoriete Linux-distributie en Python 3. Op het internet vindt u hoe dat in zijn werk gaat.

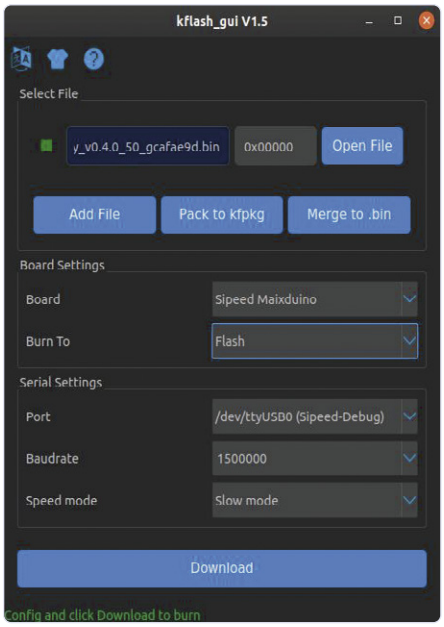
Maixduino spreekt MicroPython

Om Python te laten draaien op systemen met minder geheugen is er een lichtgewicht versie beschikbaar die MicroPython heet; deze kan worden geïnstalleerd op platforms zoals Maixduino, ESP32 en andere. MicroPython bevat een uitgebalanceerd repertoire aan commando's en 55 extra modules voor tal van wiskundige- en systeemfuncties. Om nieuwe versies of AI-modellen toe te voegen, is het flash-tool Kflash nodig.

Installeren van Kflash onder Linux

- Download versie 1.5.3 of hoger van [2] als ingepakt bestand *kflash_gui_v1.5.3_linux.tar.xz.*
- Sla het op in een map naar keuze.
- Pak uit met het commando `tar xvf kflash_gui_v1.5.3_linux.tar.xz`.
- Ga naar de nieuw aangemaakte map */kflash_gui_v1.5.2_linux/kflash_gui.*
- Start op met *./kflash_gui* (als dat niet lukt, vink dan het vakje *Bestand uitvoeren als programma* aan in de eigenschappen van dit bestand).

Hiermee wordt de grafische gebruikersinterface van Kflash gestart (**figuur 4**). Nu kunnen we firmware- of AI-modellen in de Maixduino laden.



Figuur 4. Gebruikersinterface van Kflash.

Firmware installeren op Maixduino

Ook al is de Maixduino bij levering al uitgerust met MicroPython, moet u in ieder geval de laatste versie te downloaden. Op het moment van schrijven is dat versie v0.5.0 van de firmware. Die is te downloaden via de weblink [3]. Kies *maixpy_v0.5.0_8_g9c3b97f* of hoger en kies daarna de variant *maixpy_v0.5.0_8_g9c3b97f_minimum_with_ide_support.bin* of hoger. Het gaat om een bestand van ongeveer 700 kB dat ook ondersteuning voor de MaixPy-IDE bevat.
Met behulp van Kflash is de nieuwe firmware gemakkelijk te installeren. U selecteert die onder de button *Open File* en selecteert dan *Board*, *Port*, *Baudrate* en *Speed mode* zoals te zien in figuur 4. Klik dan op *Download* om het downloadproces te starten. Daarna kunt u de eerste Python-commando's al met een terminalemulator (bijvoorbeeld Putty) uitvoeren via de poort */dev/ttyUSB0* op de Maixduino. Hier is een klein voorbeeld met array-commando's:

```
>>>                                  # Python-Prompt
>>> import array as arr              # Import Array-Modul
>>> a = arr.array('i',[1,2,3])       # Anlegen Array a mit Integern
>>> b = arr.array('i',[1,1,1])       # Anlegen Array b mit Integern
>>> c = sum(a + b)                   # Summe aus allen Array-Werten
bilden
>>> print(a,b,c)                     # und ausgeben
array('i', [1, 2, 3]) array('i', [1, 1, 1]) 9     # Ausgabe
>>>
```

Met bibliotheken als *numpy* (onder Linux) en/of *umatlib* zijn nog meer functies beschikbaar.

