**Machine-Learning Bibliothek**

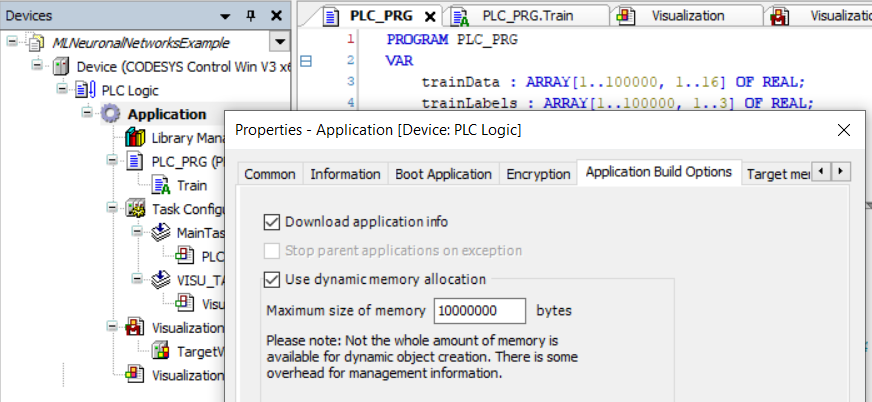
**Was?** Machine Learning stellt für viele Maschinenbauer ein interessantes Thema dar, dass aber auf Grund fehlender Infrastrukturen und zusätzlich nötiger Hardware oftmals nur schwer anzugehen ist. Diese Bibliothek zielt darauf ab, die grundlegenden Software-Bausteine die zum Machine Learning in künstlichen neuronalen Netzen (wie sie Google, etc. nutzen) nötig sind direkt in IEC anzubieten. Damit kann mit den Methoden des Machine Learning auf beliebigen Steuerungen experimentiert werden ohne zusätzlichen Einsatz von Infrastruktur oder Hardware.

**Wer?** Unabhängig davon, dass die vorliegende Bibliothek sehr einfach in IEC auf jeder Steuerung eingebunden werden kann, setzt sie dennoch einfache Grundlagen und Verständnis zum Machine Learning voraus. Eine Buchempfehlung die diese Grundlagen sehr schnell und effizient darstellt ist „Neuronale Netze selbst Programmieren“ von Tariq Rashid.

**Wie?** Zur einfacheren effizienteren Nutzung sind einige typische FB-Bausteine, die Anwendungen wiederspiegeln vorbereitet. Zudem sind ein einfacheres und ein komplexeres Beispiel der Anwendung der Bibliothek umgesetzt und mitgeliefert.

**Wieso?** Die gesamte Bibliothek ist Open-Source und darf und kann entsprechend erweitert, verbessert oder verändert werden. Gleichermaßen besteht kein Anspruch auf bestimmte Funktionalitäten oder Fehlerfreiheit.

***Wichtig: Die Bibliothek nutzt „Dynamic Memory“. Diese Option muss unter den Eigenschaften der Applikation aktiviert, bzw. mit Speicher versehen werden:***

******

***Beispiele:***

* ***MLNeuronalNetworkExample.project:*** *Die Idee ist es beliebige 16 Eingänge mit einem Ampelsystem (rot, gelb, grün) zu verknüpfen. Entsprechend wird der FB VariableNeuronalNetwork16To3 eingesetzt. Auf Knopfdruck werden Trainingsdaten generiert und das Netzwerk trainiert, das dann im Anschluss mit einer bestimmten Eingangskonfiguration angefragt werden kann.*
* ***MLNeuronalNetworkComplexExample.project:*** *Die Idee ist es Vibrationsdaten (als Amplitude-Zeit-Daten der Länge 2048) in einer Anomalieerkennung zu erlernen. Dazu wird der FB AnomalieDetection2048 eingesetzt, der so konfiguriert ist, dass er die Daten zunächst fourier transformiert. Jeden Zyklus werden Trainingsdaten des simulierten Gut-Zustandes erzeugt, mit denen das Netzwerk trainiert wird. Alle 10 Durchläufe werden Daten eines simulierten Schlecht-Zustandes erzeugt. Jeweils wird der Anomaliefaktor zum Vergleich, zusammen mit einer grafischen Ausgabe der erzeugten Spektren angezeigt.*

**1 – Anomalieerkennung**

**Idee (Unsupervised Learning):**Oftmals liefert eine Maschine Daten, die ihren Zustand charakterisieren. Im besten Falle sind das in den aller meisten Fällen Daten, die den „guten Zustand“ der Maschine kennzeichnen. Die Anomalieerkennung zielt darauf ab den guten Zustand sehr gut kennenzulernen, um dann, falls ein schlechter/abweichender Zustand eintritt, diesen durch einen gestiegenen Anomaliefaktor kennzeichnen zu können.

Die dafür vorbereiteten Funktionsbausteine besitzen den Namen „AnomalieDetection[…]“ wobei […] durch 64, 1024, etc. ersetzt wird. Die Nomenklatur impliziert durch die Zahl, wie groß das Eingangsdaten-Array ist, d.h. wieviel Einträge das Array besitzen muss mit dem Trainiert oder Angefragt wird. Bei der Anfrage liefern die FBs jeweils einen REAL-Wert als Anomalie-Faktor, der je größer eine umso „größere Anomalie“ kennzeichnet.

