c't

SPASS MIT TECHNIK

Einfache Computerprojekte zum Selbermachen

Lego-Roboter bauen & steuern

Maschinelles Lernen mit Lego Mindstorms
Spielzeugroboter mit dem Raspberry Pi

In Minecraft coden

3D-Klötzchenwelt als Lernhilfe

Songs arrangieren

Mit Google Song Maker experimentieren

Mit Licht malen

Tolle Fotos mit der Taschenlampe

Text-Adventure programmieren

Einsteigerfreundliches Abenteuerspiel gestalten

Fidget-Spinner selber machen

Konstruieren und mit dem 3D-Drucker ausdrucken

www.ctspecial.de

Coole Kreativ-Projekte

Trickfilme animieren • Comics gestalten • Podcasts produzieren

```
n_hidden_1 = 6  # Anz. verst. Neuronen
w_h1 = tf.Variable(tf.random_normal(
            [n_input, n_hidden_1]))
b_h1 = tf.Variable(tf.random_normal(
            [n_hidden_1]))
w_out = tf.Variable(tf.random_normal(
            [n_hidden_1,n_classes]))
b_out = tf.Variable(tf.random_normal(
            [n_classes]))
```

Sobald TensorFlow die Variablen kennt, können Sie im Python-Code damit rechnen. Grundrechenarten versteht TensorFlow einfach so, für komplexere Aufgaben wie Matrixmultiplikationen stehen Funktionen bereit. Die Formel für die Aktivierungen der ersten Schicht des Netzwerks sieht in TensorFlow- Python so aus:

```
h1_lin = tf.matmul(x, w_h1) + b_h1
h1_act = tf.nn.relu(h1_lin)
```

Mit diesen Aktivierungen rechnen Sie einfach weiter, für die Ausgabeschicht:

```
out = tf.matmul(h1_act, w_out) + b_out
```

Trainieren

Damit haben Sie schon das ganze neuronale Netz definiert. Zum Optimieren fehlt aber noch eine Funktion, die die Qualität der Ausgaben des Netzes bewertet. Im Prinzip können Sie dafür eine beliebige Funktion definieren – sie sollte nur stets für Ausgaben, die besser zu den Trainingsdaten passen, kleinere Werte ausgeben.

Die Arbeit können Sie sich meist sparen, denn TensorFlow bringt die üblichen Funktionen für diese Aufgabe gleich mit. Wir haben uns für `tf.nn.softmax_cross_entropy_with_logits_v2()` entschieden. Diese Funktion eignet sich nur für den Fall, dass man genau eine korrekte Klasse erwartet. Sie normiert die Aktivierungen der Ausgabeschicht erst so, dass sie in der Summe 1 ergeben (wie bei einer Wahrscheinlichkeitsverteilung) und berechnet die negative Summe aus dem Produkt aus der Wahrscheinlichkeit aus den Trainingsdaten mit dem Logarithmus der Wahrscheinlichkeit aus der Vorhersage des Netzwerks. Dieser unter dem Namen „Kreuzentropie“ bekannte Wert stammt aus der Statistik und ist ein übliches Maß für die Abweichung zwischen zwei Wahrscheinlichkeitsverteilungen.

TensorFlow geht davon aus, dass Sie nicht nur mit einzelnen Datensätzen trainieren, sondern gleich mehrere (einen „Batch“) durch das Netzwerk schleusen. Für die Qualität der Ausgaben eines ganzen Batches müssen Sie den Mittelwert der einzelnen Kreuzentropie-Werte berechnen:

```
cost =tf.reduce_mean(tf.nn.softmax_
      ↳cross_entropy_with_logits_v2(
            logits=out, labels=y))
```

Die Variable `cost` kann nun einen Optimierungsalgorithmus verwenden, um diejenigen Gewichte und Biases zu finden, bei denen `cost` besonders klein wird. TensorFlow bringt einen Zoo von Optimierungsalgorithmen mit. Sie arbeiten alle nach dem Grundprinzip des Gradientenabstiegs (siehe [1]) und unterscheiden sich vor allem darin, wie viele Schritte sie brauchen, um gute Werte zu finden. Wir haben mit dem Adam-Algorithmus in Standardeinstellung gute Ergebnisse erzielt:

```
optimizer =tf.train.AdamOptimizer(
  learning_rate=0.001).minimize(cost)
```

Die Lernrate gibt an, wie groß die Schritte sind, die der Algorithmus geht. Ist sie zu groß, findet er keine guten Parameter, da er das Optimum ständig überspringt. Ist sie zu klein, dauert das Training zu lang. Wie Sie Hyperparameter wie die Lernrate optimieren, erklären wir unter [2].