Installeren van de MaixPy-IDE

Het ontwikkelen en testen gaat veel comfortabeler met de ontwikkelomgeving MaixPy-IDE. Python-programma's kunnen daarmee ontwikkeld, getest, geladen en uitgevoerd worden op de Maixduino. Daarnaast zijn er hulpmiddelen voor beeldanalyse beschikbaar, zoals te zien is in **figuur 5**. De installatie gaat als volgt:

- Download de versie *maixpy-ide-linux-x86_64-0.2.4-installer-archive.7z* of hoger van weblink [4].
- Zet die in een map naar keuze.
- Pak het bestand uit met het commando *tar maixpy-ide-linux-x86_64-0.2.4-installer-archive.7z.*
- Ga naar de nieuwe map *maixpy-ide-li-*

nux-x86_64-0.2.4-installer-archive en voer de volgende commando's in:

```
./setup.sh
./bin/maixpyide.sh
```

Daarna start de IDE. Om hem later opnieuw te starten, is natuurlijk alleen het laatste commando nodig.

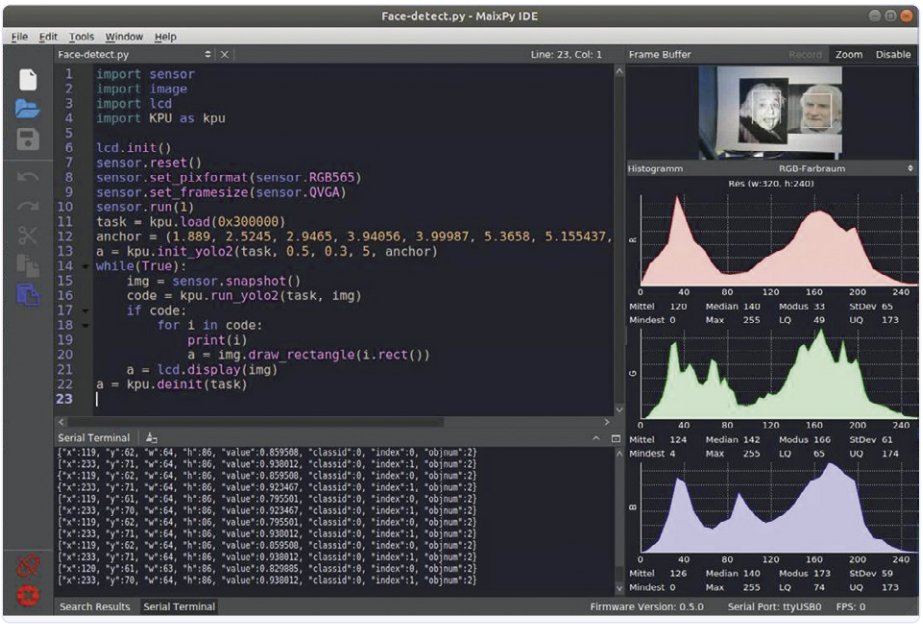
Nu zijn alle instrumenten voor de implementatie van AI-modellen beschikbaar. We gaan nu experimenteren met gezichtsherkenning.

Gezicht herkend!

