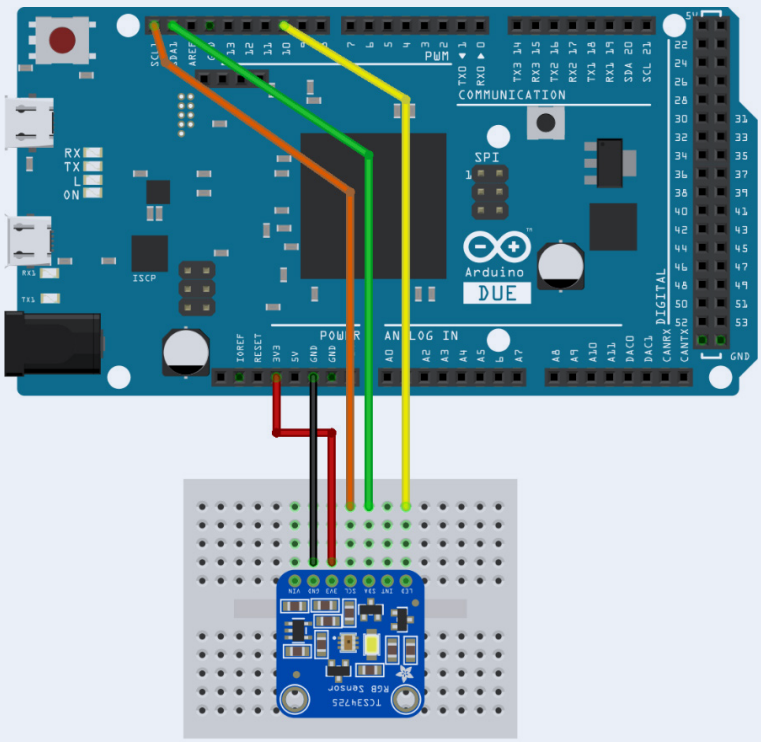
Kennismaking met neuronen in neurale netwerken

deel 4: embedded neuronen

Stuart Cording (Elektor)

Nu ons neurale netwerk op PC's en laptops volledig functioneert , kunnen we vertrouwen op onze meerlagige perceptron-implementatie. Veel toepassingen vereisen het lage stroomverbruik en de minimale latentie die een microcontroller biedt. Anderen willen hun privégegevens eenvoudigweg niet delen met cloud AI-diensten van derden. Hier maken we ons neuraal netwerk 'Arduino-ready' en verplaatsen we onze verkeerslicht-classificatiecode naar de wereld van embedded systemen.



Figuur 1. Aansluiting van de TCS34725 RGB-sensor op een Arduino Due.

Tools en platforms voor machine learning (ML) lijken als paddenstoelen uit de grond te schieten. Hoe complex uw taak ook is of hoe groot uw dataset ook is, er is een cloud-service die het aan kan. Er zijn echter veel gevallen waarin een cloud-gebaseerde ML-implementatie ongeschikt is. Als u met gevoelige of persoonlijke gegevens werkt, wilt u deze misschien niet via het internet verzenden om ze door een ML-tool te laten analyseren. Een ander voorbeeld zijn automotive of andere real-time embedded toepassingen. Dergelijke systemen vereisen een onmiddellijke beslissing, dus gezien de latentie van een internetverbinding (of de mogelijkheid dat er helemaal geen verbinding is) is een lokale ML-oplossing onoverkomelijk. Dit staat ook bekend als 'ML on the edge' [1].