Nun fehlen dem Netzwerk noch die Daten, die Sie mit nur einem Numpy-Befehl aus `.csv`-Dateien laden:

```
import numpy as np
xTrain=np.loadtxt(
  'xTrain_TwoCubesCylinder375-24.csv')
yTrain=np.loadtxt(
  'yTrain_TwoCubesCylinder375-3.csv')
```

Bevor TensorFlow mit diesen Daten an die Arbeit gehen kann, müssen Sie eine `Session` erzeugen und initialisieren:

```
with tf.Session() as sess:
  init = tf.global_variables_
                    ↳initializer()
  sess.run(init)
```

Dieser Session können Sie danach mit `sess.run()` Arbeit geben. Dafür teilen Sie der Funktion mit, welche Liste an Variablen sie berechnen soll und als zweiten Parameter, welche Werte sie für die Platzhalter einsetzen soll. TensorFlow findet selbstständig heraus, welche Zwischenwerte es dafür berechnen muss und optimiert die internen Berechnungen, ohne dass Sie sich darum kümmern müssen.

Um einen Trainingsschritt mit einem einzelnen Merkmalsvektor zu berechnen, fordern Sie von TensorFlow das Ergebnis der Variable `optimizer` und übergeben die Zahlen für `x` und `y` als `feed_dict`:

```
_ = sess.run([optimizer], feed_dict={
                    x: [xTrain[0]],
                    y: [yTrain[0]]})
```

Der Rückgabewert dieses Befehls ist eine interne Repräsentation der Veränderung an den Parametern des Netzwerks, die der Algorithmus aber schon automatisch vorgenommen hat – den Wert kann man also getrost verwerfen.

Fordert man weitere Werte von Variablen wie `cost` an, gibt `sess.run()` diese in einem Tupel zusätzlich zurück. Da TensorFlow die Variable `cost` ohnehin für die Optimierung berechnet, erhöht das Anfordern dieses Werts die Rechenzeit nicht. Um 250 Mal für jeden Merkmalsvektor je einen Schritt zu trainieren und den aktuellen Trainingsstand auszugeben, reichen zwei Python-Schleifen:

```
for i in range(250):
  for j in range(num_examples):
    _, c = sess.run([optimizer, cost],
          feed_dict={x: [xTrain[j]],
                     y: [yTrain[j]]})
  if i % 25 == 0:
    print('epoch {0}: cost = {1}'
                      .format(i, c))
print('epoch {0}: cost = {1}'
                      .format(i, c))
print('Training beendet.')
```

Starten Sie im Anschluss das Training mit `python train_NN.py` ruhig mehrmals. Bei der Initialisierung mit Zufallszahlen können Sie nämlich Glück oder Pech haben. Im schlimmsten Fall kommt das Netzwerk nie übers Raten hinaus (Accuracy 33 Prozent). Oft stabilisiert es sich im Bereich zwischen 80 Prozent und 90 Prozent. Mit etwas Glück kommen Sie auf 95 Prozent – die Parame-

ter dieses Durchlaufs sollten Sie auf den Legostein übertragen.

