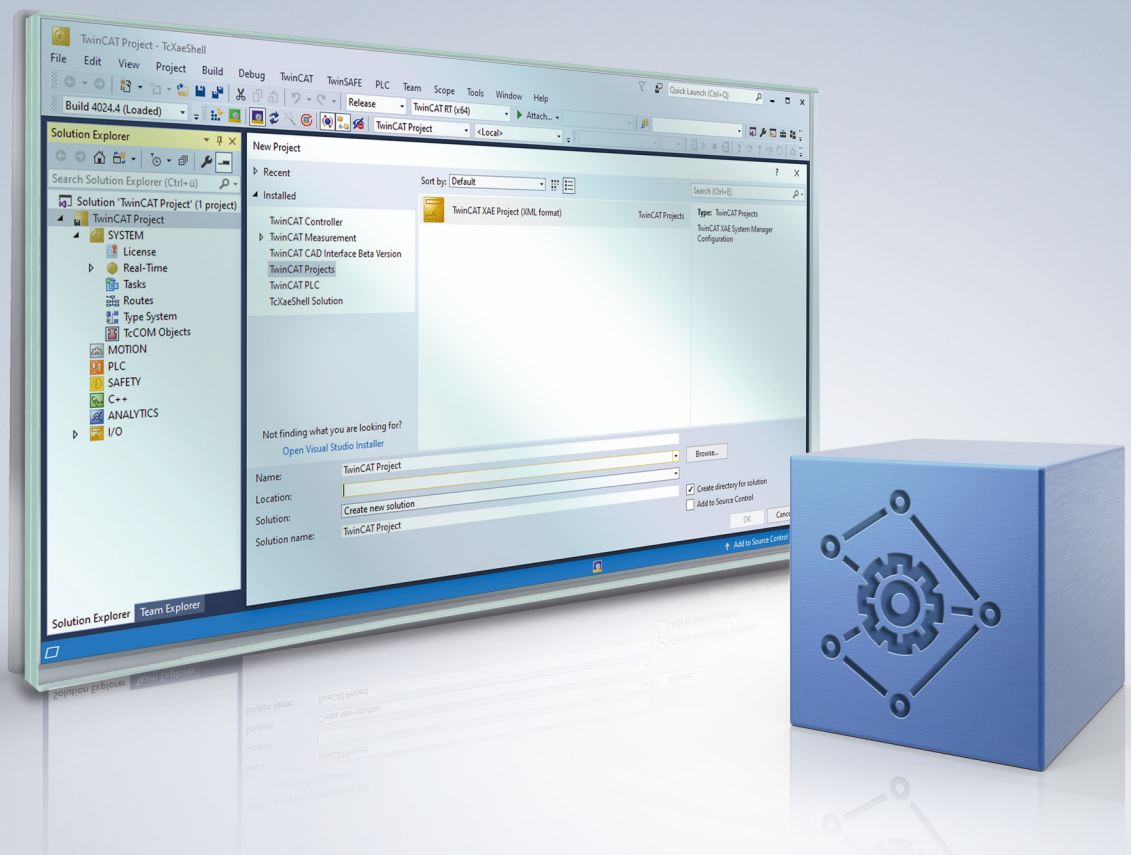
BECKHOFF New Automation Technology

Handbuch | DE

TF3820

TwinCAT 3 | Machine Learning Server



21.11.2024 | Version: 1.0.2

Inhaltsverzeichnis

1 Vorwort ... 5
1.1 Hinweise zur Dokumentation ... 5
1.2 Zu Ihrer Sicherheit ... 6
1.3 Hinweise zur Informationssicherheit ... 7
1.4 Ausgabestände dieser Dokumentation ... 8
2 Übersicht ... 9
3 Installation ... 11
3.1 Lizenzierung ... 12
3.2 Einrichtung einer NVIDIA Grafikkarte ... 14
4 Quickstart ... 16
5 Technische Einführung ... 20
5.1 Workflow ... 20
5.1.1 ONNX für die Verwendung mit TwinCAT Machine Learning Server vorbereiten ... 20
5.1.2 Modellbeschreibungsdateien auf Server-Device verfügbar machen ... 23
5.1.3 Server aus SPS-Client heraus konfigurieren ... 24
5.1.4 KI-Modell ausführen ... 25
5.1.5 KI-Modell updaten ... 27
5.2 TwinCAT Machine Learning Model Manager ... 28
5.2.1 Graphisches User Interface ... 28
5.2.2 Python Interface ... 31
5.2.3 Command Line Interface ... 31
5.3 ONNX Support ... 32
5.4 TcMlServer Service ... 34
5.4.1 Execution Provider ... 34
6 API ... 36
6.1 Funktionsbausteine ... 36
6.1.1 FB_MlSvrPrediction ... 36
6.2 Datentypen ... 41
6.2.1 E_ExecutionProvider ... 41
6.2.2 ST_PredictionParameter ... 42
7 Beispiele ... 43
7.1 KI-basierte Bildverarbeitung ... 43
7.2 Custom Attributes ... 45
8 Anhang ... 49
8.1 Error Codes ... 49
8.2 Log files ... 52
8.3 Third-party components ... 52
8.4 Support und Service ... 53

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

1.4 Ausgabestände dieser Dokumentation

Version	Änderungen
1.0.2	In den Error Codes [▶ 49] wurden die Hexadezimalwerte ergänzt.
1.0.x	
1.0.0	Erste Veröffentlichung

2 Übersicht

Einführung

Der TwinCAT Machine Learning Server ermöglicht die Ausführung von KI-Modellen direkt auf dem Steuerungs-IPC oder auf einem Edge-Gerät.

Der TwinCAT Machine Learning Server besteht aus zwei Komponenten:

- Dem SPS-Funktionsbaustein als Client des Machine Learning Servers.
- Dem Machine Learning Server als Bereitsteller von Diensten (Laden, Ausführen, … von KI-Modellen).

Diese Komponenten bieten eine asynchrone Ausführungsfunktionalität für SPS-Programme. Das Konzept der asynchronen Berechnung entkoppelt die KI-Modell-Ausführungszeit effektiv vom zyklischen Betrieb der SPS. Der Machine Learning Server ermöglicht die Ausführung beliebig komplexer KI-Modelle sowohl auf CPUs als auch auf NVIDIA GPUs und ist damit insbesondere für die Verwendung mit dem Industrie-PC C6043 geeignet. Der Machine Learning Server wird im Usermode des Betriebssystems ausgeführt. Das führt zu nicht-deterministischem Verhalten, das nur teilweise durch entsprechende Konfiguration der Usermode-Komponenten gemindert werden kann.

Der TwinCAT Machine Learning Server lädt als ONNX-Datei bereitgestellte KI-Modelle. Alle relevanten KI-Frameworks, wie Tensor Flow, Pytorch, Scikit Learn usw. unterstützen diesen Interoperabilitätsstandard. Daraus resultiert eine Entkopplung der Trainingsumgebung und der Ausführungsumgebung. Es können beliebige Trainingsumgebungen genutzt werden, um KI-Modelle zu erstellen und diese können dann mit dem TwinCAT Machine Learning Server zur Ausführung gebracht werden.

Zielgruppen und Anwendungsfälle

Der Machine Learning Server richtet sich unter anderem an die folgenden Anwendungsfälle:

- Einsatz von rechenintensiven KI-Modellen, bei denen die erwartete Rechenzeitreduktion durch Beschleunigung auf einer GPU die zu erwartenden Rechenzeitschwankungen (Jitter) überkompensiert.
  - Insbesondere sind hier beispielhaft Vision-KI-Modelle zu nennen für Bildklassifikation, Objekterkennung oder Segmentierung.
- Einsatz von KI-Modellen in niederpriorisierten Tasks, die nur lose mit dem deterministischen SPS-Programm gekoppelt sind.
  - KI-Modelle, deren Ergebnisse nicht von der Steuerung weiterverwendet werden, sondern an Systeme oberhalb der Steuerungsebene kommuniziert werden. Zum Beispiel Prozessanalyse-Modelle, bei denen der Maschinenführer informiert wird, Prädiktive Maintenance-Modelle, bei denen das Service Personal informiert wird, usw.
  - KI-Modelle, deren Ergebnisse nicht zu einem bestimmten Zeitpunkt von der Steuerung benötigt werden. Beispielsweise KI-Modelle zur Bereitstellung von optimierten oder angepassten Prozessparametern.

Abgrenzung und Vergleich zu ähnlichen TwinCAT-Produkten

Neben dem TwinCAT Machine Learning Server existieren weitere TwinCAT-Produkte mit ähnlicher Funktionalität, also dem Ausführen von KI-Modellen.

- TF3800 TwinCAT Machine Learning Inference Engine
- TF3810 TwinCAT Neural Network Inference Engine
- TF7800 TwinCAT Vision Machine Learning
- TF7810 TwinCAT Vision Neural Network

Die wesentlichen Unterschiede zwischen den aufgeführten Produkten und dem TwinCAT Machine Learning Server sind in der folgenden Tabelle aufgeführt.

Tab. 1: Produkteigenschaften im Vergleich

Deterministische KI: TF3800, TF3810, TF7810	Beschleunigte KI: TF3820
Deterministische KI-Ausführung im TwinCAT-Prozess	Nahe-Echtzeitausführung im separaten Prozess
Ausführung auf standard x64 CPUs	Hardwarebeschleunigung auf NVIDIA GPUs möglich
Unterstützt ausgewählte KI-Modelle und Operatoren	Unterstützt aktuelle ONNX Opset Version und damit aktuelle und vielfältige KI-Modelle
Standard SPS-Funktionsbaustein zur einfachen Integration in TwinCAT	Standard SPS-Funktionsbaustein zur einfachen Integration in TwinCAT
Interoperabilität durch ONNX-Support	Interoperabilität durch ONNX-Support
Lizenz-Bundle: TF3810 beinhaltet TF3800, TF7800 und TF7810	Auch als Server im Netzwerk nutzbar mit mehreren Clients

3 Installation

Um den TwinCAT Machine Learning Server nutzen zu können, benötigen Sie drei Komponenten:

- **TF3820 | TwinCAT Machine Learning Server**
  Installieren und lizenzieren Sie diesen Workload entweder direkt auf Ihrem Steuerungs-IPC oder auf einem Edge-IPC.
  - Installiert den Service TcMlServer [▶ 34] auf dem System.
- **TF3830 | TwinCAT Machine Learning Server Client**
  Installieren Sie diesen Workload auf Ihrem Engineering-PC, um die notwenige SPS-Bibliothek nutzen zu können.
  - Installiert die SPS-Bibliothek Tc3_MlServer [▶ 36] für das TwinCAT 3 XAE.
- **TF38xx | TwinCAT Machine Learning Model Manager**
  Installieren Sie diesen Workload auf Ihrem Engineering-PC, um die ONNX-Datei für die Nutzung in TwinCAT vorbereiten [▶ 20] zu können.

Systemvoraussetzungen: TwinCAT Machine Learning Server (TF3820)

Technische Daten	Voraussetzungen
Betriebssystem	Windows10
Zielplattform	x64
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4026
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAR
Erforderliche TwinCAT-Lizenz	TC1000

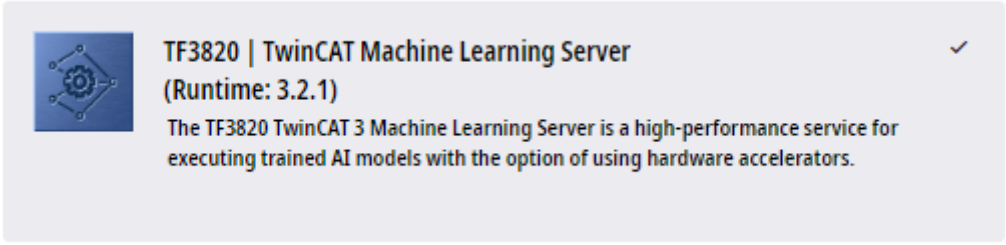
Systemvoraussetzungen: TwinCAT Machine Learning Server Client (TF3830)

Technische Daten	Voraussetzungen
Betriebssystem	Windows10
Zielplattform	x64
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4026
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAE
Erforderliche TwinCAT-Lizenz	TC1200

TwinCAT Package Manager: Installation (TwinCAT 3.1 Build 4026)

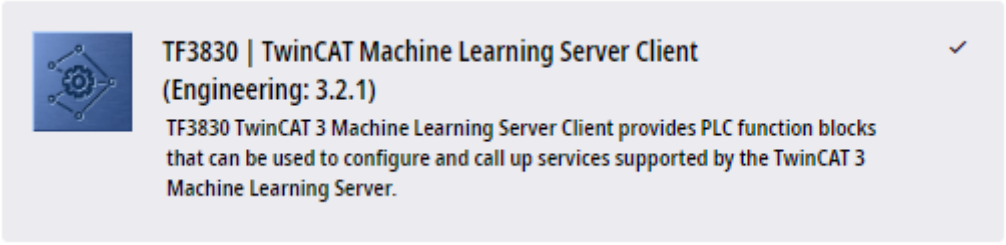
Eine ausführliche Anleitung zur Installation von Produkten finden Sie im Kapitel Workloads installieren in der Installationsanleitung TwinCAT 3.1 Build 4026.

Installieren Sie den folgenden Workload, um das Produkt nutzen zu können:



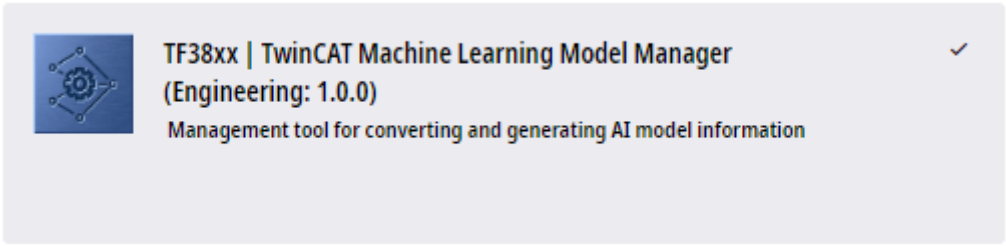
TwinCAT Package Manager UI: TF3820 | TwinCAT 3 Machine Learning Server

TwinCAT Package Manager CLI:
```
tcpkg install TF3820.MachineLearningServer.XAR
```



TwinCAT Package Manager UI: TF3830 | TwinCAT 3 Machine Learning Server Client

TwinCAT Package Manager CLI:
```
tcpkg install TF3830.MachineLearningServerClient.XAE
```



TwinCAT Package Manager UI: TF38xx | TwinCAT 3 Machine Learning Model Manager

TwinCAT Package Manager CLI:
```
tcpkg install TF38xx.MachineLearningModelManager.XAE
```

Installation auf Systemen mit TF3800 oder TF3810 Version 3.2.6 und jünger

Potenzieller Konflikt mit alten Installationen

Es kann zu Konflikten bezüglich der Pakete für den TwinCAT Machine Learning Model Manager kommen. Ein Uninstall löst diesen Konflikt schnell auf.

