Kurs: KI T-InfT-008 und 010 Datenmengen und Embedded Systems

Cândido Vieira

02.09.2024

Balthasar-Neumann-Technikum (BNT)

Inhaltsverzeichnis

1. Einführung und Kursübersicht

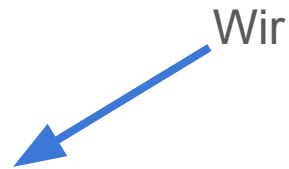
- Kurze Vorstellung

- Überblick über den Kurs: Ziele, Inhalte und erwartete Ergebnisse

1. Einführung und Kursübersicht

- Kurze Vorstellung

- Überblick über den Kurs: Ziele, Inhalte und erwartete Ergebnisse



1. **Problem definieren**
   - *Was modellieren wir?*
2. **Daten kuratieren (aufbereiten)**
   - *Welche Daten informieren das Modell?*
3. **Architektur entwerfen**
   - *RNN, Autoencoder, DMD, SINDy?*
4. **Lossfunktion gestalten**
   - *Welche Modelle sind „gut“?*
5. **Optimierung anwenden**
   - *Welche Algorithmen trainieren das Modell?*

Quelle: https://www.youtube.com/watch?v=JoFW2uSd3Uo&list=PLMrJAkhIeNNQ0BaKuBKY43k4xMo6NSbBa

2. Grundlagen der Sensoren

Definition von Sensoren:

- Geräte, die physikalische, chemische oder biologische Parameter erfassen und in elektronische Signale umwandeln.

Bedeutung von Sensoren in der Datenerfassung:

- In der modernen Datenanalyse sind Sensoren die primäre Quelle für die Erfassung von Rohdaten, die dann durch verschiedene Algorithmen und Modelle analysiert werden.

RFID
Quelle: https://de.wikipedia.org/wiki/RFID

Methan Detektor
Quelle: https://de.wikipedia.org/wiki/RFID

2. Grundlagen der Sensoren

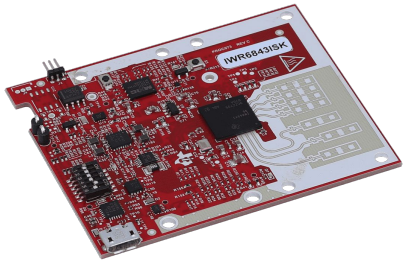
Beispiele für verschiedene Sensortypen:

- **Physikalische Sensoren:** Temperatur, Druck, Feuchtigkeit.
- **Chemische Sensoren:** Gassensoren, pH-Sensoren.
- **Biologische Sensoren:** Glukosesensoren, DNA-Chips.
- **Bildsensoren:** Kameras, Infrarotsensoren.

Druck-Sensor
Quelle: https://de.wikipedia.org/wiki/RFID

Funktionsweise von Sensoren:

- Umwandlung von physikalischen Größen in elektrische Signale.
- Signalverarbeitung und Datenkonvertierung.



60GHz long-range antenna mmWave sensor
Quelle: https://www.ti.com/tool/IWR6843ISK

3. Datenformate und Kommunikationsprotokolle

1. Typische Datenformate:

Datenformate sind entscheidend für die Speicherung und den Austausch von Daten zwischen verschiedenen Systemen. Verschiedene Formate werden je nach Anwendungsfall und Anforderungen an die Datenstruktur, Lesbarkeit und Effizienz verwendet.

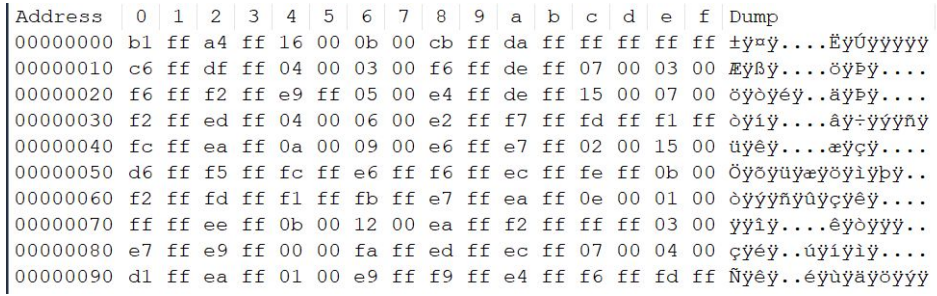
3. Datenformate und Kommunikationsprotokolle

1. Typische Datenformate:

Binärformat (Bin):

- **Beschreibung:** Ein kompaktes, nicht-menschlich lesbares Format, das direkt in binären Daten gespeichert wird. Es wird häufig verwendet, wenn Effizienz und Platzersparnis entscheidend sind.

- **Anwendung:** Sensoren mit hoher Datenrate, Bild- und Audiodaten, industrielle Steuerungen, etc.



Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	b1	ff	a4	ff	16	00	0b	00	cb	ff	da	ff	ff	ff	ff	ff	±ÿ¤ÿ....ËÿÚÿÿÿÿÿ
00000010	c6	ff	df	ff	04	00	03	00	f6	ff	de	ff	07	00	03	00	Æÿßÿ....öÿÞÿ....
00000020	f6	ff	f2	ff	e9	ff	05	00	e4	ff	de	ff	15	00	07	00	öÿòÿéÿ..äÿÞÿ....
00000030	f2	ff	ed	ff	04	00	06	00	e2	ff	f7	ff	fd	ff	f1	ff	òÿíÿ....âÿ÷ÿýÿñÿ
00000040	fc	ff	ea	ff	0a	00	09	00	e6	ff	e7	ff	02	00	15	00	üÿêÿ....æÿçÿ....
00000050	d6	ff	f5	ff	fc	ff	e6	ff	f6	ff	ec	ff	fe	ff	0b	00	Öÿõÿüÿæÿöÿìÿþÿ..
00000060	f2	ff	fd	ff	f1	ff	fb	ff	e7	ff	ea	ff	0e	00	01	00	òÿýÿñÿûÿçÿêÿ....
00000070	ff	ff	ee	ff	0b	00	12	00	ea	ff	f2	ff	ff	ff	03	00	ÿÿîÿ....êÿòÿÿÿ..
00000080	e7	ff	e9	ff	00	00	fa	ff	ed	ff	ec	ff	07	00	04	00	çÿéÿ..úÿíÿìÿ....
00000090	d1	ff	ea	ff	01	00	e9	ff	f9	ff	e4	ff	f6	ff	fd	ff	Ñÿêÿ..éÿùÿäÿöÿýÿ

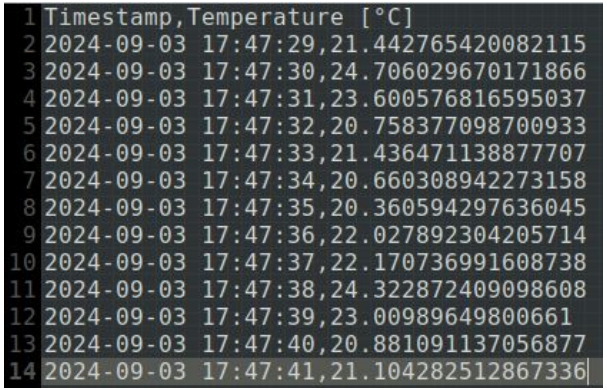
3. Datenformate und Kommunikationsprotokolle

1. Typische Datenformate:

CSV (Comma-Separated Values):

- **Beschreibung:** Ein einfaches, textbasiertes Format, in dem Daten in Form von Zeilen und Spalten organisiert sind. Jede Zeile repräsentiert einen Datensatz, und Spalten sind durch Kommas getrennt.

- **Anwendung:** Tabellenkalkulationen, einfache Datenbanken, Protokollierung von Sensorwerten.



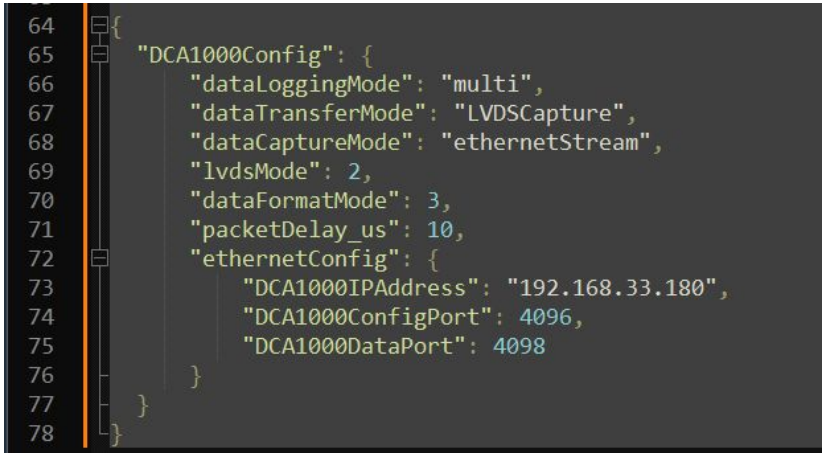
3. Datenformate und Kommunikationsprotokolle

1. Typische Datenformate:

JSON (JavaScript Object Notation):

- **Beschreibung:** Ein leichtgewichtiges, menschenlesbares Format zur Darstellung strukturierter Daten als Schlüssel-Wert-Paare. Es wird oft in Webanwendungen verwendet.

- **Anwendung:** Webservices, APIs, mobile Anwendungen, IoT-Datenübertragung.