Python zu Lego

Mit den Parametern, die das Netzwerk nach dem Training kennt, kann auch ein neuronales Netz auf dem Legostein neue Objekte auseinanderhalten. Dafür müssen Sie die aber in einem Format exportieren, das die Lego-Software lesen kann. Zuerst bringen Sie TensorFlow mit der Methode `eval()` dazu, die Werte der Variablen preiszugeben:

```
wh1 = w_h1.eval(sess)
bb1 = b_h1.eval(sess)
wo = w_out.eval(sess)
bo = b_out.eval(sess)
```

Die Lego-Software liest `.rtf`-Dateien, die zuerst in zwei Zeilen die Größe der Matrix angeben und anschließend alle Werte stumpf als eigene Zeilen anhängen. Um dieses Format zu erzeugen, stellt Numpy einige praktische Funktionen bereit:

```
tmpArray=np.reshape(wh1,
       (wh1.shape[0] * wh1.shape[1],))
result=[wh1.shape[0],
        wh1.shape[1]] + tmpArray.tolist()
savetxt('NNweights_h1.rtf',
        result, fmt='%10.8f',
        delimiter='\r', newline='\r')
result=[1,bb1.shape[0]] + bb1.tolist()
savetxt('NNbiases_b1.rtf',
        result, fmt='%10.8f',
        delimiter='\r', newline='\r')
tmpArray=reshape(wo,
         (wo.shape[0] * wo.shape[1],))
result=[wo.shape[0],
        wo.shape[1]] + tmpArray.tolist()
savetxt('NNweights_out.rtf',
        result, fmt='%10.8f',
        delimiter='\r', newline='\r')
result=[1,bo.shape[0]] + bo.tolist()
savetxt('NNbiases_out.rtf',
        result, fmt='%10.8f',
        delimiter='\r', newline='\r')
```

Die beiden `reshape()` hängen die Zeilen der Gewichtsmatrizen hintereinander, sodass eine lange Liste entsteht. Die Variable `result` enthält jeweils die Größe der Matrizen (die ersten zwei Zahlen) und danach die Liste aller Werte. Die Funktion `savetxt()` speichert diese Liste in einer Datei mit einem Wert pro Zeile, wobei das Carriage-Return-Zeichen hier als Zeilentrenner dient.

Auf den Legostein bekommen Sie diese Dateien mit dem in die Entwicklungsumgebung integrierten Datei-Browser. Dafür speichern Sie sie am einfachsten erst auf einer Micro-SD-Karte, stecken die in den Stein und verschieben die Dateien dann mit der Mindstorms-Software.

Programmieren mit Blöcken

Lego setzt für den einfachen Einstieg auf eine komplett grafische Programmiersprache. In der bauen Sie lange Ketten an Blöcken, die beispielsweise Motoren anschalten, Sensoren auslesen oder Variablen belegen. Damit alles schön „einfach“ bleibt, integriert Lego nur die Grundrechenarten und ganz einfache Array-Operationen. Eine Matrixmultiplikation oder -addition wird damit eine längere Sache. Die abgebildeten Blöcke geben Ihnen einen Eindruck, wie solche Operationen in der Lego-Sprache aussehen. Glücklicherweise kann man eigene Blöcke definieren, die wie Funktionen in herkömmlichen Programmiersprachen mit beliebigen Eingaben rechnen.

In der Projektdatei unter ct.de/wd4u haben wir schon Blöcke für alle Matrixoperationen definiert, die Sie brauchen, um das neuronale Netz zu berechnen. Der grafische Code tut dasselbe wie der vorher beschriebene Python-Code – nur ohne Framework und etwas langsamer: Ein Durchlauf dauert etwa 180 Millisekunden.

Sogar auf der kleinen Hardware des Lego-Roboters läuft das kleine Netz schnell genug. Damit das klappt, mussten wir uns auf wenige Eingabedaten beschränken und ein Netz mit nur wenigen Schichten und wenigen Neuronen pro Schicht bauen. Da das Prinzip aber immer gleich funktioniert, können Sie mit stärkerer Hardware auf die gleiche Weise auch größere Netze berechnen. Ein Raspberry Pi 3 mit vier Kernen und 1,2 GHz Taktfrequenz ist beispielsweise schon stark genug für kleine Netzwerke zur Bilderkennung. Auf dem läuft dann auch schon direkt TensorFlow, sodass Sie für Training und Inferencing (Anwenden des trainierten Netzes) denselben Code verwenden können.