Sollten Sie eine Version des Workloads TF3800.MachineLearningInferenceEngine.XAE oder TF3810.NeuralNetworkInferenceEngine.XAE in der Version 3.2.6 oder kleiner installiert haben, sollten Sie zuvor diese Workloads deinstallieren:

```
tcpkg uninstall TF3800.MachineLearningInferenceEngine.XAE --include-dependencies
tcpkg uninstall TF3810.NeuralNetworkInferenceEngine.XAE --include-dependencies
```

Erst danach sollten Sie die Workloads in einer Version 3.2.10 oder höher installieren.

3.1 Lizenzierung

Die **Lizenz TF3820** TwinCAT Machine Learning Server wird von der Serverkomponente (Service TcMLServer) abgefragt. Die Lizenz ist entsprechend auf dem IPC abzulegen, auf der der Server-Prozess ausgeführt wird. Die Lizenz TF3820 beinhaltet die Lizenz TF3830, sodass eine lokale Kommunikation zwischen Funktionsbaustein und Server ohne weitere Lizenzen möglich ist.

Die **Lizenz TF3830** TwinCAT Machine Learning Server Client wird vom Funktionsbaustein der SPS-Bibliothek Tc3_MlServer abgefragt. Die Lizenz wird auf Runtime-Systemen benötigt, die remote auf den TwinCAT Machine Learning Server zugreifen. Bei lokaler Kommunikation zwischen Funktionsbaustein und Server ist keine separate TF3830-Lizenz erforderlich.

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

Zyklische Lizenzabfrage des TwinCAT Machine Learning Servers

Wenn keine oder nur eine 7-Tage-Testlizenz für den TwinCAT Machine Learning Server vorhanden ist, wird zyklisch alle 10 Sekunden eine Lizenzabfrage durchgeführt.

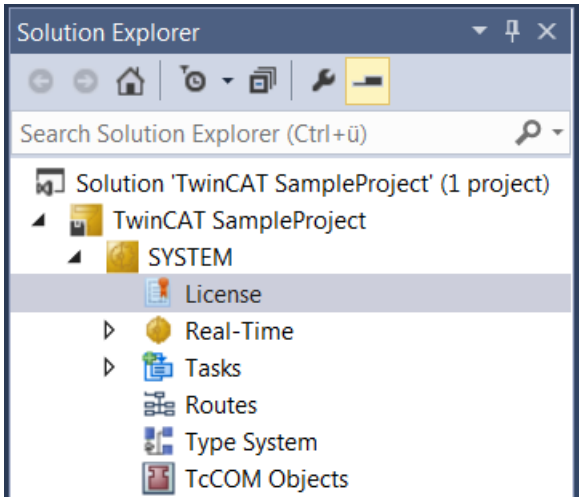
Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT-3-Lizenzierung“.

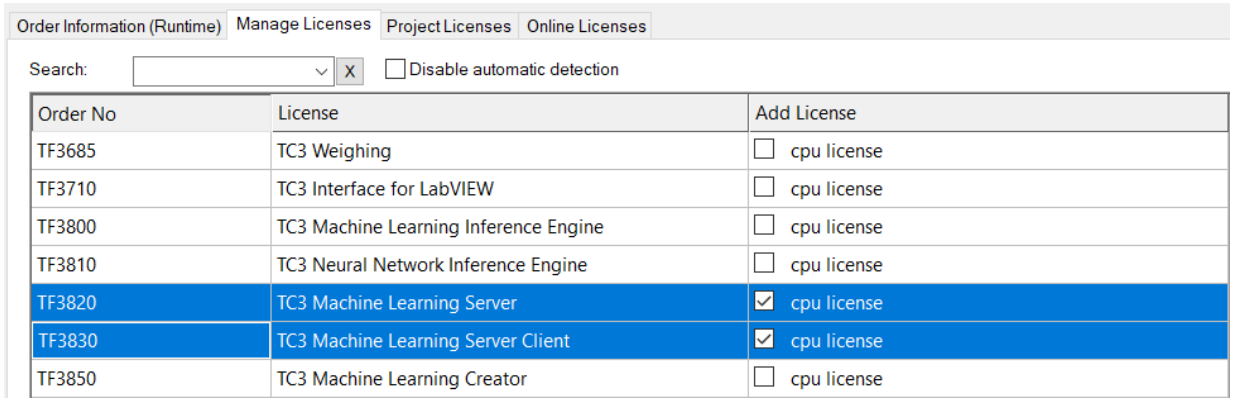
Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.
4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**



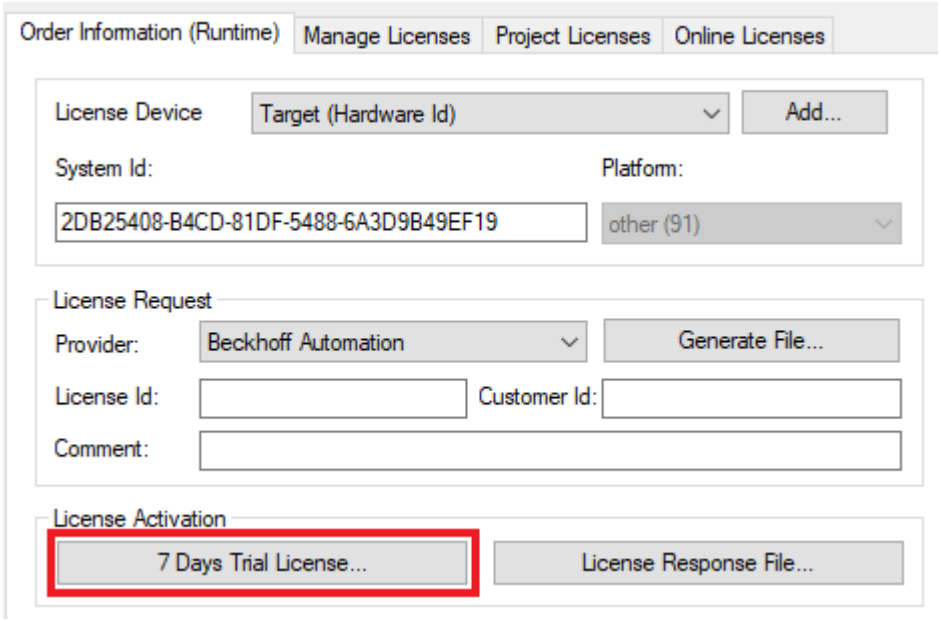
   ⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.
5. Öffnen Sie die Registerkarte **Manage Licenses.**
6. Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (in diesem Fall „TF3820 TC3 Machine Learning Server“ und/oder „TF3830 TC3 Machine Learning Server Client“).
   Beachten Sie, dass TF3820 die Lizenz TF3830 für das lokale System inkludiert. Ist eine TF3820 auf dem lokalen System, benötigen Sie nicht zusätzlich die TF3830 für den Client. TF3830 wird nur auf remote Clients benötigt.



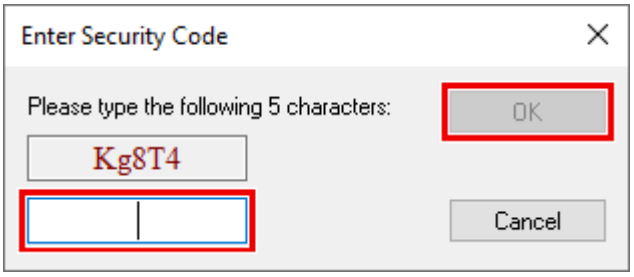
7. Öffnen Sie die Registerkarte **Order Information (Runtime).**

⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt.

8. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



9. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
10. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
    ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
11. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

3.2 Einrichtung einer NVIDIA Grafikkarte

Wenn für die GPU-beschleunigte KI-Modellausführung **keine** Beckhoff-Hardware mit zugehörigem Beckhoff Image verwendet wird, ist der Kunde selbstständig dafür verantwortlich, auf seinem System die notwendigen Rahmenbedingungen für einen Betrieb der Grafikkarte herzustellen.

Sollten Sie einen Beckhoff IPC mit GPU und zugehörigem Beckhoff Image verwenden, können Sie diesen Abschnitt überspringen.

Systemvoraussetzungen

Die Systemvoraussetzungen betreffen insbesondere die erforderlichen Komponenten von NVIDIA. Die im folgenden aufgeführten Software-Versionen sollten nicht aktueller sein als die angegeben Versionen (z. B. nicht CUDA 12.6 installieren), da es ansonsten zu Inkompatibilitäten kommt.

NVIDIA-Treiber

- Normale NVIDIA GPU: Version 560.67
- Quadro/RTX GPU: Version 522.86

CUDA (Compute Unified Device Architecture)

- CUDA Version12.5.1

cuDNN

- cuDNN Version 9.2.1.18 (als zip-Datei, nicht als Windows Installer nutzen)
- Instruktionen zur Installation finden Sie bei NVIDIA: Tarball Installation.
- Es ist notwendig, den Pfad zu den cuDNN-Bibliotheken zu der PATH-Variable des Systems, nicht der des Users, hinzuzufügen.

In der Regel werden durch die obigen Maßnahmen die folgenden Voraussetzungen auf dem System erfüllt:

- Verfügbarkeit der cuda.dll Bibliothek
- Verfügbarkeit der nvml.dll Bibliothek

Optimiertes Laufzeitverhalten

Bei Verwendung von Beckhoff-Hardware, z. B. Industrie-PC C6043 mit GPU und zugehörigem Beckhoff Image, ist das Laufzeitverhalten der NVIDIA GPU optimal für das Zusammenspiel mit der TwinCAT-Echtzeit ausgelegt. Bei Verwendung von Drittanbieter-GPUs können, abhängig von der konkreten GPU und dessen Einstellungen, erhebliche Laufzeitschwankungen bei der Modellausführung auftreten.

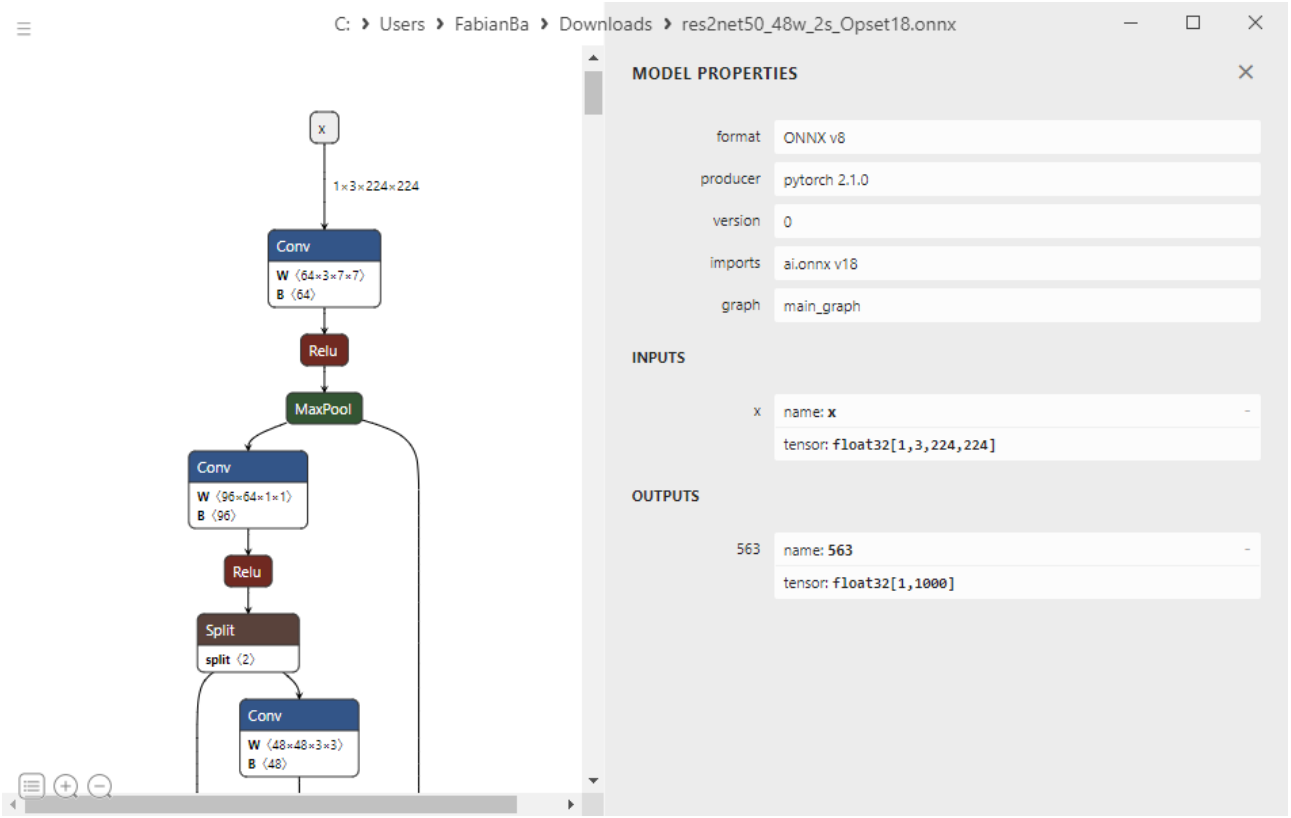
Der TwinCAT Machine Learning Server informiert direkt nach dem Starten des Servers im Log-File [▶ 52] über mögliche Herausforderungen der verwendeten Grafikkarte.

4 Quickstart

ONNX-Datei erstellen oder herunterladen

Sollten Sie keine eigene ONNX für einen ersten Test zur Hand haben, können Sie sich beispielsweise bei GitHub im ONNX Model Zoo für Tests bedienen. Im Folgenden wird beispielhaft das ResNet50 aus dem ONNX Model Zoo verwendet.

Mit Netron kann einfach inspiziert werden, ob die Voraussetzungen [▶ 32] für die Ausführung mit dem TwinCAT Machine Learning Server erfüllt sind.



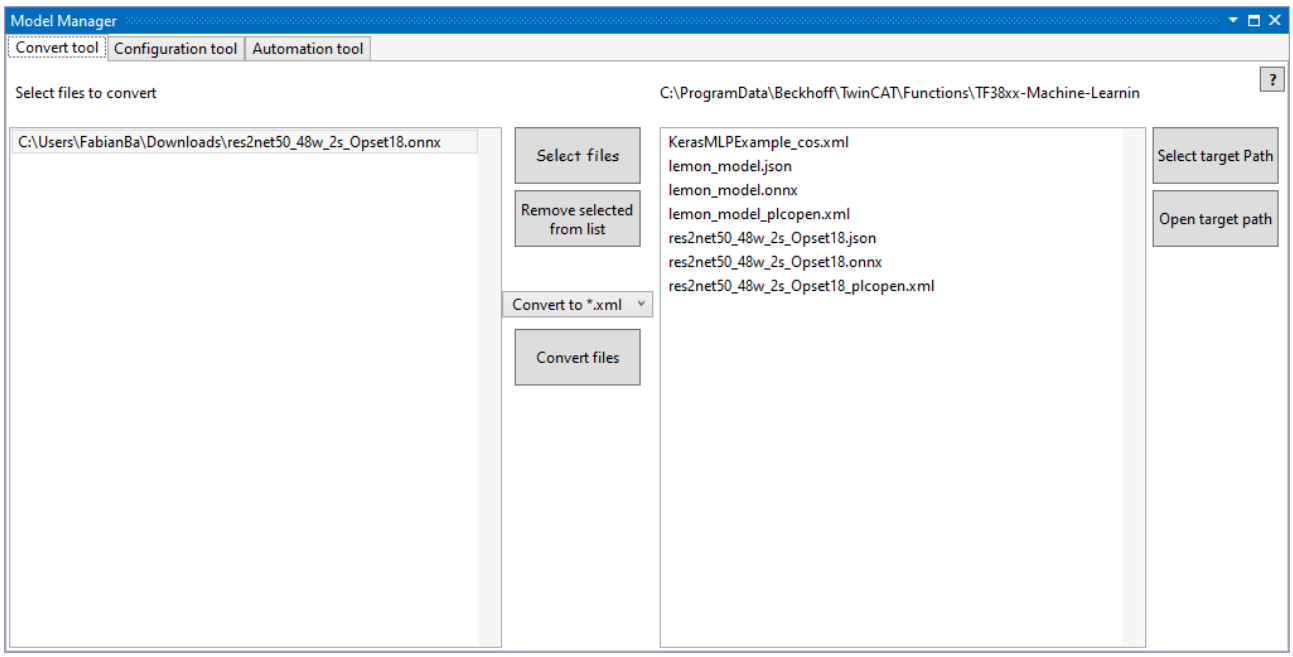
Die Input Nodes sind nicht dynamisch und der verwendete ONNX Opset wird ebenfalls unterstützt.

