Stap-voor-stap Handleiding: Neuraal Netwerk in TensorFlow

Uitleg en toelichting op Python-code voor het ontwikkelen van een AI-regelaar ter vervanging van een PID-systeem

Mitchel Reints - 1040953@hr.nl

Hogeschool Rotterdam - Elektrotechniek - PEE51

14 juni 2025

1 introductie

In deze handleiding wordt stap voor stap uitgelegd hoe het neuraal netwerk voor PEE51 project is opgesteld. Dit project is een onderdeel van de opleiding Elektrotechniek aan de Hogeschool Rotterdam. Het doel van deze handleiding is om stap voor stap uit te leggen hoe een neuraal netwerk ontwikkeld, getraind en geëvalueerd kan worden in Python met TensorFlow. Dit netwerk is bedoeld om het gedrag van een klassieke PID-regelaar na te bootsen of te verbeteren in een systeem waarbij een pingpongbal op een ingestelde hoogte wordt gehouden door middel van een ventilator. De aanpak richt zich op het bouwen van een werkend AI-model, met als einddoel het inzetten van dit model op een embedded platform. Daarbij ligt de nadruk op nauwkeurige en stabiele hoogtecontrole, net als bij de bestaande PID-regeling. In deze handleiding wordt uitsluitend ingegaan op de technische implementatie van het neuraal netwerk; aspecten zoals gebruikersinterface, documentatie of demonstratieopstelling blijven buiten beschouwing. Voor meer informatie over het prestatieonderzoek en de vergelijking met de PID-regelaar, zie het bijbehorende onderzoeksrapport: https://example.com/onderzoek-ai-regelaar.

2 Benodigheden

Voor dat we aan de slag kunnen met het neuraal netwerk, zijn er een aantal benodigdheden die we moeten installeren. Voor het installeren van de juiste packages en het opzetten van de omgeving, is het belangrijk om te zorgen dat je de juiste versie van Python en de benodigde bibliotheken hebt. De handleiding gaat ervan uit dat je bekend bent met de basisprincipes van Python en hoe je een Python-omgeving kunt opzetten. Als je nog niet bekend bent met Python, raden we aan om eerst een introductiecursus te volgen of de officiële documentatie te lezen. Deze benodigdheden zijn:

Deze benodigdheden zijn essentieel voor het ontwikkelen, trainen en evalueren van het neuraal netwerk. Hieronder worden deze benodigdheden kort toegelicht:

- **Python 3.10 of hoger:** De programmeertaal waarin het neuraal netwerk wordt ontwikkeld. Zorg ervoor dat je de nieuwste versie van Python hebt geïnstalleerd.
- **TensorFlow 2.11 of hoger:** Een populaire open-source bibliotheek voor machine learning en deep learning, die wordt gebruikt om neurale netwerken te bouwen en te trainen.
- **NumPy:** Een fundamentele bibliotheek voor wetenschappelijk rekenen in Python, die wordt gebruikt voor het werken met arrays en wiskundige functies.
- **Matplotlib:** Een bibliotheek voor het maken van visualisaties in Python, die nuttig is voor het plotten van resultaten en het visualiseren van data.
- **Jupyter Notebook (optioneel):** Een interactieve omgeving voor het schrijven en uitvoeren van Python-code, die handig is voor het ontwikkelen en testen van het neuraal netwerk.
- **Een teksteditor of IDE:** Een omgeving waarin je de Python-code kunt schrijven en uitvoeren. Populaire keuzes zijn Visual Studio Code, PyCharm of Jupyter Notebook.

Tijdens het doorlopen van de stappen in deze handleiding worden alle geïmporteerde pakketten en hun toepassing in de code uitvoerig toegelicht, zodat duidelijk wordt waarom elk pakket nodig is.

Versies van gebruikte pakketten

Voor dit project zijn de volgende pakketversies gebruikt om compatibiliteit en reproduceerbaarheid te waarborgen:

- **Pandas:** 2.2.3
- **NumPy:** 1.26.4
- **Matplotlib:** 3.10.1
- **TensorFlow:** 2.15.0
- **Keras:** 2.15.0

De onderstaande pakketten zijn in dit project gebruikt en worden in de code geïmporteerd. Zorg ervoor dat je ze allemaal installeert voordat je aan de slag gaat:

- **pandas, numpy, matplotlib:** voor data-analyse en visualisatie
- **tensorflow, keras:** voor het bouwen, trainen en evalueren van het neuraal netwerk
- **scikit-learn:** voor dataset splitsing en normalisatie

Hieronder staat een voorbeeld van de benodigde imports in Python. Deze code moet aan het begin van je Python-script of Jupyter Notebook worden toegevoegd om de benodigde bibliotheken te importeren:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras import layers, models, callbacks, preprocessing
from sklearn.model_selection import train_test_split
from keras.layers import Input, Dense, Conv1D, Flatten
from keras.models import Model, Sequential
from sklearn.preprocessing import MinMaxScaler
import keras
```

3 datavoorbereiden

In deze sectie wordt stap voor stap uitgelegd hoe de data wordt voorbereid voor het trainen van het neuraal netwerk. De data bestaat uit een CSV-bestand met de volgende kolommen: tijd, hoogte, snelheid, en ventilatorsnelheid.