Egal ob Sie ein neuronales Netz trainieren, um direkt mit den Sensordaten zu arbeiten (Deep Learning), oder wie beim klassischen maschinellen Lernen einen Vektor mit relevanten Datenpunkten per Hand zusammenstellen, es gilt: „Small is beautiful“. Für ein kleines Netz müssen Sie weniger Trainingsdaten sammeln und es trainiert schneller. Achten Sie auf jeden Fall darauf, die Trainingsdaten unter den gleichen Bedingungen zu erstellen, unter denen Sie das Netz auch anwenden möchten. Der BrickClassifi3r erkennt beispielsweise Testobjekte in einer anderen Farbe wesentlich schlechter, da sie mehr oder weniger Licht an den Infrarotsensor zurückwerfen. Auch das Umgebungslicht spielt eine Rolle: Unser Aufbau arbeitet bei Kunstlicht schlechter, da wir ihn im Sonnenlicht trainiert haben. Sie können ein Netz durchaus trainieren, mit solchen Unterschieden umzugehen, allerdings brauchen Sie dann wieder mehr Daten und mehr Zeit fürs Training.

Probieren Sie mit dem BrickClassifi3r ruhig ein bisschen herum. Neben den Trainingsparametern und der Netzwerkstruktur lädt auch die Lego-Konstruktion zu Experimenten ein: Was passiert, wenn Sie die Objekte mit zwei Sensoren erfassen? Hilft ein Scanner-Tunnel, wenn sich die Lichtverhältnisse ändern? Sind mit mehr Trainingsdaten auch mehr als 95 Prozent Erkennungsrate möglich? Schreiben Sie es uns, wenn Sie es herausgefunden haben. Sie erreichen uns per Mail, über die Kommentare auf der Artikelseite oder auf der Mindstorms-Community-Seite zum Projekt. (pmk) c't

Literatur

[1] Andrea Trinkwalder, Netzgespinste, Die Mathematik neuronaler Netze: einfache Mechanismen, komplexe Konstruktion, c't 6/2016, S. 130

[2] Pina Merkert, Bilder skalieren mit TrenserFlow, c't Python-Projekte (2018), S. 116

Repository, Dokumentation:
www.ct.de/wd4u

DEVELOPER-KONFERENZEN + -WORKSHOPS 2018

Data Science & Machine Learning

Termin: 25.-27.09.2018
Ort: Print Media Academy, Heidelberg

Sichere Software- & Webentwicklung

Termin: 16.-18.10.2018
Ort: Print Media Academy, Heidelberg

Java, .NET & JavaScript

Termin: 04.-06.09.2018
Ort: Techn. Hochschule Georg Simon Ohm, Nürnberg

Docker, Kubernetes & Co.

Termin: 13.-16.11.2018
Ort: Congress Center Rosengarten, Mannheim

Continuous Delivery & DevOps

Termin: 13.-16.11.2018
Ort: Congress Center Rosengarten, Mannheim

Cloud Native Computing

Termin: 12.-14.11.2018
Ort: Euston Square 30, London

data2day / 2018

[Container Conf]

// heise devSec()

» Continuous Lifecycle »