ONNX-Datei mit TwinCAT Machine Learning Model Manager aufbereiten

Öffnen Sie TwinCAT XAE und navigieren Sie zu TwinCAT > Machine Learning > Machine Learning Model Manager [▶ 28].

Laden Sie mit „Select file“ das heruntergeladene ONNX und wählen Sie dann „Convert files“. Im Target Path werden Ihnen nun die ONNX sowie die erstellten zugehörigen JSON und PlcOpenXml angezeigt.

Wählen Sie „Open target path“ um den File Explorer auf diesem Pfad zu öffnen.



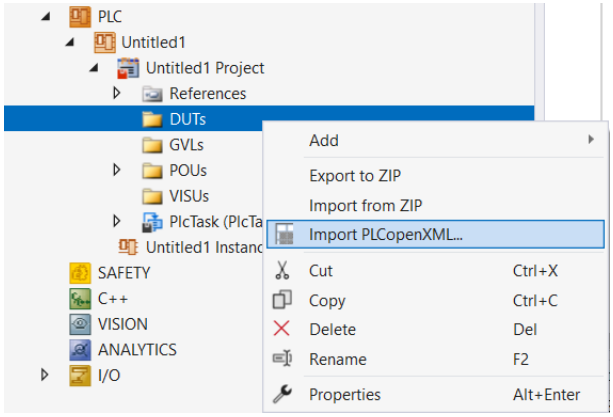
Dateien auf dem Zielsystem verfügbar machen

In diesem Quickstart wird davon ausgegangen, dass der TwinCAT Machine Learning Server auf demselben Gerät betrieben wird wie die SPS. Entsprechend werden die Modelldateien (res2net50_48w_2s_Opset18.onnx und res2net50_48w_2s_Opset18.json) auf dem Zielgerät unter dem Pfad C:\models abgelegt.

Mehr Informationen zu diesem Schritt finden Sie hier: Modellbeschreibungsdateien auf Server-Device verfügbar machen [▶ 23].

Quellcode schreiben

Es wird begonnen mit einem leeren SPS-Projekt. Zunächst importieren Sie die erstelle res2net50_48w_2s_Opset18_plcopen.xml durch Rechtsklick auf den Ordner DUTs und wählen dort „Import PLCopenXML“.



Fügen Sie außerdem unter References die SPS-Bibliothek Tc3_MlServer hinzu.

Der Code besteht im Minimalbeispiel aus zwei Schritten. Zunächst wird eine Session auf dem TwinCAT Machine Learning Server erstellt und dann die Inferenz des geladenen Modells ausgeführt.

Deklaration

```
stModelInput : ST_res2net50_48w_2s_Opset18Input;
stModelOutput : ST_res2net50_48w_2s_Opset18Output;

fbMlSvr     : FB_MlSvrPrediction;
bConfigured : BOOL := FALSE;
bError      : BOOL := FALSE;
```

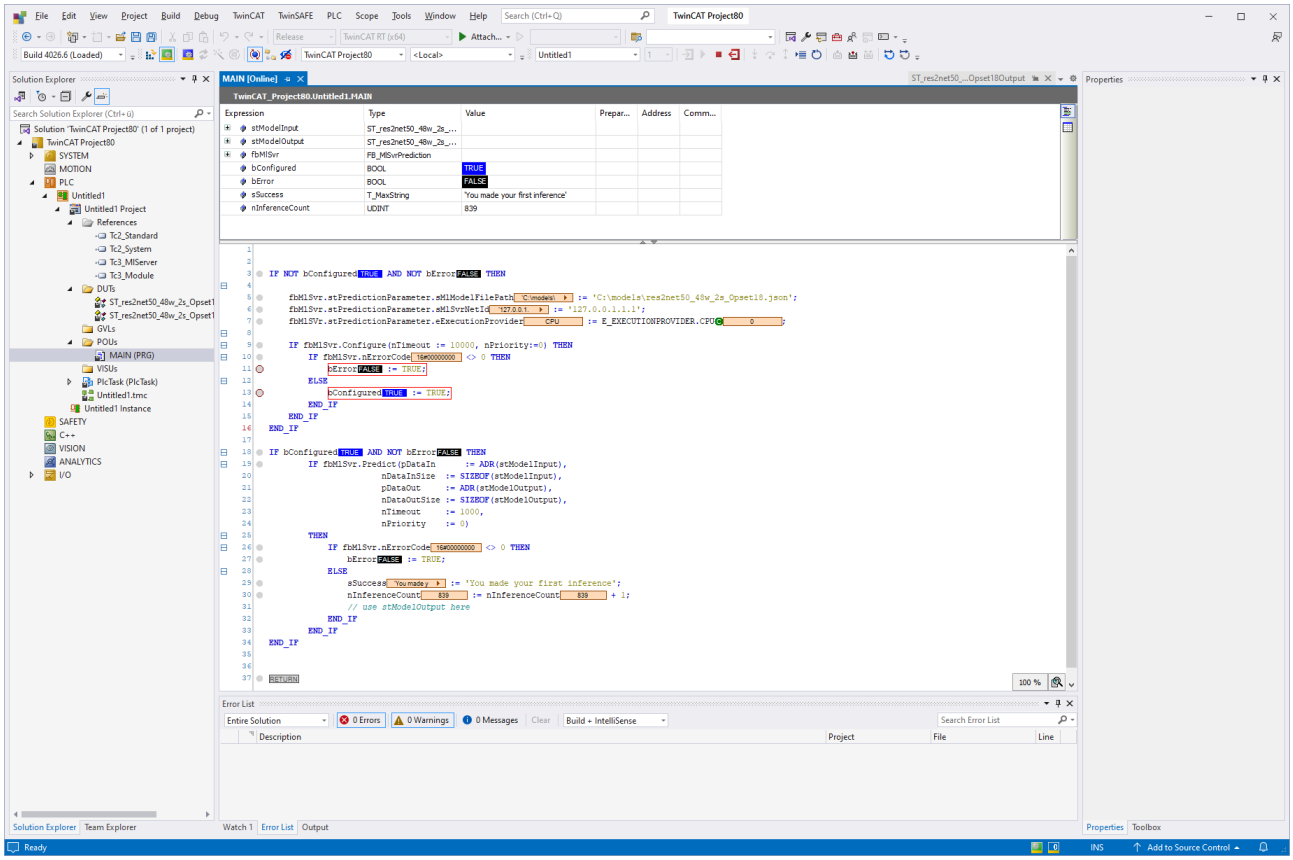
```
sSuccess        : T_MaxString;
nInferenceCount : UDINT := 0;
```

Code

```
IF NOT bConfigured AND NOT bError THEN

   fbMlSvr.stPredictionParameter.sMlModelFilePath := 'C:\models\res2net50_48w_2s_Opset18.json';
   fbMlSvr.stPredictionParameter.sMlSvrNetId := '127.0.0.1.1.1';
   fbMlSVr.stPredictionParameter.eExecutionProvider := E_ExecutionProvider.CPU;

   IF fbMlSvr.Configure(nTimeout := 10000, nPriority:=0) THEN
      IF fbMlSvr.nErrorCode <> 0 THEN
         bError := TRUE;
      ELSE
         bConfigured := TRUE;
      END_IF
   END_IF
END_IF

IF bConfigured AND NOT bError THEN
      IF fbMlSvr.Predict(
                  pDataIn      := ADR(stModelInput),
                  nDataInSize  := SIZEOF(stModelInput),
                  pDataOut     := ADR(stModelOutput),
                  nDataOutSize := SIZEOF(stModelOutput),
                  nTimeout     := 1000,
                  nPriority    := 0)
      THEN
         IF fbMlSvr.nErrorCode <> 0 THEN
            bError := TRUE;
         ELSE
            sSuccess := 'You made your first inference';
            nInferenceCount := nInferenceCount + 1;
            // use stModelOutput here
         END_IF
      END_IF
END_IF
```

Konfiguration aktivieren

Aktivieren Sie Ihre Konfiguration und starten Sie die SPS. Das Ergebnis wird unten dargestellt. Der Counter-Wert `nInferenceCount` steigt und die Variable `sSuccess` zeigt eine Erfolgsmeldung an.


TwinCAT_Project80.Untitled1.MAIN
Expression
Type
Value
stModelInput
stModelOutput
fbMlSvr
bConfigured
bError
sSuccess
nInferenceCount
IF NOT bConfigured THEN
fbMlSvr.Configure(nTimeout := 10000, nPriority:=0) THEN
IF bConfigured AND NOT bError THEN
fbMlSvr.Predict(pDataIn := ADR(stModelInput),
sSuccess := 'You made your first inference';
// use stModelOutput here
Error List
Ready

5 Technische Einführung

5.1 Workflow

Der Workflow besteht aus den folgenden Schritten:

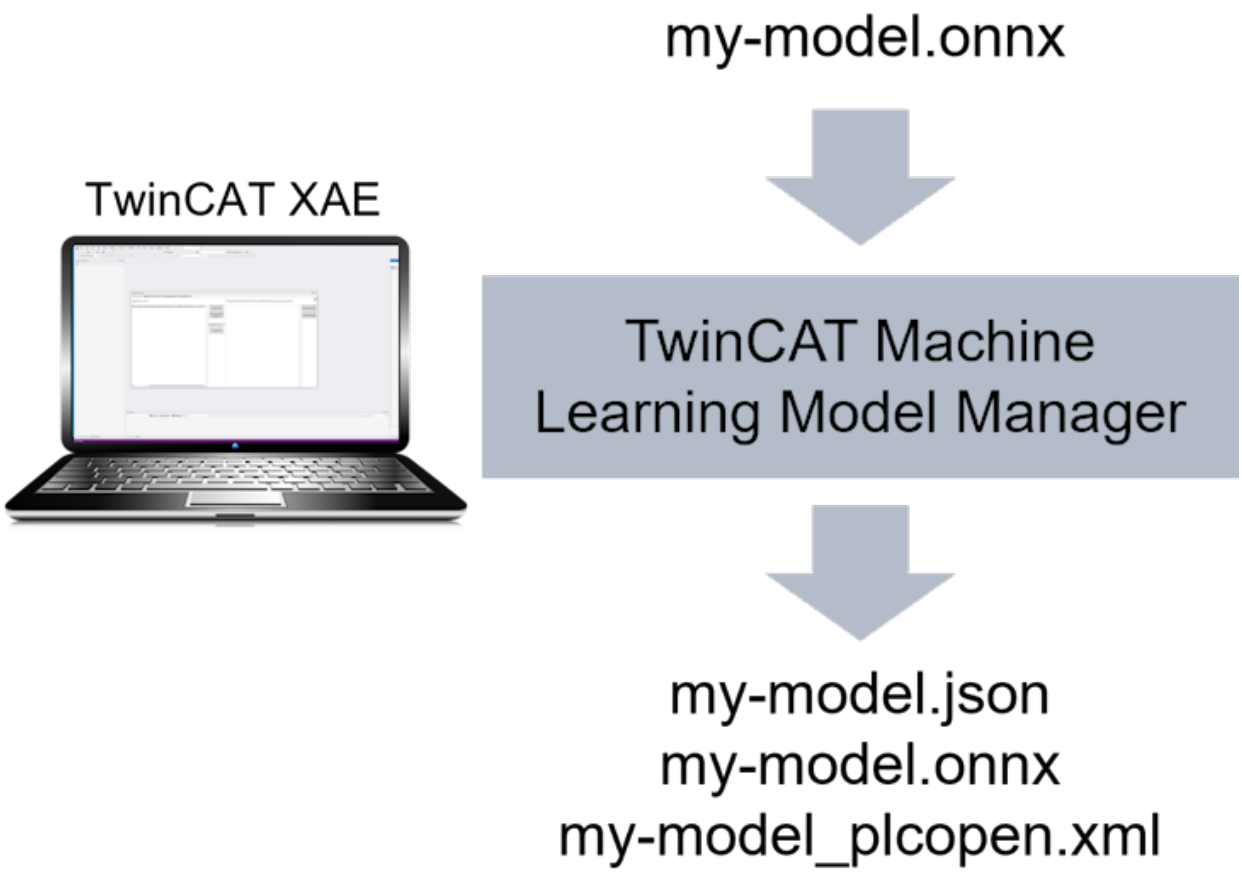
- SPS-Interfacebeschreibung [▶ 20] aus dem ONNX-File extrahieren (mit dem TwinCAT Machine Learning Model Manager [▶ 28]).
  - Aus dem ONNX-File wird eine PlcOpenXml erstellt, welche den Eingangs- und Ausgangsdatentyp des Modells für die SPS bereitstellt.
  - Ein JSON-File wird erzeugt, welches Meta-Informationen über das Modell bereithält. Meta-Informationen sind einerseits TwinCAT-spezifisch, andererseits Anwendungsspezifisch vom Nutzer.
- JSON-File und ONNX-File auf dem Target System bereitstellen [▶ 23].
- PlcOpenXml im TwinCAT Engineering importieren [▶ 20] und mit dem FB_MlSvrPrediction [▶ 36] in das SPS-Programm einarbeiten.
- Den TwinCAT Machine Learning Server aus der SPS heraus konfigurieren [▶ 24] und das KI-Modell laden.
- Das geladene KI-Modell asynchron zum SPS-Task-Zyklus aufrufen [▶ 25].
- Ein KI-Modell zur Laufzeit der Maschine austauschen/updaten [▶ 27].

5.1.1 ONNX für die Verwendung mit TwinCAT Machine Learning Server vorbereiten

Interfacebeschreibungen für die SPS erzeugen

Um eine ONNX mit dem FB_MlSvrPrediction [▶ 36] in der TwinCAT SPS nutzen zu können, werden Interfaceinformationen benötigt. Diese werden durch den TwinCAT Machine Learning Model Manager [▶ 28] erzeugt.