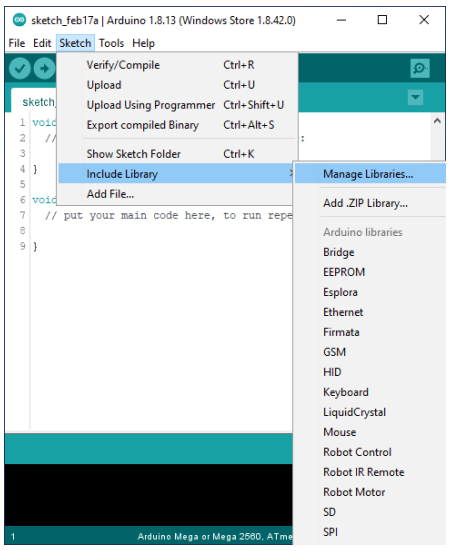
Het werken met een neuraal netwerk on the edge betekent dat zelfs eenvoudige microcontrollers ML-algoritmen kunnen uitvoeren. Vanwege de eisen die training aan een processor stelt, wordt het netwerk echter vaak in de cloud of op een krachtige PC getraind. De resulterende gewichtsfactoren worden dan naar de microcontroller gedownload voor internetvrij, low latency-gebruik.

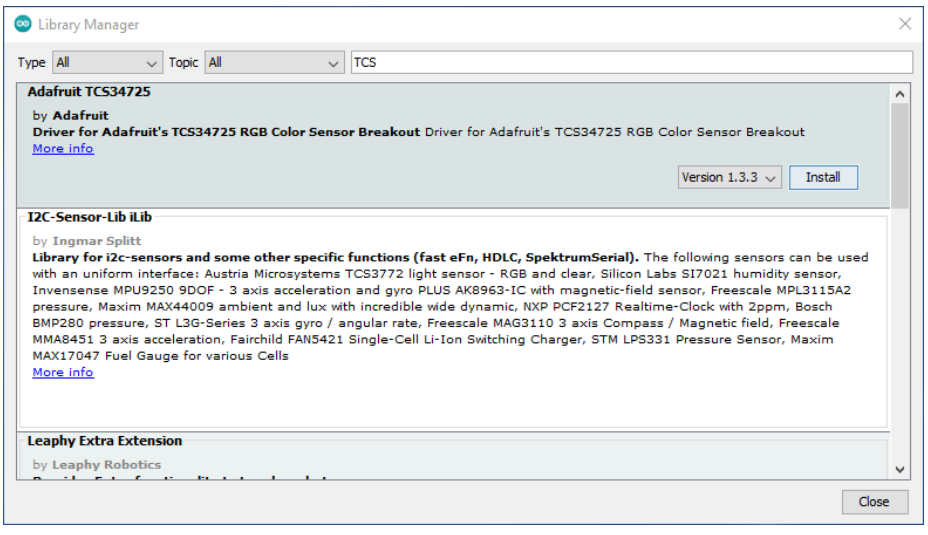
Dit laatste artikel van deze serie zet ons multilayer-perceptron (MLP) neurale netwerk over op een Arduino. Gekoppeld aan een RGB-sensor, leren we het om de kleuren van verkeerslichten te classificeren zoals we deden in Processing op uw PC of laptop. Met behulp van een 32-bit Arduino (DUE of Mo Pro) kunnen we de leerfase op de microcontroller uitvoeren. We zullen echter zien dat dit nogal traag is. Daarom zullen we de taak zo opsplitsen dat het netwerk wordt getraind met een PC, en vervolgens wordt uitgevoerd in de low-latency, low-power omgeving van een embedded microcontroller-applicatie.