Herbstcampus

SERVERLESS COMPUTING

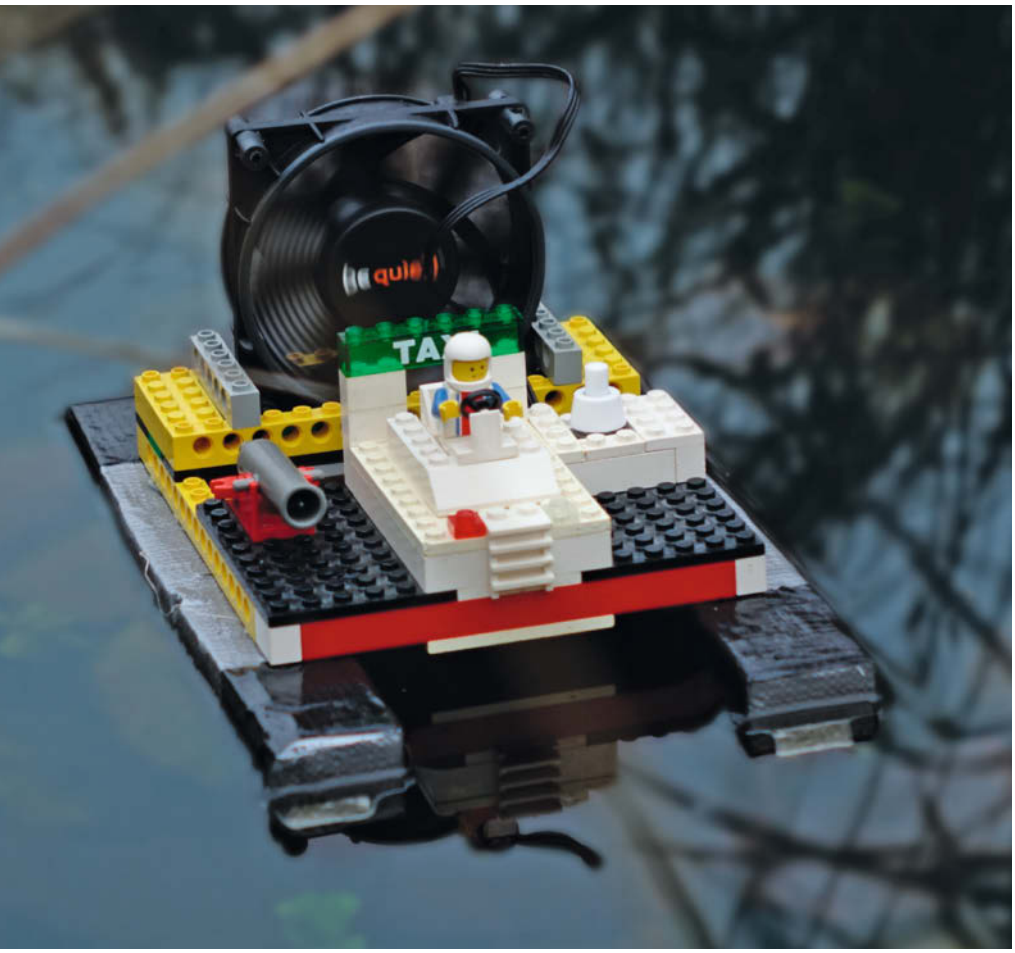
Veranstalter:

Weitere Informationen unter:

www.heise.de/developer/

Martin Reche

Katamaran mit PC-Lüfter-Antrieb



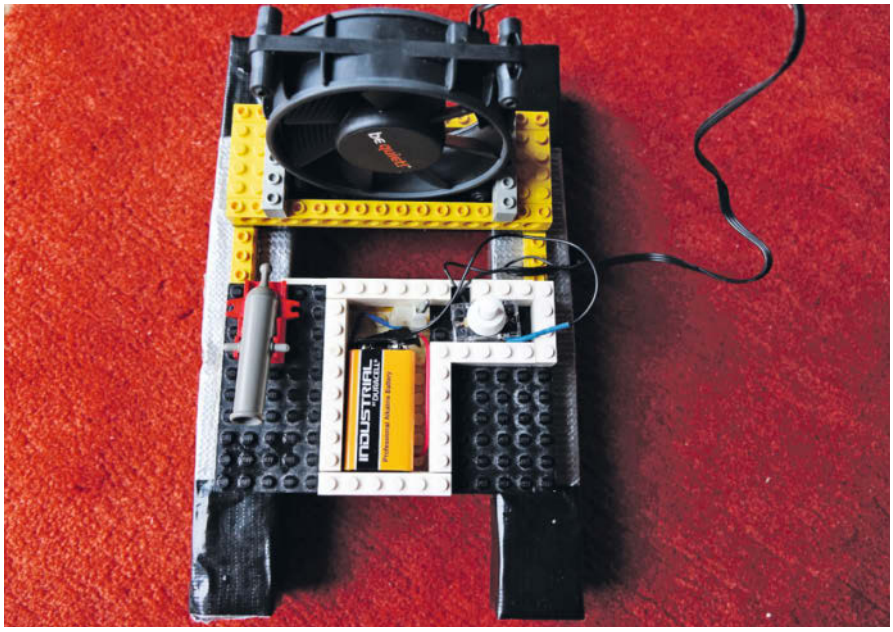
Spielzeugboote mit Motor bekommt man im Laden an der Ecke. Aber das ist langweilig. Und kostet Geld. Selbst machen lautet die Devise! Für einen in Eigenregie gebastelten Katamaran mit PC-Lüfter-Antrieb braucht man im Idealfall weder viele Vorkenntnisse noch Geld noch exotische Bauteile.