**Methoden:**

**Initialize**

Initialisierung eines Neuronalen Netzwerks zur Anomalieerkennung

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | learningrate | REAL | Lernrate des Netzwerks  (default = 0.4, Empfehlung: 0.3…0.5) |
| Eingang | doFFT | BOOL | TRUE: Eingangswerte werden Fourier-Transformiert |
| Eingang | doNormalize | BOOL | TRUE: Eingangswerte werden normalisiert |
| Eingang | doShuffleShift | BOOL | TRUE: Die Eingangswerte werden zyklisch, zufällig für das Training verschoben. |

**Train**

Trainieren des Neuronalen Netzwerks

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | trainData | ARRAY [1..uiSize] OF REAL |  |

Anmerkung: uiSize = Anzahl der Datenpunkte (z.B. bei AnomalieDetection2048 → uiSize = 2048)

**Query**

Abfrage des Neuronalen Netzwerks zur Erkennung von Anomalie

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | queryData | ARRAY [1..uiSize] OF REAL |  |

Anmerkung: uiSize = Anzahl der Datenpunkte (z.B. bei AnomalieDetection2048 → uiSize = 2048)

**File\_Save\_Weights**

Speichern der Konfiguration eines Neuronalen Netzwerks zu einem bestimmten Zeitpunkt

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | Filename | STRING(255) | Dateinamen der aktuellen Konfiguration |

**File\_Load\_Weights**

Laden einer zuvor abgespeicherten Netzwerk-Konfiguration

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | Filename | STRING(255) | Dateinamen der zu ladenden Konfiguration |

**File\_GetFileList**

Rückgabe einer Liste mit allen gespeicherten Konfigurations-Dateien

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | List | POINTER TO STRING(255) | STRING-Array, in das die Dateiliste geschrieben wird |

**File\_Delete\_Weights**

Löschen einer bestimmten Konfigurations-Datei

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | Filename | STRING(255) | Dateinamen der zu löschenden Konfiguration |

**File\_Delete\_WeightsALL**

Löschen aller Konfigurationsdateien

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | Filename | STRING(255) | Dateinamen der zu ladenden Konfiguration |

**2 – Variables Neuronales Netzwerk**

**Idee (Supervised Learning):** In vielen Situationen erkennt man an Hand der Konstellation von Sensorwerten z.B. ob ein Bauteil gut oder schlecht ist. Diese Klassifikation findet von Hand oder in groben Grenzen statt. Ein „klassisches Neuronales Netz“ kann dazu genutzt werden, Daten z.B. zu klassifizieren. In diesem Fall werden beliebige Eingangswerte bestimmten vorgegebenen Ausgangswerten zugeordnet und das Netzwerk damit trainiert. Ziel ist es für unbekannte Ausgangswerte eine Abschätzung für die Ausgangswerte zu erlernen.

Die dafür vorbereiteten Funktionsbausteine besitzen den Namen „VariableNeuronalNetwork[…]“ wobei […] durch 1024To10, 16To3, etc. ersetzt wird. Die Nomenklatur impliziert durch die Zahlen, wie groß das Eingangsdaten-Array ist (z.B. 16) und auf ein wie langes Ausgangsdaten-Array (z.B. 3) abgebildet wird. Bei der Anfrage liefern die FBs jeweils ein Ausgangsdaten-Array zurück, das je nach Anwendungsfall interpretiert wird.

**Methoden:**

**Initialize**

Initialisierung eines variablen Neuronalen Netzwerks

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | learningrate | REAL | Lernrate des Netzwerks  (default = 0.4, Empfehlung: 0.3…0.5) |
| Eingang | hiddenLayers | POINTER TO UINT | Adresse des Arrays mit der Knotenanzahl des jeweiligen Layers |
| Eingang | hiddenTypes | POINTER TO UINT | Adresse des Arrays mit der Typkennzahl des jeweiligen Layers |
| Eingang | hiddenLayersCount | UINT | Anzahl der Hidden Layer (ohne In- und Outlayer) |

**Train**

Trainieren des variablen Neuronalen Netzwerks

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | InputTrain | POINTER TO REAL | Adresse des Arrays mit den Train-Inputs |
| Eingang | TargetsTrain | POINTER TO REAL | Adresse des Arrays mit den Train-Targets |

**Query**

Abfragen des variablen Neuronalen Netzwerks

|  |  |  |  |
| --- | --- | --- | --- |
| **Gültigkeitsbereich** | **Name** | **Typ** | **Beschreibung** |
| Eingang | InputQuery | POINTER TO REAL | Adresse des Arrays mit den Query-Inputs |
| Eingang | OutputQuery | POINTER TO REAL | Adresse eines leeren Arrays mit den Query-Outputs |

Weitere Methoden zum Speichern und Laden von Netzwerkkonfigurationen:

* **File\_Save\_Weights**
* **File\_Load\_Weights**
* **File\_Get\_FileList**
* **File\_Delete\_Weights**
* **File\_Delete\_WeightsALL**

Beschreibung dazu siehe oben bei „1-Anomalieerkennung“.