Keuze van een sensor

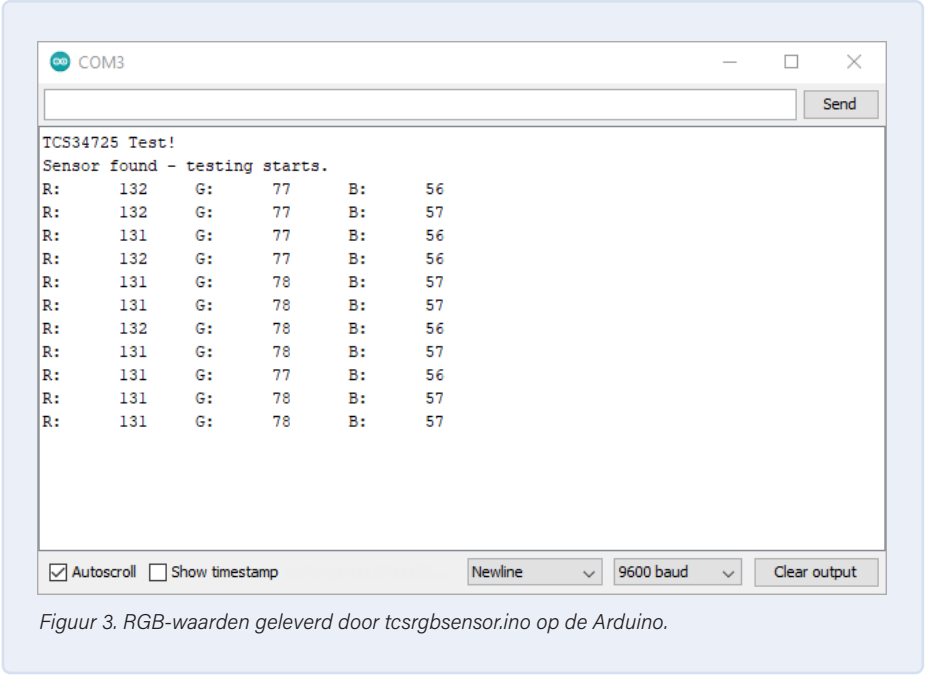
Om de kleuren van de verkeerslichten te herkennen, heeft de Arduino een geschikte sensor nodig. De TCS34725 is een goede RGB-sensor, kant-en-klaar verkrijgbaar bij Adafruit in de vorm van een printje van 2 × 2 cm (0,8" × 0,8") [2]. Hiermee kunnen we net zoals we eerder deden met de camera RGB-data verzamelen en deze loslaten op dezelfde MLP-configuratie die gebruikt werd in de voorgaande PC-gebaseerde voorbeelden. De print is geschikt voor gebruik met zowel 5V- als 3,3V-Arduino's; de sensorgegevens worden ingelezen via I²C, waarvoor de Wire-library nodig is.

Om een consistente verlichting van het te analyseren monster te garanderen, bevat het board ook een witte LED die wordt





Figuur 2. Installatie van de Adafruit TCS34725 RGB-sensor softwarebibliotheek in de Arduino IDE.



Figuur 3. RGB-waarden geleverd door tcsrgbsensor.ino op de Arduino.

aangestuurd met een digitale uitgang van Arduino – we hebben pin 10 gekozen (**figuur 1**). Gelukkig wordt het board ondersteund door een goed samengestelde bibliotheek, zodat de ingebruikname eenvoudig en ongecompliceerd is. Net als voorheen zullen we de code gebruiken uit de GitHub-repository die voor deze artikelenreeks is voorbereid [3].

Voordat we enige code uitvoeren, moeten we de bibliotheek voor de RGB-sensor installeren. Dat gaat gelukkig ook gemakkelijk, omdat die toegankelijk is via de Library Manager binnen de Arduino IDE. Selecteer in de menubalk *Sketch -> Include Library -> Manage Libraries...* en voer dan 'tcs' in het zoekveld van de Library Manager in. De 'Adafruit TCS34725'-library moet nu verschijnen. Klik op 'Install' om deze te installeren (**figuur 2**). Mochten er moeilijkheden zijn, dan kunnen de broncode en het headerbestand worden gedownload van de Adafruit GitHub-repository [4] en handmatig aan de projecten worden toegevoegd.

Om er zeker van te zijn dat de sensor werkt en om de RGB-waarden te verzamelen we nodig hebben om de ML te trainen, beginnen we met de sketch *arduino/tcsrgbsensor/tcsrgbsensor.ino*. Na het initialiseren van de RGB-sensor schakelt de code de LED in en begint de RGB-waarden via de seriële uitgang te rapporteren (**figuur 3**). Open *Tools -> Serial Monitor* om ze te bekijken. De ingestelde baudrate is 9600. Om de kwaliteit van de metingen te verbeteren en de invloed van andere lichtbronnen te verminderen, is het de moeite waard een afscherming rond het sensorboard te maken met een strook zwart karton van ongeveer 3 cm hoog en 10 cm lang (**figuur 4**). De afscherming zorgt er ook voor dat de afbeelding van de verkeerslichten op een constante afstand van de RGB-sensor kan worden gehouden.