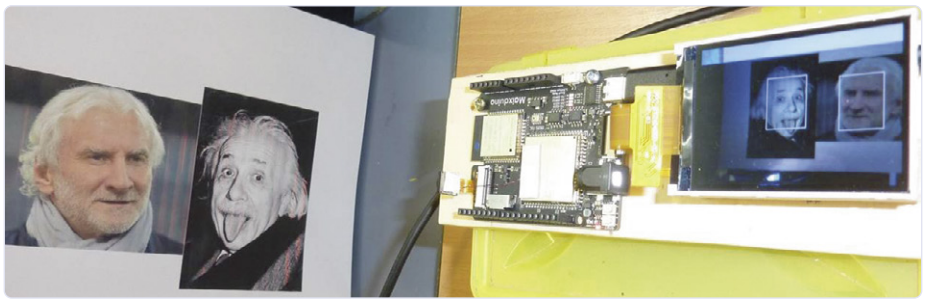
Voor de gezichtsherkenning gebruiken we een vooraf getraind AI-model, dat al enkele duizenden gezichten heeft geanalyseerd op basis van karakteristieke kenmerken. De gewichtsfactoren van het gebruikte NN worden daarmee ingesteld. Het model kan worden gedownload van de weblink [5] onder de naam *face_model_at_0x300000.kfpkg.* De basis van de ontwikkeling is het AI-framework Yolo2 (You Only Look Once), dat de beeldobjecten in verschillende zones verdeelt, ze afzonderlijk analyseert en zo hoge herkenningspercentages bereikt (ik kom later in deze serie nog terug op de AI-frameworks). Het AI-model voor de AI-processor is verpakt in *kfpkg-formaat* en moet worden geflasht op adres 0x300000 om het te kunnen gebruiken. Dit kan ook met Kflash: zoek het bestand op met *Open File* en laad het in het board met de parameters die in figuur 4 zijn weergegeven. De MaixPy-IDE wordt gebruikt om comfortabel met Python-scripts te werken. Met deze IDE kunt u programma's ontwikkelen en testen en deze overbrengen naar Maixduino. **Figuur 5** toont de, over drie vensters verdeelde, interface:

- **Editor**, linksboven: dit gebied wordt gebruikt voor programma-invoer met syntax highlighting.
- **Terminal**, linksonder: weergave van de programma-uitvoer.
- **Beeldanalyse**, rechts: hier worden de beelden en hun spectrale verdeling in de kleuren rood, groen en blauw weergegeven.

Naast andere beschikbare knoppen en menu-items zijn de twee knoppen linksonder belangrijk: de 'paperclip' wordt gebruikt om de Maixduino aan te sluiten (groen) of af te koppelen (rood) via de poort 'ttyUSB0.' Een groene driehoek eronder start het script; daarna verandert deze knop in een rode stip met een 'x' die dient om het programma te stoppen.



Figuur 5. Gebruikersinterface van de MaixPy-IDE.



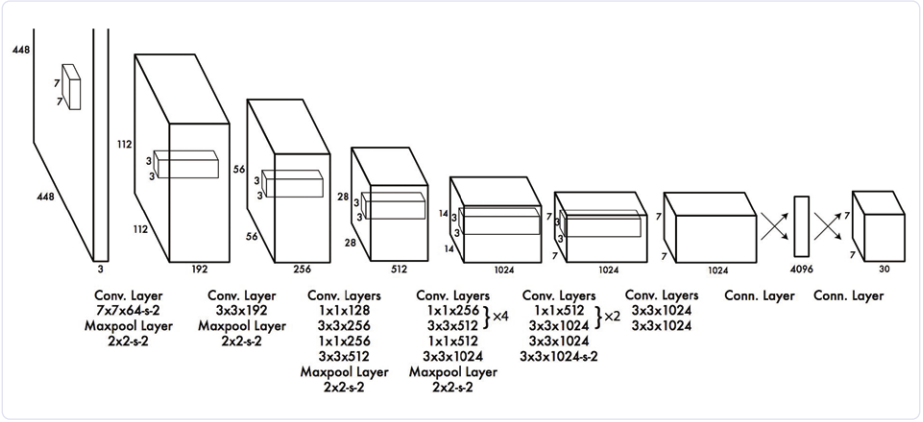
Figuur 6. Testopstelling voor gezichtsherkenning.

Voor de test heb ik de gezichten van twee bekende personen (Albert Einstein en Rudi Völler) geprint en op een wand geprikt. Men zegt dat die beide gezichten een zekere gelijkenis vertonen, maar zelf kon ik dat niet zien. Bij het registreren van deze beelden werden de gezichten onmiddellijk herkend en met een kader gemarkeerd. Let erop dat de afbeeldingen in landscape-formaat (liggend) moeten zijn, anders neemt het herkenningspercentage merkbaar af.