```python
CSV_PATH = '/Users/username/Documents/PEE51/data/data.csv'
CSV_DATA = pd.read_csv(CSV_PATH)
# Toon aantal samples
total_samples = len(CSV_DATA)
```

```python
# Selecteer features en target
features = CSV_DATA[['Setpoint (m)', 'Hoogte (m)', 'Fout',
      'Fout_Integratie', 'Fout_Afgeleide']].to_numpy()
target = CSV_DATA['PWM'].to_numpy()
```

```python
n_input = 2  # aantal tijdstappen per voorbeeld

X, y = [], []

for i in range(len(features) - n_input):
```

```
    X.append(features[i:i + n_input])
    y.append(target[i + n_input])  # de PWM bij het volgende moment

X = np.array(X)  # shape: (samples, n_input, features)
y = np.array(y)

```

```
print("X shape:", X.shape)  # (samples, tijdstappen, features)
print("y shape:", y.shape)  # (samples,)
print("X:", X)  # (samples, tijdstappen, features)
print("y:", y)
```

4 Model bouwen en compileren

```
model = Sequential([
    Conv1D(filters=32, kernel_size=2, activation='relu', input_shape=(n_input, X.shape[2])),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1)  #   n   PWM-waarde voorspellen
])

model.summary()
```


```
model = Sequential([
    Conv1D(filters=32, kernel_size=2, activation='relu', input_shape=(n_input, X.shape[2])),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1)  #   n   PWM-waarde voorspellen
])

model.summary()
```


5 Model compileren

6 Model trainen

7 Model evalueren

8 model opslaan/exporteren

Referenties

[1] Mike Bowker and Phil Williams. Helsinki and west european security. *International Affairs*, 61(4):607–618, 1985.

Begrippenlijst

PID-regelaar Een regelsysteem dat gebruikmaakt van proportionele, integrale en afgeleide termen.

Neuraal netwerk Een algoritme geïnspireerd op het menselijk brein dat patronen leert herkennen.

Backpropagation Methode om foutmarges in een neuraal netwerk terug te voeren en gewichten aan te passen.

For more details, refer to[?].

A Meetresultaten

In deze bijlage worden de meetresultaten weergegeven die zijn verzameld tijdens de experimenten.

B Tabel met resultaten

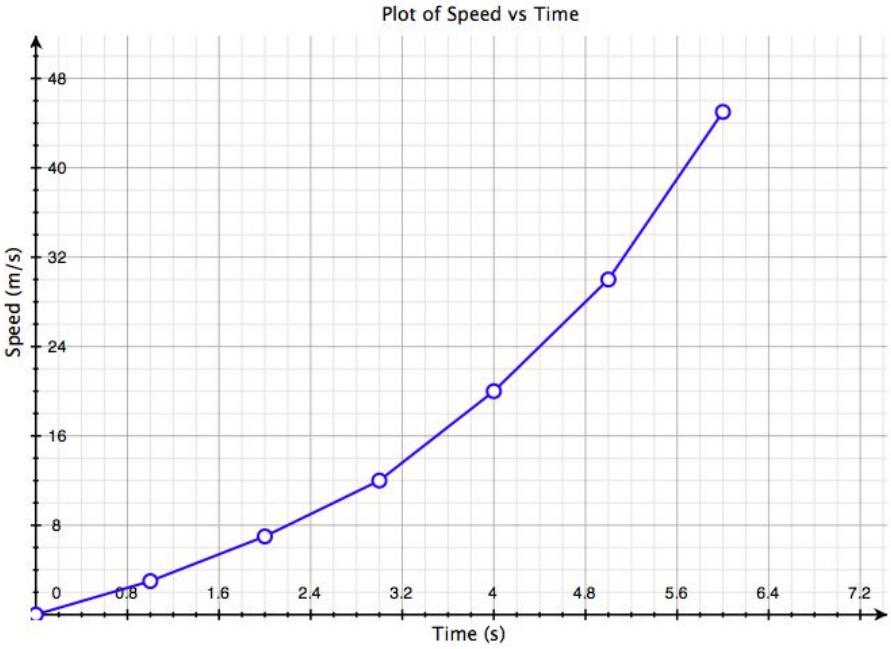
Hieronder staat een overzichtstabel van de gemeten waarden.

Tijd (s)	Spanning (V)	Stroom (A)
0.0	3.30	0.10
1.0	3.28	0.11
2.0	3.27	0.12

Tabel 1: Gemeten waarden tijdens test 1

C Grafiek

Hier zie je een voorbeeld van een grafiek (je moet je eigen figuur invoegen):



Figuur 1: Spanning over tijd

D Broncode

E python-script

```
    def build_feedforward_model(input_layer_size, hidden_layer_size, output_layer_size):
        """Bouw een neuraal netwerkmodel volgens TensorFlow-conventies."""
        model = keras.Sequential(name="Feedfoward_Model")
        model.add(layers.Input(shape=(input_layer_size,), name="Input_Layer"))
        model.add(layers.Dense(hidden_layer_size, activation='sigmoid', name="Hidden_Layer"))
        model.add(layers.Dense(hidden_layer_size_2, activation='sigmoid', name="Hidden_Layer_2"))
        model.add(layers.Dense(output_layer_size, activation='softmax', name="Output_Layer"))
        return model
```