Angaben zum unterstützten ONNX Opset und Einschränkungen finden Sie hier: ONNX Support [▶ 32].



JSON-File

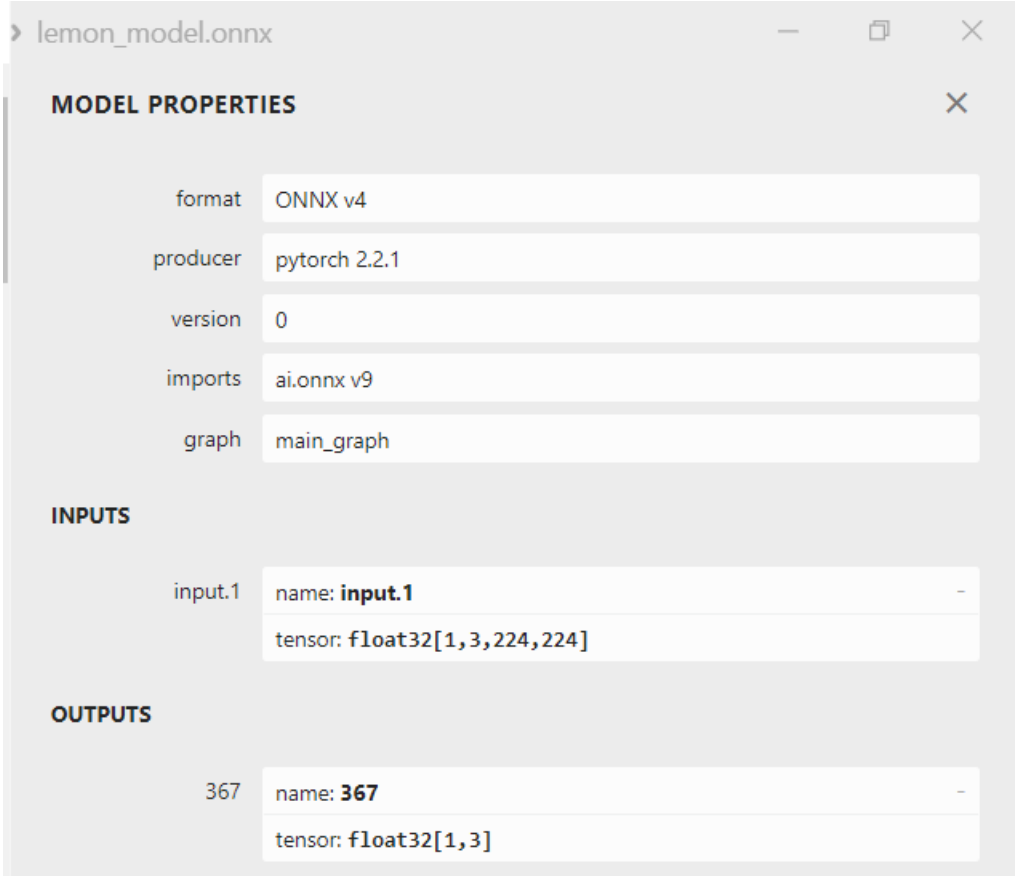
Das JSON-File beinhaltet Meta-Informationen, welche der Funktionsbaustein `FB_MlSvrPrediction` benötigt. Darüber hinaus können Nutzer dem JSON-File eigene Meta-Informationen über die Custom-Attributes [▶ 45] mitgeben. Das JSON-File wird vom Funktionsbaustein geladen, siehe Configure-Methode [▶ 37]. Sowohl das JSON-File als auch die ONNX-Datei müssen auf dem Laufzeit-PC zum Laden verfügbar sein, siehe Modellbeschreibungsdateien auf Server-Device verfügbar machen [▶ 23].

PlcOpenXml

Das PlcOpenXml beinhaltet die SPS-Typbeschreibung der Input- und Output Nodes. Die automatisch generierten Input-/Output-Strukturen (DUTs) spiegeln die Input-/Output-Definition der von Ihnen bereitgestellten ONNX-Datei wider. Stellen Sie daher sicher, dass Sie *aussagekräftige Namen* für die Input- und Output Nodes in ihrer ONNX verwenden.

Die Nutzung dieser erstellen Datentypen ist streng erforderlich.

Beispiel zur Namensgenerierung in der PlcOpenXml:



Dateiname: **lemon_model**.onnx

Name Input node: **input.1**

Name Output node: **367**

Es werden zwei DUTs vom Typ STRUCT erzeugt mit den Namen ST_**lemon_model**Input und ST_**lemon_model**Output. Jedes STRUCT hat einen header (nicht verändern!) und einen Datenbereich. Die Datenbereich-Elemente werden entsprechend der Input- und Output Nodes benannt, im obigen Fall in in_**input1** und out_**367**. In der SPS nicht gestattete Zeichen werden automatisch entfernt. Besitzt ein KI-Modell mehrere Input- oder Output Nodes, werden diese jeweils als Element des STRUCT dargestellt.

Beispiel für den generierten Inputdatentypen

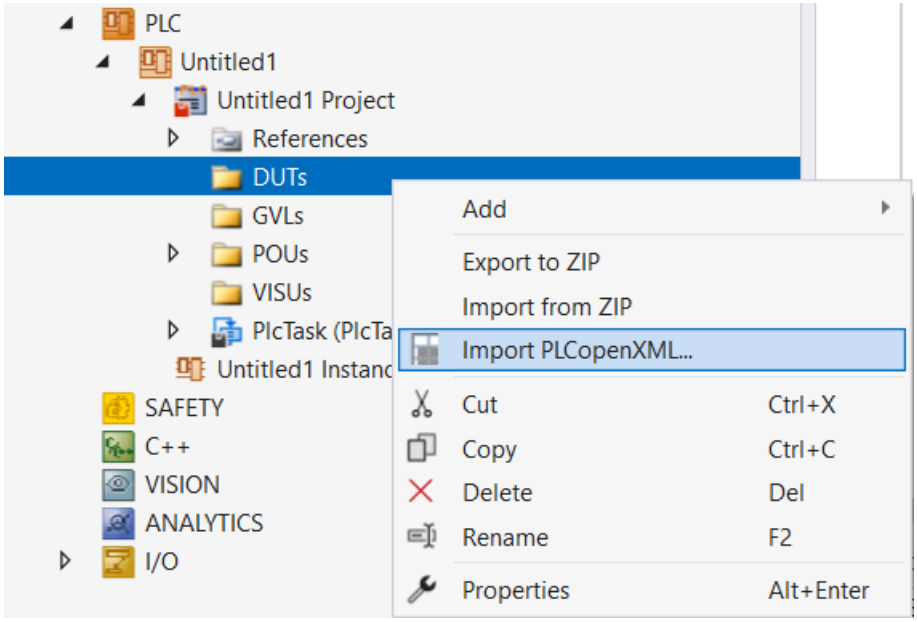
```
TYPE ST_lemon_modelInput :
STRUCT
   {attribute 'hide'} _header_DO_NOT_CHANGE : ARRAY[0..63] OF SINT := [-1,-112,120,86,52,18,-1,-1,1,
-1,0,0,-1,-1,-112,-112,104,-72,-97,-115,-17,4,98,-85,-50,-67,-12,-50,-6,33,-10,-11,88,80,121,12,119,
-56,-24,8,110,-32,-69,113,-21,3,102,-60,64,48,9,0,0,0,0,0,0,0,0,0,0,0,0];
   in_input1 : ARRAY[0..0,0..2,0..223,0..223] OF REAL;
END_STRUCT
END_TYPE
```

Der **Header repräsentiert einen Hash** über die Shape der Eingangstensoren (bzw. beim Outputdatentypen über die Outputtensoren) des KI-Modells. Damit wird sichergestellt, dass der Datentyp zusammen mit der richtigen ONNX verwendet wird. Damit ergibt sich auch, dass Modelle mit identischer Input- und Output-Shape und lexikographischer Reihenfolge dieselben Hashes tragen und damit die Datentypen interoperabel sind. Das ist insbesondere beim Modell-Update [▶ 27] zu bedenken.

Das Byte-Array des Headers darf nicht verändert werden!
Die Namen der Elemente des STRUCT können bei Bedarf angepasst werden, jedoch nicht deren Reihenfolge im STRUCT.

Import einer PlcOpenXml in TwinCAT 3

Die generierte PlcOpenXml kann in der SPS mit Rechtklick auf einen Ordner (z. B. DUTs) über das Feld „Import PLCopenXML“ in das SPS-Projekt übernommen werden.



HINWEIS

Verifizierte Signatur: Betrieb des Machine Learning Servers nur über generierte Eingangs- und Ausgangsmodelltypen

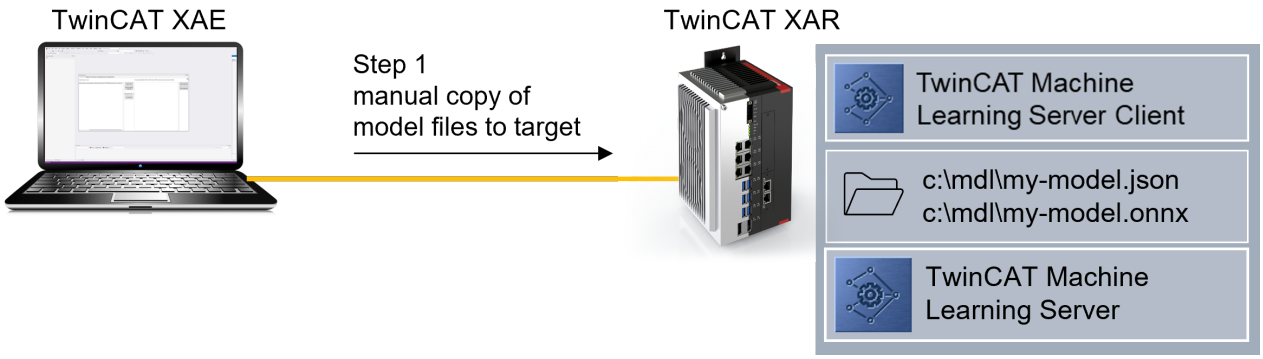
Bitte beachten Sie, dass die Verwendung der in der mitgelieferten PlcOpen-Datei definierten Eingangs-/Ausgangsmodelltypen zwingend erforderlich ist. Die Typen verfügen über eine vom TcMlServer verifizierte Signatur, um sichere Inferenzoperationen zu gewährleisten.

5.1.2 Modellbeschreibungsdateien auf Server-Device verfügbar machen

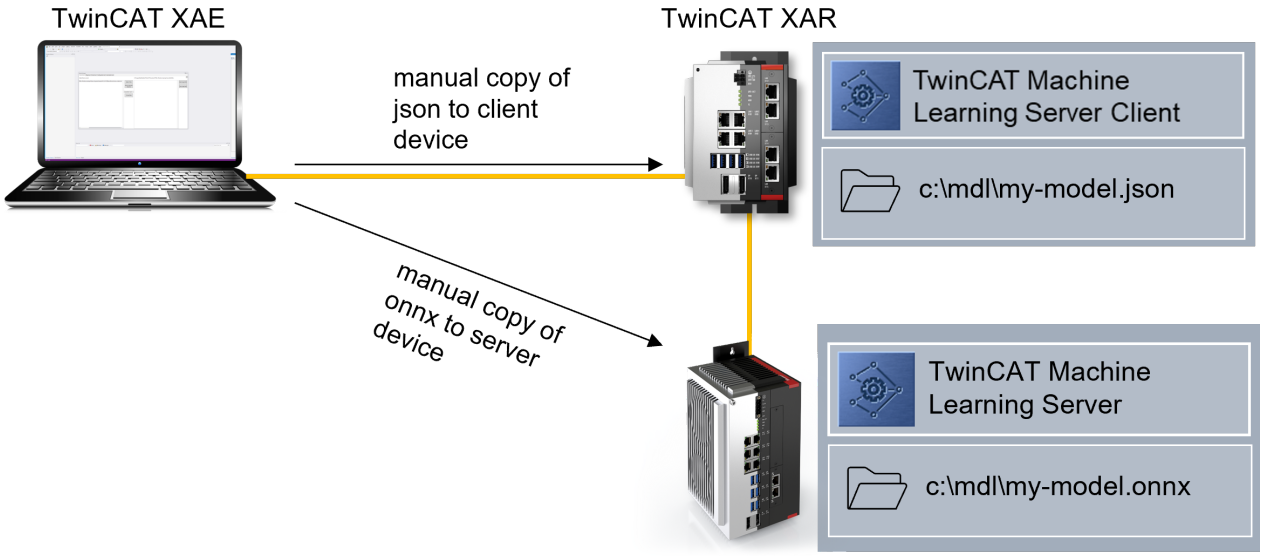
Um Modelle laden zu können, müssen diese dem TwinCAT Machine Learning Server bekannt gemacht werden. Das bedeutet, der Server muss den Ablageort der erstellten JSON- und ONNX-Datei auf dem Dateisystem kennen, um das KI-Modell erfolgreich laden zu können. Diese Bekanntmachung erfolgt über die Configure [▶ 37]-Methode. Der Methode wird der Fullpath zur JSON-Datei übergeben, welche die Interface-Beschreibung trägt. Die zugehörige ONNX-Datei muss im gleichen Ordnerpfad abgelegt sein. Der Zusammenhang zwischen JSON und ONNX wird immer über einen Hash geprüft.

Die ONNX-Datei wird auf dem Gerät benötigt, auf dem der TwinCAT Machine Learning Server installiert ist. Die JSON-Datei wiederum wird auf dem Client-Gerät benötigt. Der Ablagepfad an sich ist frei wählbar; JSON und ONNX müssen aber den gleichen Pfad haben.

Variante 1 - Client und Server sind auf demselben IPC installiert: Die JSON- und ONNX-Datei sind auf dem IPC in einem beliebigen Pfad abzulegen.

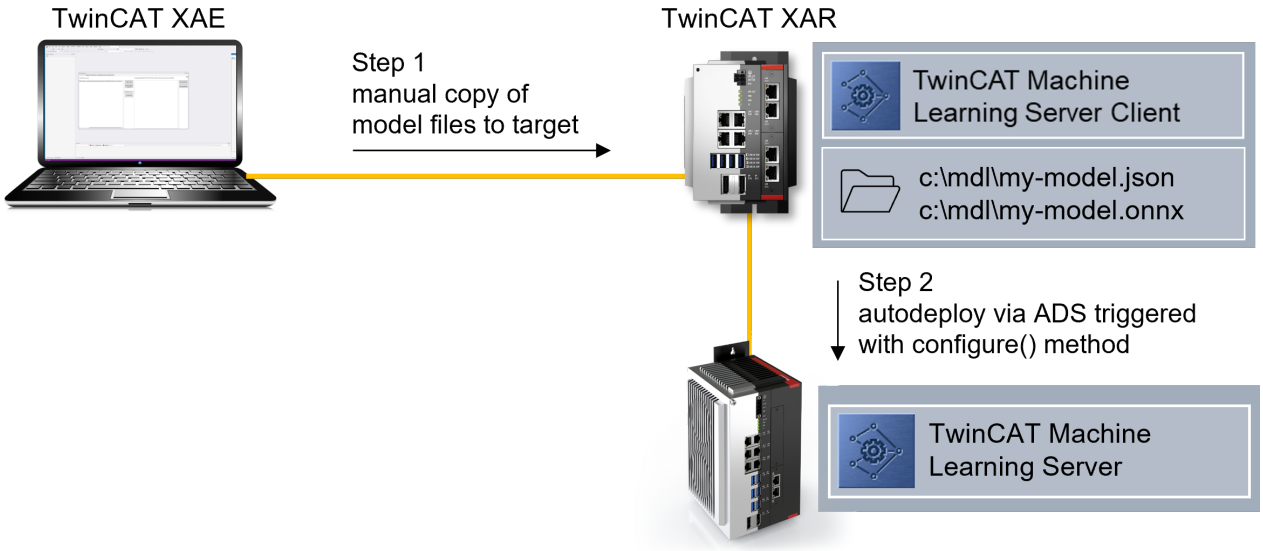


Variante 2 - Client und Server sind auf unterschiedlichen IPCs installiert: Die ONNX-Datei kann direkt auf dem Server abgelegt werden. Die JSON-Datei ist dann auf dem Client-Gerät unter dem gleichen Pfad abzulegen.