Nu de RGB-sensor betrouwbare resultaten oplevert, kunnen we zijn beoordeling van de rode, gele en groene kleur verkrijgen met behulp van onze printout van het verkeerslicht (van `trafficlight/resources`). De door de auteur verkregen resultaten zijn weergegeven in **tabel 1**.

Tabel 1. RGB-waarden vastgelegd met de TCS34725 voor de drie kleuren van het verkeerslicht.

Kleur	R	G	B
Rood	149	56	61
Geel	123	77	61
Groen	67	100	90

Een MLP-library voor de Arduino

Met de nu bepaalde RGB-waarden hebben we de neurale netwerkimplementatie nodig voor het Arduino-platform. Omdat Processing Java gebruikt, kunnen we niet eenvoudigweg de code van de Neuralclass nemen en deze toevoegen aan een Arduino-project. Daarom is de Java Neural-klasse licht aangepast om er een C++ klasse van te maken. In de Arduino-wereld kunnen dergelijke herbruikbare code-objecten worden omgezet in 'libraries'. Deze bestaan uit een C++ klasse en een extra bestand voor het markeren van de sleutelwoorden van de class. Mocht u uw eigen library willen schrijven of beter willen begrijpen hoe C++ klassen werken bij de Arduino, dan is er een uitstekende tutorial op de Arduino-website [5].

De code voor onze Neural-library staat in *arduino/neural*. De volgende Arduino-projecten voegen eenvoudig de broncode van de Neural-klasse in de sketch in om alles simpel te houden. Het *neural.h* header-bestand moet ook worden toegevoegd aan de map waar het project is opgeslagen.

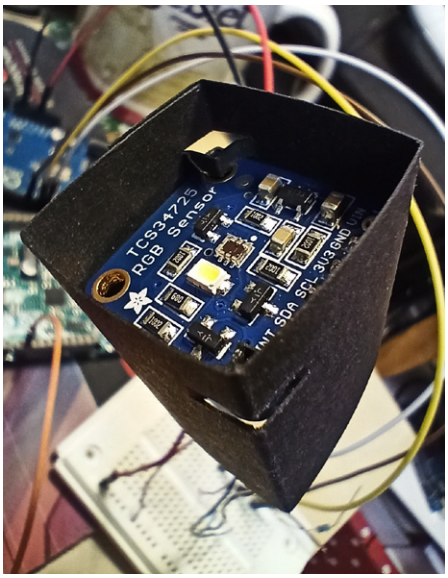
Het gebruik van de `Neural`-klasse op een Arduino is grotendeels hetzelfde als het gebruik ervan in Processing. Het creëren van ons `network`-object als een globale variabele is iets anders en wordt als volgt aangepakt:

```
Neural network;
```

Om het object met het gewenste aantal ingangs-, verborgen en uitgangs-knooppunten (input, hidden en output) te construeren, voeren we het volgende in:

```
network = new Neural(3,6,4);
```

De basisconfiguratie van de voorinstellingen en de leersnelheid worden vervolgens gecodeerd zoals eerder in Processing:



Figuur 4. TCS34725 RGB-sensor met afscherming van zwart karton.

```
network.setLearningRate(0.5);
network.setBiasInputToHidden(0.35);
network.setBiasHiddenToOutput(0.60);
```

Vanaf hier kunnen we de methodes blijven gebruiken zoals we dat eerder in Processing deden.