Het programma *Face-detect.py is te* vinden in de downloadmap op de website van Elektor [6]. Het is een heel kort programma, waaruit opnieuw blijkt hoe krachtig de gebruikte bibliotheken zijn. Om te beginnen worden de benodigde bibliotheken voor camera, LCD en KPU geïntegreerd en geïnitialiseerd. Daarna wordt het NN in de KPU geladen vanaf adres 0x300000. Wanneer het AI-model wordt geïnitialiseerd met het commando `kpu.init_yolo2`, worden extra constanten overgedragen voor het instellen van de nauwkeurigheid en optimalisatie. De beeldherkenning wordt uitgevoerd in een oneindige `while`-lus: telkens wordt een beeld geregistreerd en aangeboden aan het NN. Als er gezichten zijn gedetecteerd, krijgt de variabele `i` voor elk gezicht de coördinaten en de afmetingen van een markeringskader, dat vervolgens in het beeld wordt getekend. Tot slot worden het beeld (op het LCD) en de markeringsgegevens (op de seriële terminal) uitgevoerd. Meer details over de KPU-commando's zijn te vinden onder [7].

In de MaixPy-IDE wordt de afbeelding ook in de rechterbovenhoek getoond en het bijbehorende kleurenspectrum wordt eronder weergegeven. Als u deze informatie niet nodig heeft, kan het betreffende beeldven-

Figuur 7. Weergave op het Maixduino LCD-scherm.



Figuur 8. Netwerkarchitectuur voor gezichtsherkenning (bron: https://bit.ly/3cK3DUR).

ster worden uitgeschakeld met de knop *Deactivate.*
Ik heb voor het gemak de Maixduino en het LCD op een plankje gemonteerd met de camera naar voren gericht (zie **figuur 6**). Dat maakt het gemakkelijk om echte gezichten, afgedrukte beelden of scherminhoud te registreren en te analyseren.

De weergave op het LCD is te zien in **figuur 7**, de herkenning van de personen is daar niet zichtbaar.
Maar wat zit er achter dit Yolo2-model? Het neurale netwerk heeft 24 convolutionele lagen en twee volledig gekoppelde uitgangslagen (zie **figuur 8**). Er zijn enkele maxpool-lagen tussengevoegd als filters om de complexiteit te verminderen en de neiging tot 'van buiten leren' te verminderen. Het valt op dat er veel gebruik wordt gemaakt van een venstergrootte van 3x3 voor het herkennen van details. Precies dat wordt ondersteund door de KPU-hardware, die ervoor zorgt dat de Maixduino zeer efficiënt is voor dergelijke taken.
Andere bekende NN-structuren hebben soms wel honderden lagen en hebben regressies of andere extra's. Er zijn geen grenzen aan de creativiteit op dit gebied, veel hangt af van het budget, dus. van de rekenkracht.

We houden hier niet op!

De krachtige hardware en nu al beschikbare software-omgeving tonen aan dat de Maixduino zeer geschikt is voor de kennismaking met de wereld van de kunstmatige intelligentie. Door het lage stroomverbruik is hij zeer geschikt voor gebruik in mobiele apparaten met reeds getrainde neurale netwerken. In het derde deel van deze serie laat ik zien hoe u een eigen neuraal netwerk kunt ontwikkelen, trainen en uitvoeren. Daarbij gebruiken we het AI-framework Keras, dat bekend staat als een comfortabele 'Lego-bouwdoos voor AI.' U krijgt ook tips over hoe u de ESP32 op het board kunt programmeren, bijvoorbeeld om analoge waarden te verkrijgen.
Blijf nieuwsgierig!

200023-B-03

WEBLINKS

[1] **AI voor beginners (1), Elektor mei/Juni 2020**: www.elektormagazine.nl/200023-04
[2] **Kflash**: https://github.com/sipeed/kflash_gui/releases
[3] **Maixduino-firmware**: http://dl.sipeed.com/MAIX/MaixPy/release/master/
[4] **MaixPy-IDE**: http://dl.sipeed.com/MAIX/MaixPy/ide/_/v0.2.4/maixpy-ide-linux-x86_64-0.2.4-installer-archive.7z
[5] **KI-modellen**: http://dl.sipeed.com/MAIX/MaixPy/model
[6] **Software**: www.elektormagazine.de/200023-B-01
[7] **KPU-commando's**: https://maixpy.sipeed.com/en/libs/Maix/kpu.html