Variante 3 - Client und Server sind auf unterschiedlichen IPCs installiert: Die Modell-Dateien (ONNX und JSON) können alternativ beide auf dem Client abgelegt werden. Bei Aufruf der Configure-Methode wird zunächst der Pfad auf dem Server-Device geprüft. Wird die ONNX-Datei dort nicht gefunden, wird der Pfad auf dem Client-Device geprüft. Werden die JSON und ONNX auf dem Client-Device gefunden, wird die ONNX-Datei per ADS auf das Server-Gerät übertragen und dann vom Server geladen. Diese Variante ist zeitaufwändig und wird nur einmalig für dieses Modell durchgeführt. Danach wird das Modell direkt vom Server- Gerät geladen.

Der ADS-Router-Speicher muss groß genug sein, um das Modell über ADS verschicken zu können.



5.1.3 Server aus SPS-Client heraus konfigurieren

Der Aufruf der Methode configure [▶ 37] instanziiert eine Session für die jeweilige Instanz des Funktionsblocks `FB_MlSvrPrediction` im TcMlServer. Bei der Instanziierung wird auf die Konfiguration, die in dem FB-Member stPredictionParameter [▶ 42] definiert ist, zurückgegriffen. In der Regel wird jeder Instanz des `FB_MlSvrPrediction` eine eigene Session im Server zugeordnet. Über den Parameter `bExclusiveSession` können aber auch Sessions gemeinsamen genutzt werden.

Beim Konfigurations-Aufruf wird insbesondere definiert:

- Wo ist der TwinCAT Machine Learning Server?

  - Wird über die AMS Net Id des Server-Geräts spezifiziert. Default-Wert ist „lokal“.
- Welches KI-Modell soll geladen werden?
  - Wird über den Pfad zur entsprechenden JSON-Datei spezifiziert.
- Auf welcher Hardware soll das KI-Modell ausgeführt werden?
  - Wird über den Execution Provider (E_ExecutionProvider [▶ 41]) und optional die DeviceId der GPU definiert.

Jede eröffnete Session allokiert Ressourcen auf dem Server-Gerät. Die Anzahl von parallelen Sessions ist softwareseitig nicht limitiert, sondern wird ausschließlich über die verfügbaren Hardwareressourcen beschränkt. Ist nicht genügend Speicher (RAM oder vRAM) verfügbar, um eine weitere Session zu eröffnen, schlägt der Configure-Befehl fehl.

Mit der Methode deconfigure [▶ 38] kann eine Session geschlossen und damit die Ressourcen freigegeben werden.

Sendet ein Client über eine definierte Zeit keine Anfrage an den Server, die sogenannte Session-Timeout-Zeit, geht der Server davon aus, dass der Client nicht mehr aktiv ist. Die Session wird automatisch geschlossen, wenn der konfigurierte Session-Timeout [▶ 42] erreicht ist.

Beispiel

Deklaration

```
fbMlSvr : FB_MlSvrPrediction();
```

Code

```
// configure session paramaters
fbMlSvr.stPredictionParameter.sMlModelFilePath := 'C:\mdl\lemon_model.json';
fbMlSvr.stPredictionParameter.sMlSvrNetId := '127.0.0.1.1.1';
fbMlSVr.stPredictionParameter.eExecutionProvider := E_ExecutionProvider.CPU;

// Submit configuration request to the TcMlServer
// Provide a generous nTimeout, as the configuration can take a substantial amount of time
IF fbMlSvr.Configure(nTimeout := 1000, nPriority:=0) THEN
// check for error
// change state
END_IF;
```

5.1.4 KI-Modell ausführen

Die Ausführung eines KI-Modells mit dem FB_MlSvrPrediction [▶ 36] wird über die zum SPS-Task-Zyklus asynchrone Methode Predict [▶ 38] (oder im Fall eines *batched* Aufrufs PredictBatched [▶ 39]) getriggert.

Der Methode werden der Eingangsdatentyp und der Ausgangsdatentyp (siehe PlcOpenXml im Bereich Machine Learning Model Manager [▶ 28]), sowie ein Timeout und eine Priorität übergeben.

Inferenzauftrag absenden:

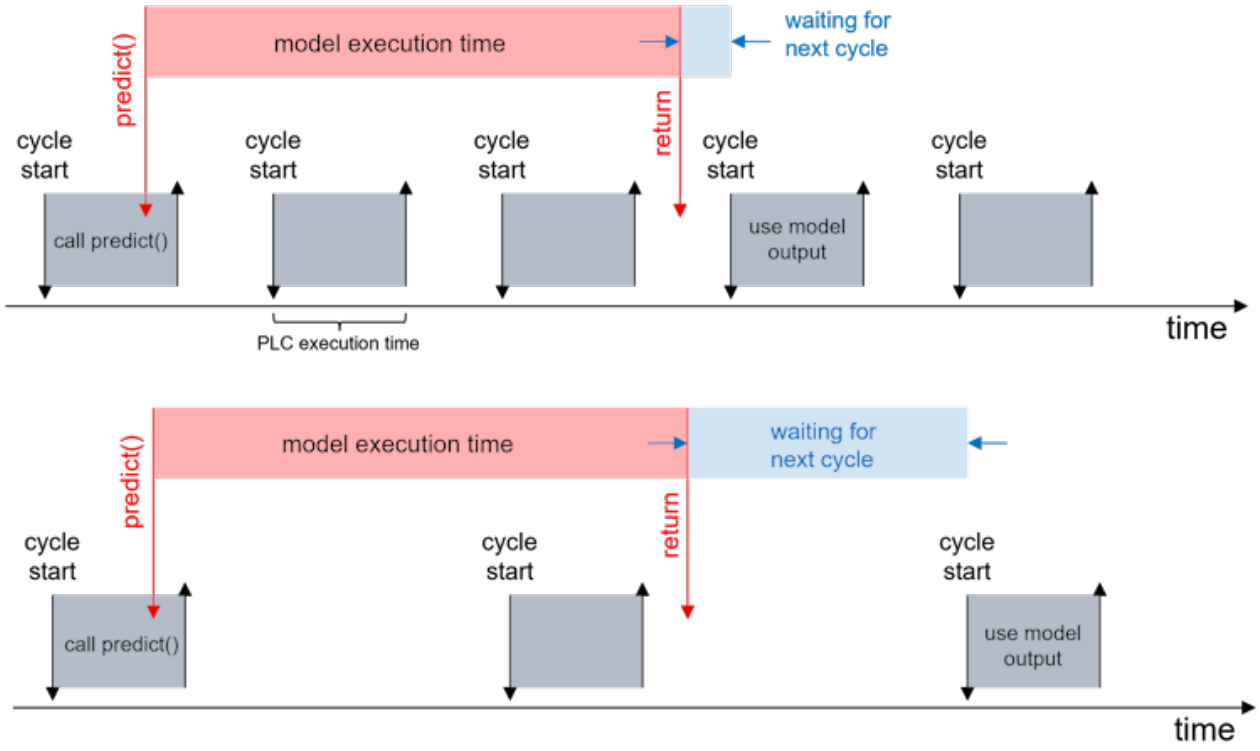
Beim Aufruf der Methode wird der Eingangsdatenbereich per ADS an den Server gesendet. Der dafür notwenige Kopiervorgang wird synchron im Task-Zyklus ausgeführt. Beachten Sie hier, dass größere Datenmengen mehr Zeit benötigen. Im Falle der Übertragung großer Datenmengen, z. B. großer Bilddaten, muss die SPS-Task-Zykluszeit so konfiguriert werden, dass keine Zykluszeitüberschreitungen auftreten.

Inferenzauftrag bearbeiten:

Der Inferenzauftrag wird dann vom TwinCAT Machine Learning Server entgegengenommen. Bei mehreren Anfragen, die nicht gleichzeitig bearbeitet werden können, wird eine Queue erstellt, wobei höhere Prioritäten weiter vorne in der Queue einsortiert werden. Aufträge von CPU-basierten Inferenzen wird immer sequenziell abgearbeitet, d. h. die Queue ist hier besonders relevant. GPU-basierte Inferenzen lassen sich hingegen parallel abarbeiten. Jede FB-Instanz kann immer nur eine ausstehende Anfrage am Server haben, d. h. die maximale Anzahl von Anfragen am Server ist die Anzahl der aktiven Clients.

Inferenzergebnis abfragen:

Es ist wichtig zu berücksichtigen, dass die Abwicklung der asynchronen Anfrage nur mit der zeitlichen Granularität der SPS-Task-Zykluszeit überwacht werden kann. Es ist daher wichtig, dass Applikationen mit geringem Verzugsbudget möglichst niedrige Zykluszeiten und entsprechend hohe Abtastraten aufweisen, um zeitliche auflösungsbedingte Verzögerungen zu minimieren. Dieser Umstand ist insbesondere im Zusammenspiel mit anderen, rechenintensiven und synchron ausgeführten Algorithmen, z. B. bei der Vor- oder Nachverarbeitung der Daten, von hoher Bedeutung.



Beispiel Predict()

Deklaration

```
stModelInput : ST_LemonModelInput; // DUT from PlcOpenXml produced with TC ML Model Manager
stModelOutput : ST_LemonModelOutput; // DUT from PlcOpenXml produced with TC ML Model Manager
fbMlSvr : FB_MlSvrPrediction();
```

Code:

```
// Submission of an inference request at the TcMlServer
// and subsequent postprocessing of the inference result
// Submission of the asynchronous inference request to the TcMlServer
IF fbMlSvr.Predict(
               pDataIn       := ADR(stModelInput),
               nDataInSize   := SIZEOF(ST_LemonModelInput),
               pDataOut      := ADR(stModelOutput),
               nDataOutSize  := SIZEOF(ST_LemonModelOutput),
               nTimeout      := 100,
               nPriority     := 0) THEN
      IF fbMlSvr.nErrorCode <> 0 AND NOT fbMlSvr.bConfigured  THEN
         // If nErrorCode -1 is encountered, increase nTimeout
          eState := E_State.eError;
      ELSE
         // Postprocessing of the inference results
   END_IF
END_IF
```

Beispiel PredictBatched()

Deklaration

```
stModelInputBatch : ARRAY[0..3] OF ST_LemonModelInput;
stModelOutputBatch : ARRAY[0..3] OF ST_LemonModelOutput;
```

Code

```
IF fbMlSvr.PredictBatched(
                pDataIn       := ADR(stModelInputBatch),
                nDataInSize   := SIZEOF(ST_LemonModelInput),
                nBatchSize    := 4,
                pDataOut      := ADR(stModelOutputBatch),
                nDataOutSize  := SIZEOF(ST_LemonModelOutput),
                nTimeout      := 100,
                nPriority     := 0) THEN
```

In den Beispielaufrufen wird der Timeout für die Inferenzanfrage jeweils auf 100 Zyklen gesetzt. In der obigen Abbildung steht das Ergebnis nach 3 Zyklen (oben) bzw. 2 Zyklen (unten) zur Verfügung. Der unter Umständen erfahrbare Jitter kann über `fbMlSvr.nMaxInferenceDuration` abgefragt werden. Dieser zeigt das Maximum der Anzahl von SPS-Zyklen, die für die Ausführung einer Inferenz erforderlich waren, an. Anhand dieses Werts lässt sich der Timeout-Wert in der Regel gut auslegen.

5.1.5 KI-Modell updaten

KI-Modelle können zur Laufzeit der Maschine ausgetauscht werden. Im Folgenden werden zwei Fälle mit den entsprechend zu befolgenden Schritten beschrieben.

Fall 1: Modellupdate ohne Änderung des Modell-Interface

Definition des Falls:

In diesem Fall bleibt das Eingangs- und Ausgangsinterface zum KI-Modell beim Tausch des Modells identisch. Dazu müssen die Input- und Output-Nodes in ihrer Reihenfolge (wenn mehrere Nodes) und in ihrer Shape unverändert bleiben.

Es wird empfohlen, die gesamte Modellarchitektur beim Modellupdate nicht zu verändern, also ein Transfertraining/Fine-Tuning auf dem bestehenden KI-Modell zu machen. Dadurch verändern sich die Interfaces zum Modell nicht und auch das Laufzeitverhalten bleibt gleich.

Ein Modellupdate ist ohne Compile-Prozess und ohne TwinCAT-Stopp durchführbar.

Schritte zum Update des Modells:

- JSON und PlcOpenXml mit dem TwinCAT Machine Learning Model Manager erstellen.
- JSON und ONNX auf den relevanten Systemen verfügbar machen. Falls der Fullpath sich geändert hat, zum Beispiel per ADS den neuen Pfad in einer Variablen bekannt machen.
- Der Hash der Ein- und Ausgangsdatentypen ändert sich nicht. Damit muss **keine** neue PlcOpenXml eingelesen werden. Die neue PlcOpenXml bleibt unbenutzt.
- In der laufenden SPS den State wechseln, z. B. per ADS eine entsprechende Variable setzen, und Deconfigure [▶ 38]() aufrufen, um die aktuelle Session zu schließen.
- Configure aufrufen, um die neue JSON zu laden.

Während der Zeit von Deconfigure bis zum Abschluss der Configure-Methode können mit diesem Funktionsbaustein keine Inferenzaufrufe zum Server gesendet werden.

Fall 2: Modellupdate mit Änderung des Modell-Interface

Definition des Falls:

In diesem Fall verändert sich das Eingangs- und Ausgangsinterface zum KI-Modell beim Tausch des Modells. Dadurch passen die Eingangs- und Ausgangsdatentypen in der SPS nicht mehr zum Modell. In der Regel sind dann mehrere Stellen im Quellcode zu ändern.