Selbst machen ist in! Im Internet findet man jede Menge Anleitungen für Bau- und Bastelprojekte, von intelligenter Beleuchtung bis zur Retro-Arcade-Machine. Bevor Sie selbst aktiv werden und sich an komplexe Maker-Projekte wagen, starten Sie mit einem einfachen Vorhaben, das garantiert gelingt. Dafür bietet sich beispielsweise ein Wasserfahrzeug mit PC-Lüfter-Antrieb an. Das baut man ohne große Vorkenntnisse zusammen; die Zutaten wie PC-Lüfter, Schalter sowie Anschlusskabel für die Batterie findet man in ausrangierten Computern und Elektrogeräten im Keller. In unserem Beispiel entsteht in Anlehnung an eine Bastel-Anleitung der Nachwuchs-Maker-Website Tuduu.org [1] ein Wasserfahrzeug auf Styropor-Basis, genauer: ein Katamaran. Für ein individuelles Aussehen lässt sich das Gefährt mit Lego-Steinen pimpen.

Umweltfreundlicher Elektroantrieb

Im ersten Schritt bauen Sie aus wenigen Komponenten einen einfachen Stromkreis für den Antrieb zusammen. So wissen Sie später, wie viel Platz die Elektronik an Deck einnehmen wird. Der Stromkreis besteht aus einer Batterie (9V-Block) mit Clip, einem Ein-Aus-Schalter und einem ausrangierten PC-Lüfter. Der Lüfter sollte mit mindestens 1400 Umdrehungen pro Minute arbeiten, damit er sein eigenes Gewicht und das von Batterie und Boot bewegen kann – es dürfen aber auch gerne deutlich mehr Umdrehungen pro Minute sein. Ein möglichst leichter Lüfter ist empfehlenswert; alte Exemplare muss man vor dem Einsatz gegebenenfalls reinigen. Im Beispiel haben wir die Minuspole von Batterie und Lüfter mit einer Lüsterklemme angeschlossen. Den Schalter verbindet man in einer Richtung mit dem Pluspol der Batterie, in der anderen mit dem 12-Volt-Anschluss des Lüfters. Jetzt kann man den „Motor“ des Bootes bequem ein- und ausschalten. Sollten Sie sich bei der Verkabelung des Lüfters unsicher sein, hilft beispielsweise die Website „Elektronik-Kompendium“ weiter (siehe ct.de/wqxw). Dort finden Sie die gängigsten Lüfteranschlüsse samt Kabelbelegungen.

Im nächsten Schritt entstehen die Schwimmkörper für den Katamaran. Dafür schneiden Sie vier gleich lange Streifen von einer Styroporplatte ab; zwei breitere und zwei schmalere. Heißkleber und Gewebeband verbinden je einen schmalen mit einem breiten Styropor-



Die Elektronik verschwindet in einem Kasten im Rumpf des Katamarans, der sie vor Spritzwasser schützt.

streifen. Im Beispiel haben wir, ebenfalls mit Heißkleber, auf jedem Schwimmkörper eine lange Lego-Technic-Stange befestigt. Diese Stangen dienen als Basis für die Legoplatten. So kann man später bei Bedarf mit immer neuen Aufbauten aus den bunten Steinchen experimentieren. Alternativ zurren Sie die Stangen mit Gummibändern an Deck fest und verhindern so, dass das Lego ins Wasser rutscht. Dabei bleiben die Legoplatten dann vom Heißkleber verschont.