Verkeerslichtkleuren detecteren met Arduino

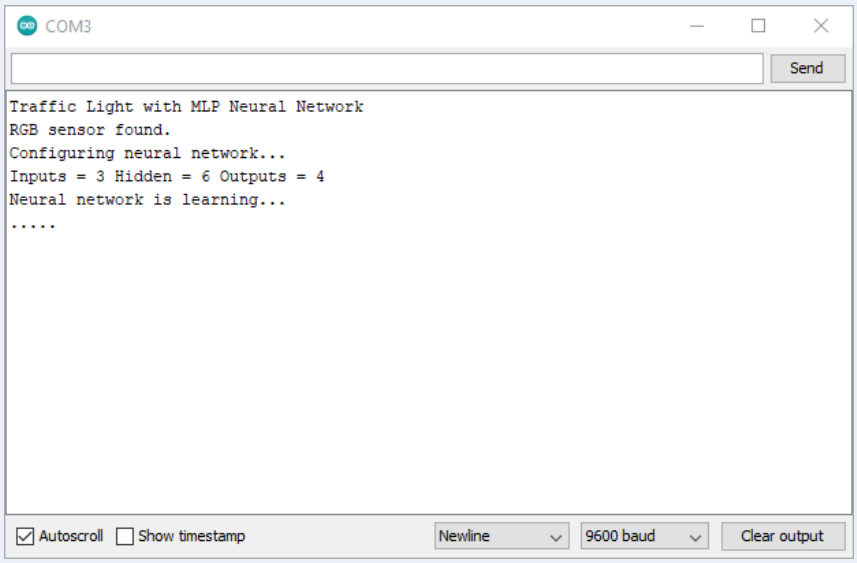
Nu kunnen we beginnen met het verkennen van de MLP op de Arduino. De sketch */arduino/tlight_detect/tlight_detect.ino* volgt dezelfde structuur als het Processingproject *tlight_detect.pde*. Het neurale netwerk (3/6/4) wordt geconfigureerd vanaf regel 40, en het wordt in een lus getraind met de RGB-gegevens vanaf regel 51. Alvorens de code uit te voeren, moeten de eerder verkregen RGB-waarden voor 'rood' (red), 'geel' (amber) en 'groen' (green) worden ingevoerd vanaf regel 56:

```
teachRed(220, 56, 8);
teachAmber(216, 130, 11);
teachGreen(123, 150, 128);
```

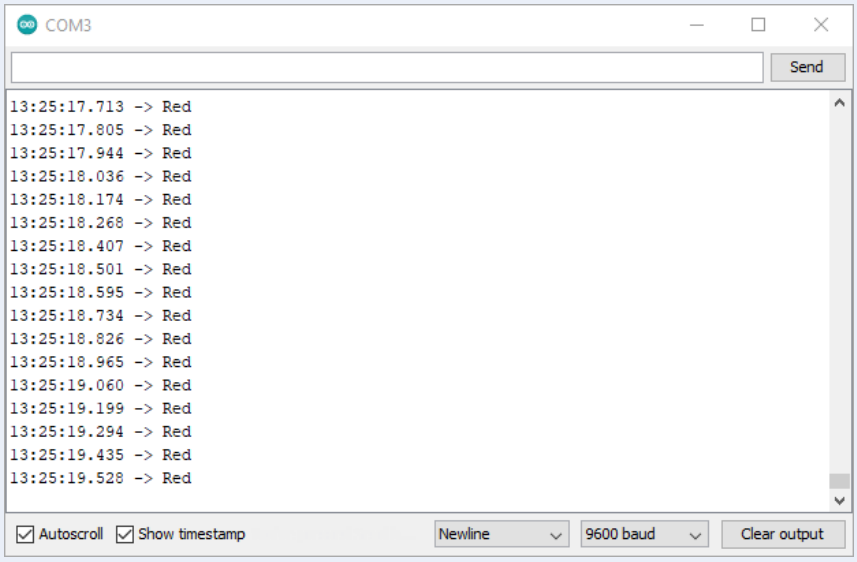
Laad de code en open de seriële monitor om de uitvoer te bekijken (**figuur 5**). Net als voorheen is de baudrate ingesteld op 9600. Het project controleert of de RGB-sensor functioneert alvorens de witte LED tijdens het leren uit te schakelen. Nadat het MLP is geconfigureerd, wordt de leercyclus 30.000 keer doorlopen, waarbij het vermogen om elk van de drie kleuren te classificeren bij elke doorloop wordt verbeterd.
Om tijdens het leren enige feedback te geven, wordt om de 1000 luscycli (3000 leercycli) een punt uitgevoerd. Vergeleken met de PC is het leren hier een langzaam proces. Elke aanroep van een leerfunctie (`learnRed()` enzovoort) neemt ongeveer 5,55 ms in beslag. Het leren van alle drie de kleuren duurt ongeveer 8,5 minuut op een Arduino M0 Pro met een SAMD21-microcontroller. Als u de verwerkingstijd van uw eigen platform wilt onderzoeken, vink dan het vakje *Show Timestamp* aan in de seriële monitor en vervang regel 66 door

```
Serial.println(".");
```