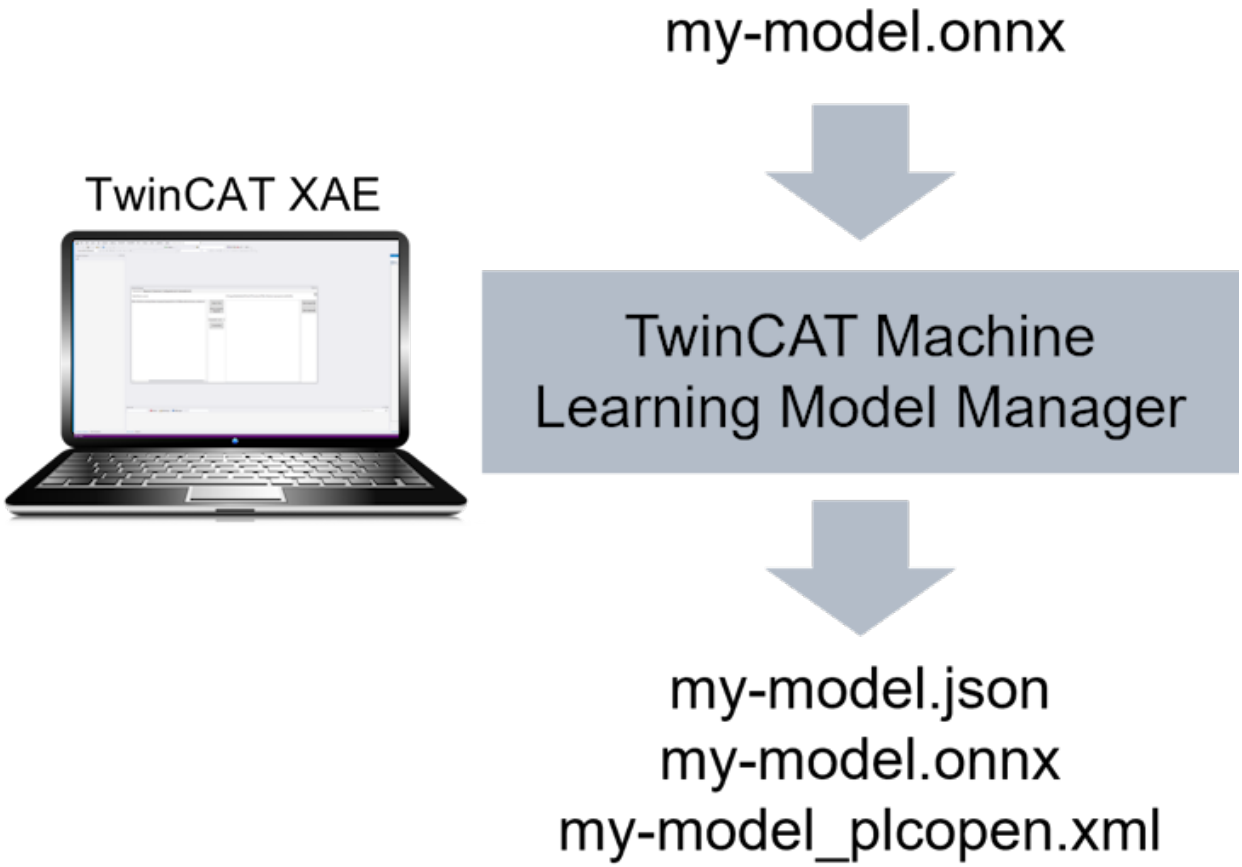
Es wird empfohlen, bei einer solchen Änderung den Quellcode im TwinCAT XAE zu überarbeiten, das Projekt neu zu testen und erst dann auf die Maschine zu laden. Beachten Sie, dass sich auch das Laufzeitverhalten des KI-Modells verändert haben kann.

In diesem Fall ist ein Modellupdate mit einem TwinCAT-Stopp beim Aufspielen des neuen, veränderten TwinCAT-Projekts verbunden.

5.2 TwinCAT Machine Learning Model Manager

Der TwinCAT Machine Learning Model Manager dient dazu, die ONNX-Modelldateien zu verwalten und für die Nutzung mit TwinCAT vorzubereiten. Die Funktionen des TwinCAT Machine Learning Model Managers sind zusammengefasst:

- Erstellung einer Meta-Informationsdatei (JSON-File).
- Erstellung einer PlcOpenXml, welche eine SPS-Datentypbeschreibung des Modell Eingangs und des Modell Ausgangs beschreibt.
- Optional: Einfügen von anwendungsspezifischen Meta-Informationen (Custom Attributes). Die Custom Attributes kann ein Nutzer zu den JSON hinzufügen. Die Attribute sind dann mit Methoden am FB_MlSvrPrediction [▶ 36] zur Laufzeit in der SPS auslesbar.



Der TwinCAT Machine Learning Model Manager hat drei unterschiedliche Interfaces:

1. Ein graphisches User Interface (Visual Studio Plugin)
2. Ein Python Interface (Python package)
3. Ein **C**ommand **L**ine **I**nterface (CLI)

Diese werden im Folgenden näher erläutert.

5.2.1 Graphisches User Interface

Der TwinCAT 3 Machine Learning Model Manager ist das zentrale UI zum Bearbeiten von ONNX-Dateien. Das Werkzeug ist in Visual Studio integriert und kann über die Menüleiste unter **TwinCAT > Machine Learning > Machine Learning Model Manager** geöffnet werden.

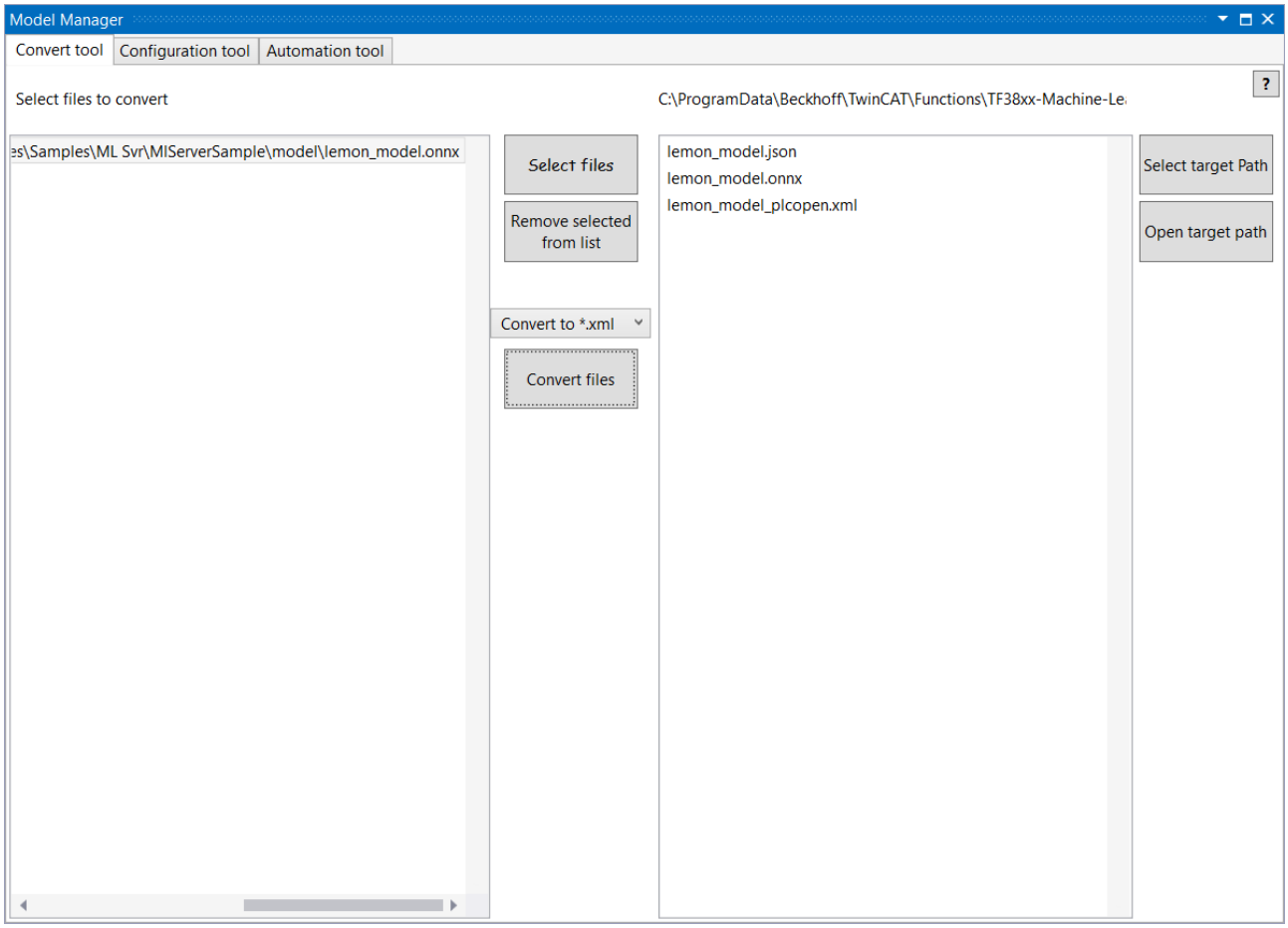
Voraussetzung an Visual Studio Version

Die graphische Oberfläche des TwinCAT 3 Machine Learning Model Managers ist kompatibel mit Visual Studio 2017, 2019, 2022 sowie der TcXaeShell.

Erstellen der JSON und PlcOpenXml

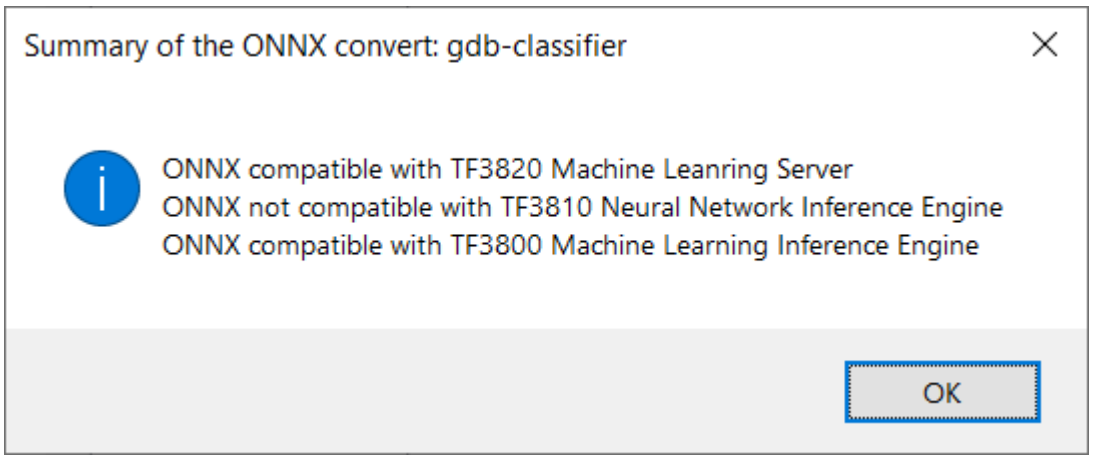
1. Öffnen Sie den Reiter **Convert Tool.**
   ⇨ Über **Select files** öffnet sich der File Browser.
2. Wählen Sie ONNX-Dateien aus (multi-select möglich durch Strg-Click).
   ⇨ Selektierte ONNX-Dateien werden auf der linken Seite mit ihrem Pfad und Dateinamen gelistet.
3. Bei Bedarf können Sie Dateien auswählen und durch Klicken des Buttons **Remove selected from list** wieder aus der Liste entfernen.
4. Klicken Sie auf **Convert files**, um die benötigten JSON und PlcOpenXml zu erstellen.
   ⇨ Die Dateien werden im converted file path abgelegt. Default ist der Pfad *<TwinCATPath>\Functions\TF38xx-Machine-Learning\ConvertToolFiles*.
5. Klicken Sie auf **Open target path**, um den converted file path im File Browser zu öffnen.

⇨ Mit **Select target path** kann der Pfad verändert werden. Die Änderung bleibt auch nach Neustart des PCs erhalten.



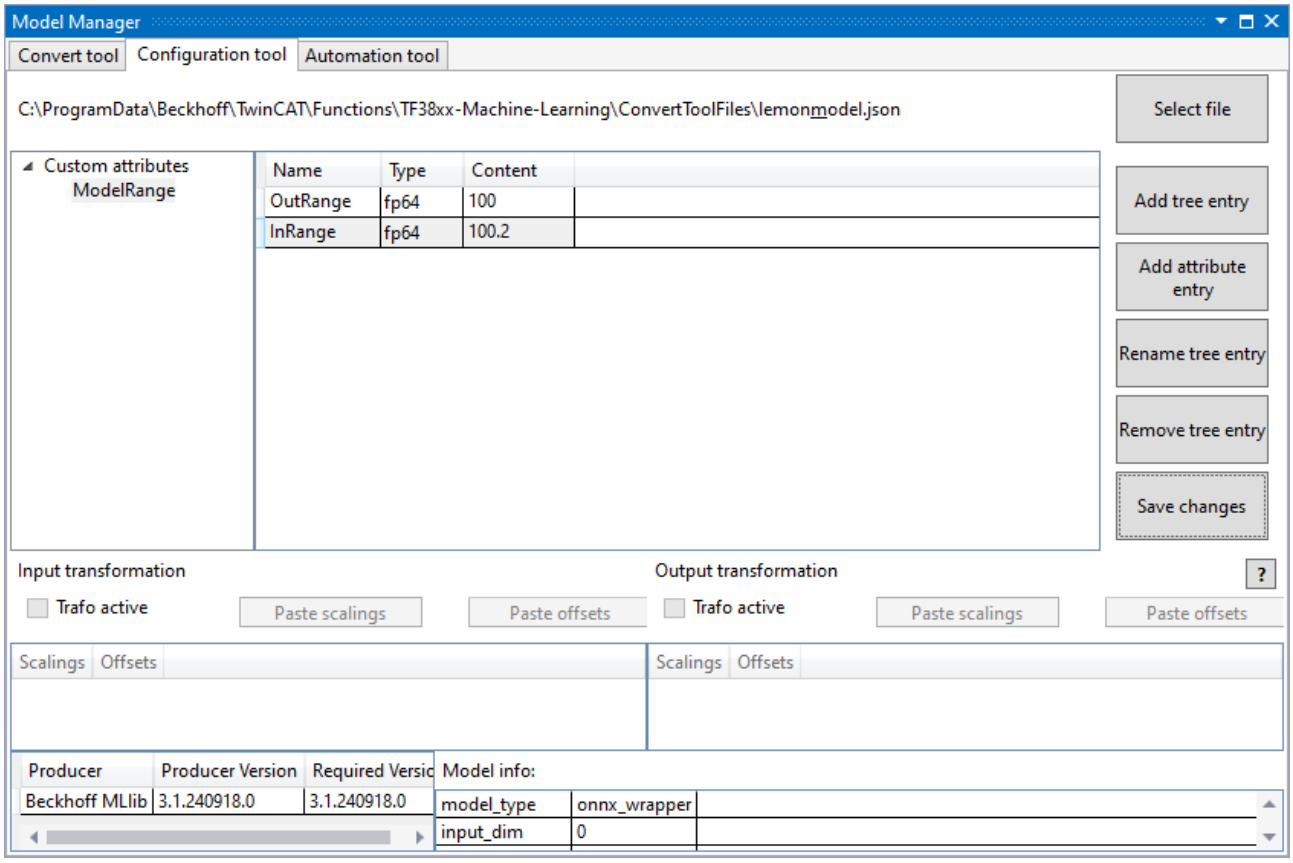
Das hier beschriebene Werkzeug ist ebenfalls das zentrale Werkzeug zum Verwalten von ONNX-Dateien für die TwinCAT Machine Learning Inference Engine und TwinCAT Neural Network Inference Engine. Aus diesem Grund werden auch Beschreibungsdateien für die optionale Ausführung des KI-Modells mit diesen Produkten erzeugt, falls die ONNX kompatibel ist. Für diese Produkte ist das Dropdown-Menü Convert to *.xml relevant. Für den TwinCAT Machine Learning Server ist dies nicht von Belang.