Gleichgewicht halten

Fixieren Sie nun den Lüfter hochkant an Deck. Platzieren Sie ihn so, dass sein Gewicht etwas nach hinten verlagert wird, das Boot dennoch möglichst sicher und austariert im Wasser liegt. Dabei hilft auch das Gewicht der Batterie, die ihren Platz etwas weiter vorne im Bug finden sollte. Probieren Sie verschiedene Positionen aus, bis es passt. Bei besonders schlanken Konstruktionen lohnt sich der zusätzliche Einsatz von Pontons an beiden Auslegern. Als Schwimmkörper kommen unter anderem kleine Plastikflaschen, Ü-Ei-Schalen, Korken und leere Zigarrenhülsen in Frage. Wichtig ist, dass man die Schwimmkörper wasserdicht versiegelt. Der Katamaran im Beispiel kommt nach ein wenig zusätzlicher Tüftelarbeit ohne Pontons aus.

Vor der Jungfernfahrt verfrachten Sie die Batterie und den Großteil der Verkabelung in ein spritzwassergeschütztes Gehäuse. Eine kleine Keks- oder Bonbon-Dose reicht dafür aus. Der Schalter bleibt über eine Aussparung im Deckel erreichbar. Lego-Fans klickern sich einen solchen Kasten selbst zusammen. Dabei müssen Sie nur aufpassen, dass dieser nicht zu schwer wird. Ansonsten schwimmt das Gesamtkonstrukt zwar, kommt aber nicht so richtig vom Fleck.

Dann naht der spannende Moment: Schwimmt das Boot oder geht es unter? Kleiner Tipp: Vor dem Einsatz in der freien Natur sollte der Stapellauf in der heimischen Badewanne erfolgen, falls Nachjustierungen nötig werden. Wir hatten Glück, unser Katamaran ist nicht untergegangen und hat auf dem Tümpel wie erhofft Fahrt aufgenommen – siehe Online-Video auf ct.de/wqxw. Zugegeben fährt er noch recht langsam und ein wenig wackelig, aber das Wichtigste ist: Das Bastelprojekt war erfolgreich.

Feintuning

Selbst wenn das Ergebnis noch nicht perfekt ist, motivieren solche erfolgreichen Basteleien die Neugier am Tüfteln und Selbermachen. Im nächsten Schritt nimmt man sich den Antrieb vor und sucht nach Möglichkeiten, ihn zu tunen – ambitionierte Bastler arbeiten mit stärkeren Lüftern und mehreren Batterien, um für schnellere und längere Ausfahrten zu sorgen. Optional bringt man zu einem späteren Zeitpunkt einen Einplatinenrechner wie den Arduino ins Spiel und setzt diesen für die passende Bootsbeleuchtung ein. Ausrangierte Computertechnik findet ein zweites Leben als Verzierung an Deck und eine bunte Lackierung sorgt für den letzten Schliff. Kurzum: Das Projekt lässt sich fast beliebig weiterspinnen. Vielleicht finden sich später im Freundeskreis Mitstreiter, die ebenfalls das Bastelfieber packt, sodass sie in einem Rennen die Leistung ihrer eigenen Konstruktionen mit der Ihres Bootes messen wollen. (mre) c't

Tipp für Eltern und Kinder

Spielzeug-Katamaran mit PC-Lüfter-Antrieb bauen

Styropor, PC-Lüfter, Gewebeband, Ein-Ausschalter, Kabel, 9V-Block, Batterie-Clip, optional Lego und Gummibänder

Etwas handwerkliches Geschick ist von Vorteil, ebenso wie Basiswissen über Stromkreisläufe.

Ein einfaches Boot baut man in weniger als einer Stunde zusammen.

Kinder ab sechs Jahren arbeiten im Team mit den Eltern, Kinder ab zehn Jahren legen alleine los.

Das Projekt lässt sich im Idealfall komplett aus Recycling-Materialien herstellen – für den 9V-Block sollte man rund zwei Euro einplanen.

Literatur

[1] Martin Reche, Löten, nähen, programmieren, Projektideen für Nachwuchs-Maker auf Tuduu.org, c't 21/2016, S. 138

Zitierte Webseiten, Informationen, Video von der Jungfernfahrt: ct.de/wqxw