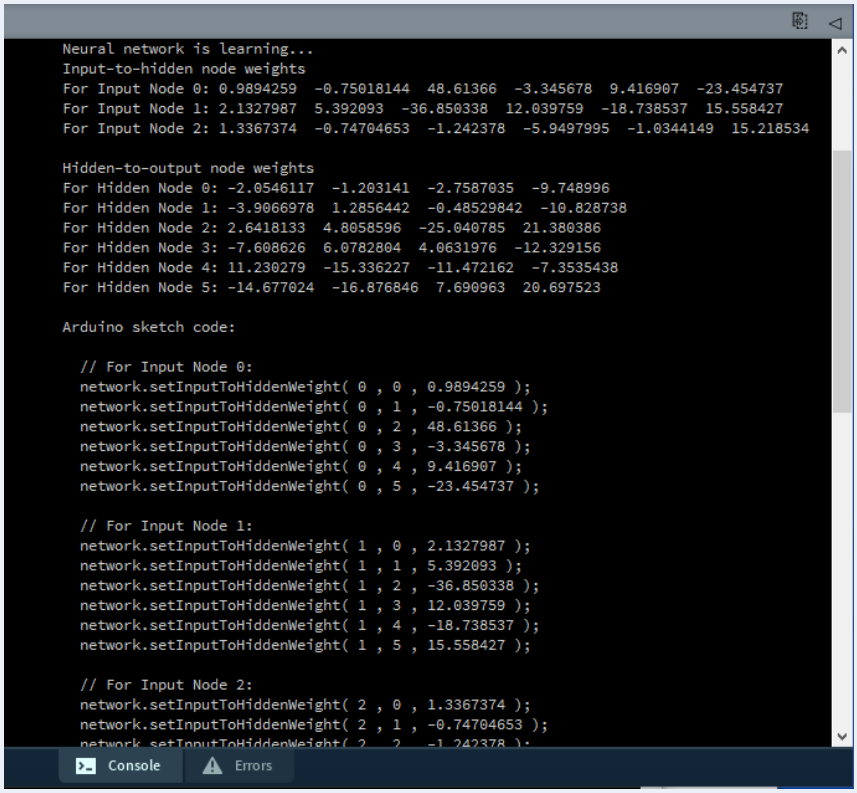
Hiermee wordt aan elke uitgevoerde punt een return-karakter toegevoegd en een timestamp na elke punt. Zodra het leren is voltooid, demonstreert de Arduino onmiddellijk zijn nieuwe kunstje. Telkens wanneer een verkeerslichtkleur wordt gedetecteerd, wordt dit uitgevoerd naar de seriële monitor (**figuur 6**). Tijdens de experimenten van de auteur detecteerde het neurale netwerk ook de kleur 'geel', zelfs wanneer niets voor de sensor werd gehouden. Hoewel dit verband leek te houden met de omgevingsverlichting, wijst het wel op een zwak punt in de implementatie. Om de code te verbeteren, kunnen we de MLP 'andere' kleuren leren, zoals we eerder in Processing hebben gedaan. Dit kan ook worden gebruikt om de classificatie 'geel' te onderdrukken wanneer er geen afbeelding voor de camera wordt gehouden. De *tcsrgbsensor.ino*-sketch kan worden gebruikt om de sensormetingen te verkrijgen voor de rand, frame en achtergrond van onze afbeelding van het verkeerslicht. Deze kunnen vervolgens worden ingevoerd in de regels 60 en verder in de *tlight_detect.ino*-sketch in de aanroepen van de `teachOther()`-functie.
De waarden in **tabel 2** werden verkregen en getest met de *tlight_detect.ino*-sketch. De classificatie verbeterde, maar de foutieve classificatie 'geel' zonder afbeelding werd niet volledig opgelost. Zoals altijd is er ruimte voor verbetering!