3. Datenformate und Kommunikationsprotokolle

1. Typische Datenformate:

XML (eXtensible Markup Language):

- **Beschreibung:** Ein flexibles, textbasiertes Format zur Darstellung hierarchischer Datenstrukturen. Es ist sehr gut lesbar, aber im Vergleich zu JSON etwas umfangreicher.

- **Anwendung:** Webservices, Dokumentenaustausch, Konfigurationsdateien, Kommunikationsprotokolle.

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Quelle: https://www.w3schools.com/xml/xml_whatis.asp

3. Datenformate und Kommunikationsprotokolle

2. Kommunikationsprotokolle:

Kommunikationsprotokolle regeln, wie Daten zwischen Geräten und Systemen ausgetauscht werden. Sie stellen sicher, dass Daten korrekt übertragen und empfangen werden, unabhängig von den zugrunde liegenden Netzwerken oder physischen Verbindungen.

3. Datenformate und Kommunikationsprotokolle

2. Kommunikationsprotokolle:

WebSocket:

- **Beschreibung:** Ein Protokoll, das eine bidirektionale Kommunikationsverbindung zwischen einem Webbrowser und einem Server ermöglicht. Es ist effizienter als herkömmliche HTTP-Kommunikation, da es nach der Einrichtung der Verbindung geringe Latenzzeiten bietet.

- **Anwendung:** Echtzeit-Webanwendungen, Online-Spiele, Chatsysteme, Echtzeitüberwachung von Sensordaten.

3. Datenformate und Kommunikationsprotokolle

2. Kommunikationsprotokolle:

UDP (User Datagram Protocol):

- **Beschreibung:** Ein verbindungsloses Protokoll, das schnelle Datenübertragung ermöglicht, indem es auf Fehlerkorrektur verzichtet. Es ist schneller als TCP, aber weniger zuverlässig.

- **Anwendung:** Echtzeit-Streaming (z.B. Video, Audio), Online-Gaming, Sensornetzwerke, die geringe Latenz erfordern.

3. Datenformate und Kommunikationsprotokolle

2. Kommunikationsprotokolle:

HTTP/HTTPS (Hypertext Transfer Protocol / Secure):

- **Beschreibung:** Ein zustandsloses Protokoll, das für die Übertragung von Webdaten über das Internet verwendet wird. HTTPS ist die verschlüsselte Version und bietet zusätzliche Sicherheit.

- **Anwendung:** Webseiten, Web-APIs, Cloud-Dienste, Datenübertragung von IoT-Geräten.

3. Datenformate und Kommunikationsprotokolle

2. Kommunikationsprotokolle:

MQTT (Message Queuing Telemetry Transport):

- **Beschreibung:** Ein leichtgewichtiges Publish/Subscribe-Protokoll, das besonders für die Kommunikation mit eingeschränkten Geräten und niedrigen Bandbreiten geeignet ist.

- **Anwendung:** IoT-Geräte, Sensornetzwerke, Hausautomation.

4. Einführung in Synthetische Daten

Was sind synthetische Daten?

- Synthetische Daten sind künstlich generierte Daten, die reale Daten imitieren, aber oft aus Simulationen, Modellen oder anderen generativen Verfahren stammen.

Warum sind synthetische Daten wichtig?

- **Datenknappheit:** Synthetische Daten können verwendet werden, um Datensätze zu erweitern, wenn reale Daten schwer zugänglich oder teuer sind.
- **Datenschutz:** Synthetische Daten ermöglichen es, Modelle zu trainieren, ohne sensible oder private Daten zu verwenden.
- **Testing und Validierung:** Synthetische Daten können verwendet werden, um Systeme unter verschiedenen Bedingungen zu testen.

4. Einführung in Synthetische Daten

Kurzer Überblick über Techniken zur Generierung synthetischer Daten:

- **Statistische Methoden:** Verwendung von statistischen Modellen, um Daten zu simulieren.

- **Generative Adversarial Networks (GANs):** Verwendung von tiefen neuronalen Netzen zur Generierung von hochrealistischen Daten, insbesondere bei Bildern und Videos.

- **Simulationsmodelle:** Einsatz von physikalischen oder mathematischen Modellen zur Erzeugung von Daten, z.B. für Wettervorhersagen oder Verkehrsflüsse, etc.

5. Synchronisation in Multimodalen Systemen

Warum ist die Synchronisation wichtig?

- In multimodalen Systemen erfassen verschiedene Sensoren gleichzeitig unterschiedliche Arten von Daten (z.B. Audio, Video, Bewegung). Um diese Daten sinnvoll zu kombinieren und zu analysieren, müssen sie synchronisiert werden.

- Beispielsweise müssen Video- und Audiodaten in einem Überwachungssystem exakt abgestimmt sein, um eine korrekte Ereignisanalyse zu ermöglichen.