Ein PopUp liefert für jede konvertierte ONNX entsprechendes Feedback darüber, mit welchem TwinCAT-Produkt die betrachtete ONNX kompatibel ist.



Erstellen von Custom Attributes

Eine JSON kann über **Select file** ausgewählt und dann bearbeitet werden. Nach der Bearbeitung wird mit **Save changes** die Originaldatei überschrieben.



Die Bearbeitung der Custom Attributes erfolgt über die Buttons:

- **Add attribute entry**: Fügt dem ausgewählten Tree item ein Attribut hinzu. Das Tree item ist in der linken Liste zu selektieren.
  - Wird ein Attribut angelegt, sind der Name, der Datentyp und der Wert bzw. die Werte anzugeben.
  - Das Löschen von Attributen erfolgt durch Selektieren des Attributes und Drücken der Taste Entfernen.
- **Add tree entry**: Fügt ein Tree item unter dem selektierten Tree item (als Sub Tree Item) ein.
- **Rename tree entry**: Das selektierte Tree Item kann umbenannt werden.
- **Remove tree entry**: Das selektierte Tree Item, inkl. der Sub Tree Items, wird gelöscht.

5.2.2 Python Interface

Installation des Python Package

Das Python Package liegt als whl-Datei im Ordner <TwinCatInstallDir>\Functions\TF38xx-Machine-Learning\Utilities\ModelManagerAPI\PythonPackage.

Um das Package zu installieren, nutzen Sie `pip install` <TwinCatInstallDir>\Functions\TF38xx-Machine-Learning\Utilities\ModelManagerAPI\PythonPackage\<whl-file-name>. Der Ordner enthält ggf. unterschiedliche Versionen des Package (nur, wenn Sie ein neues Setup über ein altes TwinCAT Machine Learning Setup installiert haben).

Achten Sie darauf, immer die aktuelle Version zu nutzen.

Verwenden der API

Das Package wird geladen mit

```
import  beckhoff.toolbox as  tb
```

Erstellen einer JSON und PlcOpenXml aus einer ONNX-Datei

```
tb.onnxprep("C:\\PathTo\\myONNX.onnx")
```

Anzeigen von **Modellinformationen**

```
tb.info("C:\\PathTo\\myONNX.json")
```

Custom Attributes hinzufügen

```
new_ca = { 'nID' : -34234, 'bTested' : True, 'fNum' : 324.3E-12, 'AnotherTreeItem' : { 'fPi' : 3.134
12, 'bFalseFlag' : False }}
tb.modify_ca("C:\\PathTo\\myONNX.json","C:\\PathTo\\myONNX_ca.json", new_ca)
```

Model Description (Name, Version, Autor usw. eines Modells) hinzufügen

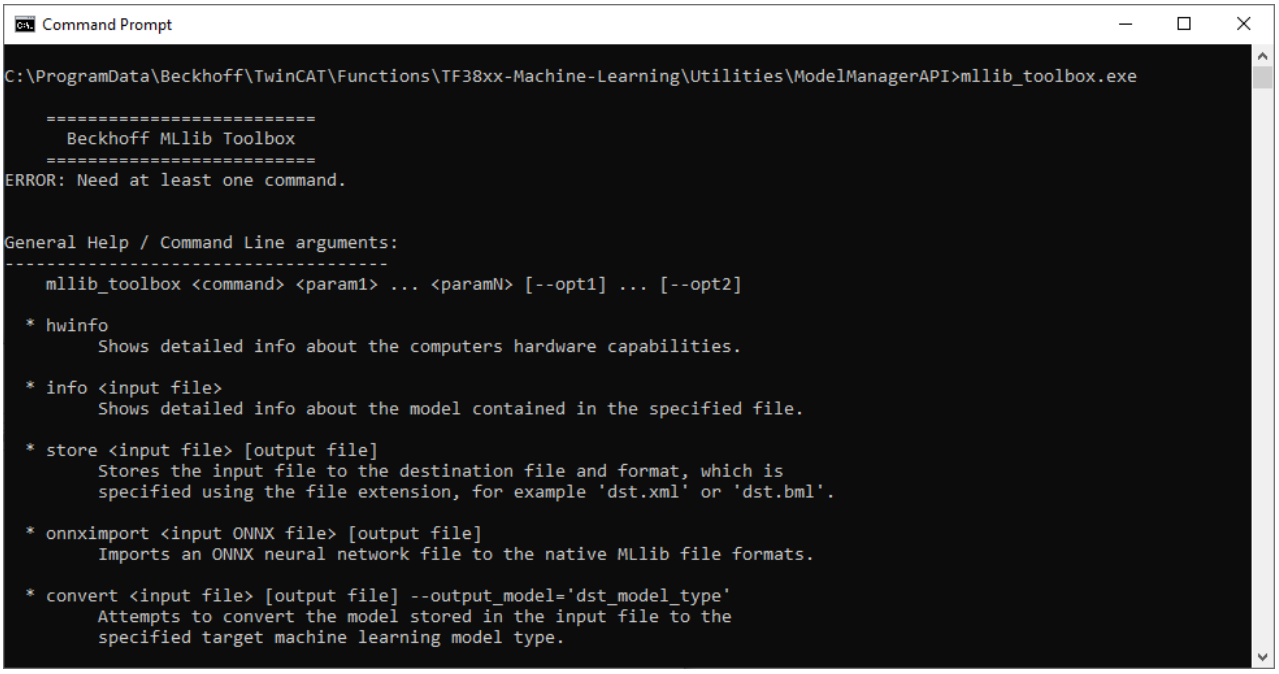
```
model_description = {
    "new_version" : "2.3.1.0",
    "new_name" : "CurrentPreControlAxis42",
    "new_desc":"This is the most awesome model to control Axis42",
    "new_author":"Max",
    "new_tags":"awesome, ingenious, astounding",
}
tb.modify_md("C:\\PathTo\\myONNX.json", "C:\\PathTo\\myONNX_md.json",
             **model_description)
```

5.2.3 Command Line Interface

Der Model Manager kann auch über das Command Prompt genutzt werden. Dazu steht die mllib_toolbox.exe zur Verfügung.

Die Executable ist unter *<TwinCatInstallDir>\Functions\TF38xx-Machine-Learning\Utilities\ModelManagerAPI* zu finden.

Die Hilfe wird Ihnen angezeigt, indem Sie die exe aufrufen ohne Argumente.



Verwenden der API

Erstellen einer JSON und PlcOpenXml aus einer ONNX-Datei

```
mllib_toolbox.exe onnxprep C:\PathTo\mymodel.onnx
```

Anzeigen von **Modellinformationen**

```
mllib_toolbox.exe info C:\PathTo\mymodel.json
```

Das Anlegen von Custom Attributes wird im CLI nicht unterstützt.

5.3 ONNX Support

Unterstützte ONNX opset Version

Der TwinCAT Machine Learning Server, genauer der Dienst TcMlServer, unterstützt die ONNX Opset Version 21. Eine Abwärtskompatibilität zu jüngeren ONNX Opset Versionen ist in der Regel seitens ONNX gegeben.

Die verwendete Opset Version wird normalerweise im ONNX-File unter *imports* geführt. Im untenstehenden Bild, visualisiert mit Netron, beispielhaft ONNX Opset Version 18.

Einschränkungen unterstützter ONNX-Eigenschaften

Keine dynamischen Input- oder Output-Shapes

Dynamische Input- oder Output-Shapes werden nicht unterstützt. Nur der führende Batchsize Parameter darf eine dynamische Größe sein (siehe nächster Absatz).

Erlaubt sind beispielsweise:

```
float32[244, 244, 1]
```

```
float32[1, 3, 244, 244]
```

```
float32[5, 244, 244, 3]
```

Nicht erlaubt sind:

```
float32[244, 244, ?]
```

```
float32[244, height, 3]

float32[?, 244, 244, ?]

float32[1, 244, 244, unknown]
```

Sie können die Shape von ONNX Nodes, z. B. mit Hilfe des onnxruntime pakets in Python, schnell fixieren, siehe Make dynamic input shape fixed | onnxruntime.

Aus einer Shape…

```
float32[-1, 3, ?, ?]
```

wird mit…

```
python -m onnxruntime.tools.make_dynamic_shape_fixed --input_name x --input_shape 1,3,960,960
model.onnx model.fixed.onnx
```

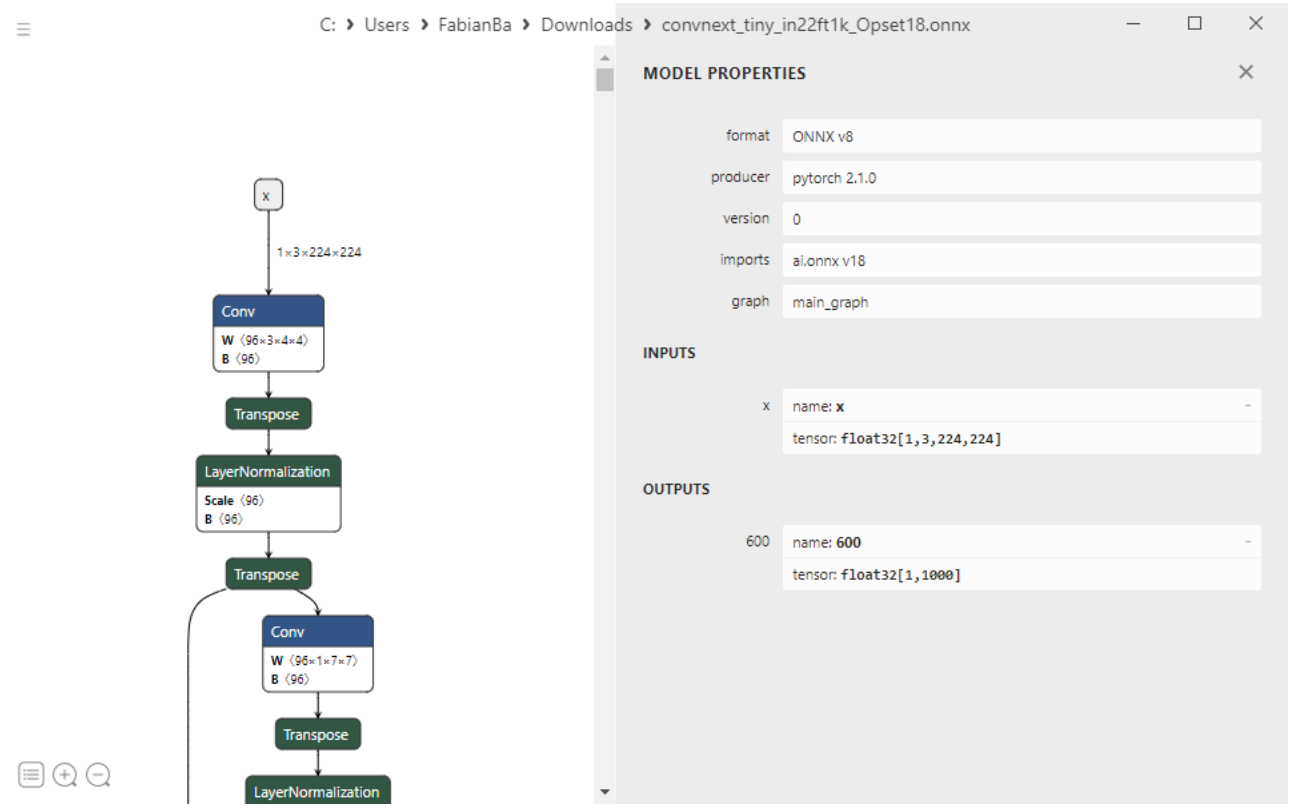
```
float32[1, 3, 960, 960]
```

Batchsize muss führend sein

Der TwinCAT Machine Learning Server unterstützt nur Modelle mit Batchsize als führenden Parameter der Input-Node. Der Batchsize Parameter darf (im Gegensatz zu allen anderen Parametern) dynamisch sein. Erlaubt sind beispielsweise:

```
float32[batch, 3, 244, 244]

float32[?, 3, 244, 244]

float32[unknown, 244, 244, 3]
```

Nur, wenn das Modell als führenden Parameter eine dynamische Batchsize enthält, kann mit der Methode PredictBatched() [▶ 39] gearbeitet werden.



ONNX-Datei Größe

Aktuell ist das Größenlimit einer ladbaren ONNX-Datei 2 GB. Kontaktieren Sie Beckhoff (siehe Support und Service [▶ 53]), falls diese Grenze für Ihre Anwendung eine Herausforderung darstellt.

5.4 TcMlServer Service

Die Installation von TF3820 kann durch Überprüfung des Vorhandenseins des Dienstes TcMlServer.exe überprüft werden. Beachten Sie, dass der TcMlServer-Dienst einen verzögerten autonomen Start beim Neustart bietet. Beachten Sie hier die verzögerte Startzeit. Der TcMlServer allokiert beim Starten auf dem System den ADS-Port 19900.

Systemanforderungen

Neben einem 64-bit Windows- und einer TC1000 TwinCAT 3 ADS Installation (Workload TC1000.ADS.XAR), stellt der TcMlServer formal keine weiteren Anforderungen an das System außer einem Festplattenbedarf von ca. 500 MB. Für ein performantes Betreiben des TcMlServers ist jedoch eine möglichst hohe Anzahl an UM-Kernen zu empfehlen.

Lizenzabfrage

Beim Starten prüft der TcMlServer, ob die TF3820-Lizenz vorhanden ist. Wenn die Lizenz zunächst nicht vorhanden ist, wird der Lizenzstatus alle zehn Sekunden abgefragt. Ein Lizenzerwerb kann daher einen kurzen Moment dauern, bis er wirksam wird. Ein entsprechender Fehlercode wird bei Anfragen ausgegeben, wenn der TcMlServer nicht korrekt lizenziert ist.

Installationspfad und Log-Files:

C:\ProgramData\Beckhoff\TwinCAT\Functions\TF38xx-Machine-Learning\TcMlServer

In Supportfällen stehen im Ordner logs Logging-Datein mit etwaig aufgezeichneten Meldungen zur Verfügung.

Der logs-Ordner weist die folgende Struktur auf:

Nach jedem neuen Aufstarten legt der TcMlServer ein neues Verzeichnis, dessen Name sich aus Erstellungsdatum und -uhrzeit zusammensetzt, an. In dem Verzeichnis werden die logs im JSON-Dateiformat abgelegt und regelmäßig, sofern vorhanden, vom TcMlServer weggeschrieben. Im Fehlerfall können daher mehrere JSON-Dateien vorhanden sein, deren Namen ein Intervall der enthaltenen Fehler oder Warnungen spezifizieren.