Figuur 5. Uitvoer van tlight_detect.ino tijdens het leerproces.



Figuur 6. Uitvoer van tlight_detect.ino na het leren en detecteren van kleuren.



Figuur 7. Gewichtsfactoren kunnen in Processing op een PC snel worden berekend met learnfast.pde. De uitvoer van de tekstconsole kan dan in tlight_weights.ino worden geplakt.

Tabel 2. RGB-waarden bepaald met de TCS34725 voor de 'ongewenste' kleuren.

Kleur	R	G	B
Donkere rand	92	90	82
Wit frame	92	90	75
Blauwe achtergrond	73	93	89

Turbo-leren

Als u het leren van de 'andere' kleuren toevoegt aan de Arduino-sketch, moet u bij een Mo-Pro ongeveer 18 minuten wachten op het leren de gewenste classificaties. Hier is duidelijk ruimte voor optimalisatie. Aangezien we een krachtige PC hebben die de gewichtsfactoren in minder dan een seconde kan berekenen, zou het zinvol zijn om het leren daar te doen en dan de resultaten naar de Arduino over te brengen. Met deze gewichtsfactoren hebben we ook een middel om meerdere microcontrollers met dezelfde 'kennis' te programmeren. Ervan uitgaande dat de RGB-sensoren allemaal hetzelfde functioneren ten opzichte van onze invoer, zou elke microcontroller de afbeelding van het verkeerslicht correct moeten classificeren. Dat is dus wat we nu gaan doen.

We gaan even terug naar Processing en openen het project */arduino/learnfast/learnfast.pde*. De hele toepassing draait in de `setup()`-functie. Het neurale netwerk is geconfigureerd met dezelfde invoer-, verborgen en uitvoer-nodes als op het Arduinoboard (3/6/4). In de leer-lus (regel 36) worden de waarden gebruikt die zijn verkregen van de Arduino met behulp van de RGB-sensor en *tcsrgbsensor.ino*. Wanneer de code wordt uitgevoerd, wordt er tekst naar de console gestuurd. De laatste sectie bevat de code die alle input-naar-hidden en hidden-naar-output gewichtsfactoren configureert (**figuur 7**). Kopieer eenvoudigweg de gegenereerde code vanaf `// For Input Node =0` tot het einde van de tekstuitvoer.

Terug in de Arduino IDE kunnen we nu de */arduino/tlight_weights/tlight_weights.ino*-sketch openen. Deze is gelijk aan de *tlight_detect.ino*-sketch, maar in plaats van het neurale netwerk te onderwijzen, worden de gewichtsfactoren voorgeprogrammeerd. Dit gebeurt op regel 51 met de functie `importWeights()`. Plak gewoon de code van de *learnfast.pde*-uitvoer in de `importWeights()`-functie op regel 86 in *tlight_weights.ino*. Programmeer het Arduino-board; het zou nu nauwkeurig de kleuren van de verkeerslichten moeten detecteren, zoals voorheen.

En nu we over dit turbo-leerproces beschikken, kunnen we zelfs dezelfde *tlight_weights.ino*-sketch in een Arduino UNO programmeren. Sluit de RGB-sensor aan op het board, open de seriële monitor, en het zal net zo nauwkeurig werken als het deed op een Arduino Mo Pro of DUE. Ter vergelijking kunt u digitale pin 9 monitoren om te zien hoe lang de `calculateOutput()`-methode voor de berekeningen nodig heeft.

Hoe gaat het verder?