5. Synchronisation in Multimodalen Systemen

Herausforderungen bei der Synchronisation:

- **Unterschiedliche Abtastraten:** Verschiedene Sensoren arbeiten mit unterschiedlichen Abtastraten, was zu Zeitverschiebungen führt.

- **Latenzzeiten:** Sensoren können unterschiedlich schnell Daten liefern, was zu Verzögerungen führt.

- **Drift:** Über längere Zeiträume können kleine Unterschiede in den Sensoruhren zu einer zunehmenden Desynchronisation führen.

5. Synchronisation in Multimodalen Systemen

Methoden zur Synchronisation:

Zeitstempel-basierte Synchronisation:

- Jeder Sensor erhält einen präzisen Zeitstempel, wenn die Daten erfasst werden. Diese Zeitstempel werden verwendet, um die Daten beim Zusammenführen zu synchronisieren.

- **Praktisches Beispiel:** Ein Video-Frame und ein Audio-Clip, die beide mit einem gemeinsamen Zeitstempel versehen sind, können zur exakten Wiedergabe synchronisiert werden.

5. Synchronisation in Multimodalen Systemen

Methoden zur Synchronisation:

Sensorfusion:

- Kombination von Daten aus mehreren Sensoren zu einem einzigen Datensatz, wobei die Synchronisation durch Algorithmen wie den Kalman-Filter erreicht wird.

- **Anwendung:** In autonomen Fahrzeugen werden Lidar, Radar und Kameradaten kombiniert, um eine genaue Umfelderkennung zu ermöglichen.

5. Synchronisation in Multimodalen Systemen

Methoden zur Synchronisation:

Latenzausgleich und Korrektur:

- Nutzung von Puffern und Zeitverzögerungskorrekturen, um die Datenströme anzugleichen.

- **Anwendung:** In Echtzeit-Anwendungen wie VR/AR wird die Latenz zwischen verschiedenen Sensoren ausgeglichen, um eine nahtlose Erfahrung zu bieten.

5. Synchronisation in Multimodalen Systemen

Methoden zur Synchronisation:

Best Practices für die Synchronisation:

- Verwende Sensoren, die eine synchrone Taktung unterstützen.
- Implementiere robuste Zeitstempelverfahren und nutze standardisierte Zeitquellen (z.B. NTP).
- Überwache und kalibriere regelmäßig die Synchronisation, besonders bei Langzeitprojekten.

6. Praktische Übung: Programmierung eines einfachen Python-Datenrecorders

Übung 1: Einfache synthetische Daten generieren, speichern und wieder lesen

1. Synthetische Daten generieren:
   1.1. Schreibe ein Python-Skript, das eine Liste von Zufallszahlen erstellt. Diese Zahlen können zum Beispiel Messwerte simulieren, wie Temperatur oder Feuchtigkeit.
2. Daten speichern:
   2.1. Speichere die generierten Daten in einer CSV-Datei.
3. Daten wieder einlesen:
   3.1. Lies die gespeicherten Daten aus der CSV-Datei wieder ein.
4. Bonus: Daten plotten:
   4.1. Optional können Sie die gelesenen Daten plotten, um die Daten visuell darzustellen.

6. Praktische Übung: Programmierung eines einfachen Python-Datenrecorders

Übung 2: Erzeugung und Verarbeitung von synthetischen Daten mit mehreren Attributen

1. Komplexe synthetische Daten generieren:
   1.1. Schreibe ein Python-Skript, das Daten mit mehreren Attributen (z.B. Temperatur, Luftfeuchtigkeit und Luftdruck) generiert.
2. Daten speichern:
   2.1. Speichere die Daten in einer CSV-Datei.
3. Daten wieder einlesen:
   3.1. Lies die gespeicherten Daten wieder ein und berechne z.B. den Durchschnittswert für jedes Attribut.
4. Daten plotten:
   4.1. Visualisieren Sie die Daten, z.B. als Scatter Plot.

6. Praktische Übung: Programmierung eines einfachen Python-Datenrecorders

Übung 3: Arbeiten mit echten Daten

1. Schritt-für-Schritt-Anleitung:
   1.1. Datensatz auswählen und herunterladen:
      1.1.1. Besuche eine Plattform wie Kaggle oder das UCI Machine Learning Repository.
      1.1.2. Wähle einen Datensatz aus, der für Sie interessant ist. Beispiele könnten ein Wetterdatensatz, ein Datensatz zur Luftqualität oder ein Aktienmarkt-Datensatz sein.
      1.1.3. Lade den Datensatz herunter und speichere ihn auf Ihrem Computer.
2. Datensatz einlesen:
   2.1. Verwende Python, um den Datensatz zu laden und eine erste Inspektion der Daten durchzuführen. Lies die Daten aus einer CSV-Datei oder einem anderen Format ein.
3. Daten plotten:
   3.1. Visualisieren Sie die Daten, z.B. als Scatter Plot.