Hoe gaan we van hieruit verder? Welnu, we hebben een functioneel MLP neuraal netwerk dat zowel op een PC als op een microcontroller werkt. We hebben ook een turbo-leerproces om de gewichtsfactoren te genereren die nodig zijn in microcontrollertoepassingen. U kunt proberen dit MLP toe te passen op andere taken die te moeilijk zijn om met `if-else`-statements en vaste grenswaarden op te lossen. Het is misschien zelfs mogelijk een eenvoudig spraakherkenningsproject uit te voeren dat een paar woorden herkent. U zou ook het volgende kunnen onderzoeken:

- De Neural-klasse gebruikt het gegevenstype `double`. Kan het sneller met `float` bij gelijkblijvende nauwkeurigheid? En hoeveel sneller?
- De sigmoid-functie gebruikt de wiskundige functie `exp()` en berekent de macht met het doorgegeven argument. Kan de activeringsfunctie worden vereenvoudigd door een benadering te gebruiken die toch een nauwkeurige classificatie oplevert?
- Als u met spraakherkenning wilt gaan experimenteren, hoe zou u dan een stemmonster voorbereiden om het aan deze MLP-implementatie aan te bieden?
- En wat dacht u van een paar grote wijzigingen? Hoe zou u een tweede laag van verborgen knooppunten implementeren? Kunt u een andere activeringsfunctie implementeren?

In deze serie artikelen hebben we veel onderwerpen behandeld. We hebben gezien hoe het onderzoek naar kunstmatige neuronen begonnen is. Van daaruit hebben we onderzocht hoe MLP's backpropagation gebruiken om te leren, en hebben de werking ervan bestudeerd met behulp van de krachtige verwerkingsmogelijkheden van een PC. Aansluitend hebben we de MLP naar een microcontroller geport als een voorbeeld van *edge ML*. Mocht u deze voorbeelden verder uit willen werken, aarzel dan niet om uw resultaten hier bij Elektor met ons te delen!

210160-D-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via stuart.cording@elektor.com.

Een bijdrage van

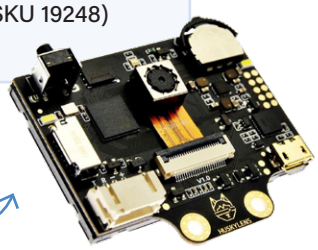
Idee, tekst en foto's: **Stuart Cording**
Redactie: **Jens Nickel, C.J. Abate**
Illustraties: **Patrick Wielders**
Vertaling: **Jelle Aarnoudse**
Layout: **Harmen Heida**

Betere landbouw

De landbouw heeft altijd vertrouwd op de natuur en de seizoenen om de beste tijd te bepalen om te zaaien en te oogsten. Traditie heeft altijd de optimale datum gedicteerd voor het planten van gewassen, om te profiteren van het regenseizoen. Maar door de veranderende weerpatronen hebben de oogsten daaronder geleden. Dankzij AI komt daar nu verandering in. Indiase boeren die deelnamen aan een proefproject, planten aardnoten eind juni, drie weken later dan normaal – op aanraden van de Microsoft Cortana Intelligence Suite. Met behulp van klimaatgegevens uit het verleden leidden de aanbevelingen die de boeren kregen tot een gemiddeld 30% hogere opbrengst [6].

GERELATEERDE PRODUCTEN

- E-boek: B. van Dam, Artificial Intelligence (SKU 18090)
  www.elektor.nl/18090
- Google AIY Vision Kit for Raspberry Pi (SKU 19365)
  www.elektor.nl/19365
- HuskyLens AI Camera with Silicone Case (SKU 19248)
  www.elektor.nl/19248



WEBLINKS

[1] M. Patrick, "ML at the Edge: a Practical Example," codemotion, september 2020: https://bit.ly/2ZQYipU
[2] RGB kleursensor met IR-filter en witte LED – TCS34725: http://bit.ly/2NKFS7T
[3] GitHub-repository – simple-neural-network: https://bit.ly/2ZHLv9p
[4] GitHub-repository – Adafruit_TCS34725: http://bit.ly/37RJg81
[5] "Writing a Library for Arduino," Arduino: http://bit.ly/3aWeNaP
[6] "Digital Agriculture: Farmers in India are using AI to increase crop yields," Microsoft News Center, november 2017: http://bit.ly/2PacsQZ