5.4.1 Execution Provider

Als ExecutionProvider stehen bislang zur Verfügung:

- CPU
- CUDA

Der ExecutionProvider [▶ 41] wird mit der Configure [▶ 37]-Methode des FB_MlSvrPrediction [▶ 36] übergeben.

CPU

Die Ausführung des geladenen KI-Modells wird auf den CPU-Ressourcen des IPC ausgeführt. Ist auf demselben Gerät eine TwinCAT-Laufzeit aktiv, können nur die CPU-Ressourcen genutzt werden, die nicht von TwinCAT beansprucht werden (Isolated Cores werden nur von TwinCAT verwendet, Shared Cores können nur zeitlich eingeschränkt genutzt werden). Das Betriebssystem übernimmt die Parallelisierung der Berechnung auf alle zur Verfügung stehenden Threads.

Erstellen mehrere Clients eine Session mit Execution Provider CPU auf einem Server, so werden die Inferenzanfragen nacheinander abgearbeitet. Beachten Sie dazu den `Priority` Parameter beim Predict-Aufruf.

CUDA

Die Ausführung des geladenen KI-Modells wird auf den GPU-Ressourcen des IPC ausgeführt. Es können mehrere Sessions auf einer GPU parallel betrieben werden insofern die Ressourcen der GPU ausreichen.

Das Feld `nDeviceId` ermöglicht die Unterscheidung zwischen potentiell mehreren installierten Graphikkarten. Für Rechner mit höchstens einer Graphikkarte kann der Wert auf dem Defaultwert belassen werden, ansonsten entspricht das Feld dem CUDA compute index der Graphikkarten. Sollte ein Rechner mit mehreren Graphikkarten verwendet werden, sollten folgende **Systemumgebungsvariable** gesetzt werden:

```
CUDA_DEVICE_ORDER='PCI_BUS_ID'
```

Weiterhin ermöglicht der TcMlServer das Teilen von Inferenzressourcen zwischen verschiedenen FBs, die eine gleiche Spezifikation für ihre Inferenzsession bereitstellen. Die Nutzung einer geteilten Inferenzmaschine (`bExclusiveSession = FALSE`) kann in Fällen vorkommen, in denen anderweitig ein Engpass -zum Beispiel an Speicher auf Graphikkarten- zu erwarten wäre. Für zustandsbehaftete Modelle (z. B. rekurrente Modelle), ist aber von einer solchen Konfiguration abzusehen.

6 API

6.1 Funktionsbausteine

6.1.1 FB_MlSvrPrediction

`FB_MlSvrPrediction` ist ein TcMlServer-Client, der asynchrone und optional hardwarebeschleunigte KI-Modellinferenz für die SPS bereitstellt.

Der Funktionsbaustein liegt in der **SPS-Bibliothek Tc3_MlServer**.

Syntax

Deklaration:

```
fbMlSvr : FB_MlSvrPrediction;
```

Definition:

```
FUNCTION_BLOCK FB_FTR_IIRCoeff
VAR_INPUT
    stPredictionParameter : ST_PredictionParameter;
END_VAR
VAR_OUTPUT
    bError                : BOOL;
    nErrorCode            : HRESULT;
    bConfigured           : BOOL;
    nMaxInferenceDuration : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stPredictionParameter [▶ 42]	ST_PredictionParameter	Konfigurationsstruktur der Session des Machine Learning Servers

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, wenn aktuell anstehende asynchrone Anfrage an den TcMLServer mit einem Fehler abgebrochen wurde.
nErrorCode	HRESULT	Rückgabecode [▶ 49] für die aktuell anstehende asynchrone Anfrage an den TcMlServer.
bConfigured	BOOL	TRUE, wenn die Konfiguration erfolgreich war. Zeigt an, ob der Funktionsbaustein eine gültige Session beim TcMlServer hat. Wenn FALSE, muss „configure“ aufgerufen werden. Bitte beachten: bConfigured kann aufgrund eines Fehlers während der Ausführung der Anfrage beim TcMlServer auf FALSE übergehen.
nMaxInferenceDuration	UDINT	Maximum der Anzahl von SPS-Zyklen, die für die Ausführung einer Inferenz erforderlich waren. Siehe auch KI-Modell ausführen [▶ 25].

Methoden

Name	Definitionsort	Beschreibung
Configure [▶ 37]	Lokal	Erzeugen einer Session für die FB-Instanz auf dem TcMlServer entsprechend der in stPredictionParameter [▶ 42] definierten Konfiguration.

Name	Definitionsort	Beschreibung
Deconfigure [▶ 38]	Lokal	Beenden einer Session des Funktionsbausteins auf dem TcMlServer. Die allokierten Ressourcen auf dem Server werden wieder freigegeben.
GetCustomAttribute_array [▶ 40]	Lokal	Lese custom attributes von Typ ARRAY des KI-models
GetCustomAttribute_bool [▶ 40]	Lokal	Lese custom attributes von Typ BOOL des KI-models
GetCustomAttribute_fp64 [▶ 41]	Lokal	Lese custom attributes von Typ LREAL des KI-models
GetCustomAttribute_int64 [▶ 41]	Lokal	Lese custom attributes von Typ LINT des KI-models
GetCustomAttribute_str [▶ 41]	Lokal	Lese custom attributes von Typ STRING des KI-models
Predict [▶ 38]	Lokal	Übermittlung einer asynchronen Inferenzanfrage an den TcMlServer
PredictBatched [▶ 39]	Lokal	Übermittlung einer asynchronen, gebündelten Inferenzanfrage an den TcMLServer

6.1.1.1 Asynchrone Methoden

Die asynchronen Methoden des `FB_MlSvrPrediction` zeichnen sich in ihrer Signature dadurch aus, dass sie einen Parameter *timeout* entgegennehmen und einen *bool*, der den Ausführungszustand der asynchronen Anfrage anzeigt, zurückgeben. Das SPS-Programm muss folglich die Beendigung der Ausführung der asynchronen Methoden abwarten und den *timeout* gemäß den Anforderungen der Applikation so konfigurieren, dass die Applikation angemessen auf eventuelle Verzögerungen des TcMlServer reagieren kann. Hierbei ist zu berücksichtigen, dass der angegebene *timeout* in der Einheit SPS-Task-Zyklen angeben wird.

Es ist weiterhin wichtig zu berücksichtigen, dass die Abwicklung der asynchronen Anfrage selbstverständlich nur mit der zeitlichen Granularität der **SPS-Task-Zykluszeit** überwacht werden kann. Es ist daher wichtig, dass Applikationen mit geringem Verzugsbudget möglichst niedrige Zykluszeiten und entsprechend hohe Abtastraten aufweisen, um zeitliche auflösungsbedingte Verzögerungen zu minimieren. Dieser Umstand ist insbesondere im Zusammenspiel mit anderen, rechenintensiven und synchron ausgeführten Algorithmen, z. B. bei der Vor- oder Nachverarbeitung der Daten, von hoher Bedeutung.

Eine Einschränkung bzgl. der minimalen SPS-Task-Zykluszeit ist jedoch durch die Menge der Daten, die vom Client zum Server transportiert werden müssen, gegeben. Im Falle der Übertragung großer Datenmengen, z. B. großer Bilddaten, muss die SPS-Task-Zykluszeit so konfiguriert werden, dass keine Zykluszeitüberschreitungen auftreten.

6.1.1.1.1 Configure()

Der Aufruf der Methode *configure* instanziiert eine Session für die jeweilige Instanz des Funktionsblock `FB_MlSvrPrediction` im TcMlServer. Bei der Instanziierung wird auf die Konfiguration, die in dem FB-Member `stPredictionParameter` definiert ist, zurückgegriffen.

Die Instanziierung einer Session kann eine erhebliche Zeit in Anspruch nehmen, da insbesondere mehrere Inferenzen zur Temperierung der Inferenzmaschine vorgenommen werden (*model warm-up*). Dieser Umstand sollte bei der Wahl des timeout-Parameters des Methodenaufrufs berücksichtigt werden.

Es sollte weiterhin berücksichtigt werden, dass der Aufruf der Methode bei Konfiguration einer CUDA beschleunigten Session vorübergehend einen exklusiven Zugang zur GPU erfordert. Die Konfiguration einer solchen Session kann also erheblich mit der Inferenzleistung anderer parallel operierender FBs interferieren.

Nachdem die Methode die Beendigung der Bearbeitung des asynchronen Instanziierungsaufrufes durch den Rückgabewert TRUE anzeigt, kann das Ergebnis über die FB-Member bError und im Fehlerfall nErrorCode ausgewertet werden. Nach erfolgreicher Instanziierung einer Inferenzsession wird der FB den Member `bConfigured` auf TRUE setzen.

Siehe auch Server aus SPS-Client heraus konfigurieren [▶ 24].

	Parameter	Typ	Default	Beschreibung
INPUT	nTimeout	ULINT		Anzahl der SPS-Task-Zyklen, bevor der Zeitüberschreitungsfehler zurückgegeben wird.
INPUT	nPriority	UDINT	0	Priorität der Anfrage. Größer bedeutet höhere Priorität.
OUTPUT	Configure	BOOL		Rückgabewert. TRUE, sobald das Ergebnis des asynchronen Aufrufs vorliegt. Das Ergebnis des Aufrufs kann dann mit den Properties 'bError' und 'nErrorCode' überprüft werden.

6.1.1.1.2 Deconfigure()

Die Methode Deconfigure schließt die Inferenzsession des FBs im TcMlServer. Nach einem erfolgreichen Aufruf von Deconfigure, kann der FB eine neue Inferenzsession über die Methode Configure einrichten.

Nach einer erfolgreichen Konfiguration einer Inferenzsession werden alle Aufrufe von Configure ignoriert, bis ein Aufruf von Deconfigure erfolgt ist.

	Parameter	Typ	Default	Beschreibung
INPUT	nTimeout	ULINT		Anzahl der SPS-Task-Zyklen, bevor der Zeitüberschreitungsfehler zurückgegeben wird.
OUTPUT	Deconfigure	BOOL		Rückgabewert. TRUE, sobald das Ergebnis des asynchronen Aufrufs vorliegt. Das Ergebnis des Aufrufs kann dann mit den Properties 'bError' und 'nErrorCode' überprüft werden.

6.1.1.1.3 Predict()

Die Methode Predict reicht einen asynchronen Inferenzauftrag beim TcMlServer ein.

Die Method erwartet die Bereitstellung von Inputdaten gemäß der Spezifikation der ONNX-Datei, siehe ONNX für die Verwendung mit TwinCAT Machine Learning Server vorbereiten [▶ 20].

Der bereitgestellte Pointer auf die Outputdaten muss valide sein und auf einen eine Instanz des Output-Datentypen gemäß der erstellten PlcOpenXml zeigen. Nach erfolgreichem Abschluss der asynchronen Inferenz sind die Daten im übergeben Output-Speicherbereiches gültig und zur Weiterverarbeitung freigegeben.

Siehe auch KI-Modell ausführen [▶ 25].

	Parameter	Typ	Default	Beschreibung
INPUT	pDataIn	PVOID		Pointer auf die Instanz des Eingangsdatentypen

	Parameter	Typ	Default	Beschreibung
INPUT	nDataInSize	UDINT	0	Größe des Eingangsdatentypen
INPUT	pDataOut	PVOID		Pointer auf die Instanz eines Ausgangsdatentype n
INPUT	pDataOutSize	UDINT		Größe des Ausgangsdatentype n
INPUT	nTimeout	ULINT		Anzahl der SPS-Task-Zyklen, bevor der Zeitüberschreitungsf ehler zurückgegeben wird.
INPUT	nPriority	UDINT	0	Priorität der Anfrage. Größer bedeutet höhere Priorität.
OUTPUT	Predict	BOOL		Rückgabewert. TRUE, sobald das Ergebnis des asynchronen Aufrufs vorliegt. Das Ergebnis des Aufrufs kann dann mit den Properties 'bError' und 'nErrorCode' überprüft werden.

6.1.1.1.4 PredictBatched()

Die Methode PredictBatched reicht einen asynchronen Batch-Inferenzauftrag beim TcMlServer ein.

Die Method erwartet die Bereitstellung eines Arrays von dem Input-Datentyp des Modelles gemäß der Spezifikation der ONNX-Datei, siehe ONNX für die Verwendung mit TwinCAT Machine Learning Server vorbereiten [▶ 20].

Der bereitgestellte Pointer auf die Outputdaten muss valide sein und auf eine Instanz eines Arrays von Output-Datentypen gemäß der erstellten PlcOpenXml zeigen. Nach erfolgreichem Abschluss der asynchronen Inferenz sind die Daten im übergeben Output-Speicherbereiches gültig und zur Weiterverarbeitung freigegeben.

Siehe auch KI-Modell ausführen [▶ 25].

	Parameter	Typ	Default	Beschreibung
INPUT	pDataIn	PVOID		Pointer auf die Instanz eines Arrays der Eingangsdatentypen
INPUT	nDataInSize	UDINT	0	Größe des Eingangsdatentypen (Größe eines Elements, nicht des Arrays)
INPUT	nBatchSize	UINT		Größe des Batch
INPUT	pDataOut	PVOID		Pointer auf die Instanz eines Ausgangsdatentype n

	Parameter	Typ	Default	Beschreibung
INPUT	pDataOutSize	UDINT		Größe des Ausgangsdatentype n
INPUT	nTimeout	ULINT		Anzahl der SPS-Task-Zyklen, bevor der Zeitüberschreitungsf ehler zurückgegeben wird.
INPUT	nPriority	UDINT	0	Priorität der Anfrage. Größer bedeutet höhere Priorität.
OUTPUT	PredictBatched	BOOL		Rückgabewert. TRUE, sobald das Ergebnis des asynchronen Aufrufs vorliegt. Das Ergebnis des Aufrufs kann dann mit den Properties 'bError' und 'nErrorCode' überprüft werden.

6.1.1.2 Synchrone Methoden

6.1.1.2.1 GetCustomAttribute_array

	Parameter	Typ	Beschreibung
INPUT	sCustomAttributeName	T_MaxString	Name des Custom Attribute
INPUT	fmtAttributeDataType	Reference to ETcMllDataType	Datenformat des Custom Attribute
INPUT	pDataBuffer	PVOID	Zieldatenpuffer, in den das benutzerdefinierte Attribut kopiert wird.
INPUT	nDataBufferLen	UDINT	Länge des Zieldatenpuffers in Bytes
INPUT	nArrayLength	Reference To UDINT	Anzahl der Datentyp-Elemente (d. h. Anzahl der fp32-Werte), die in dem benutzerdefinierten Attribut gefunden wurden.
INPUT	pnBytesWritten	Pointer To UDINT	Gibt die Anzahl der Bytes zurück, die in den Zielpuffer geschrieben wurden.
OUTPUT	GetCustomAttribute_array	BOOL	TRUE, wenn in der Methode ein Fehler aufgetreten ist.

6.1.1.2.2 GetCustomAttribute_bool

	Parameter	Typ	Beschreibung
INPUT	sCustomAttributeName	T_MaxString	Name des Custom Attribute

	Parameter	Typ	Beschreibung
INPUT	nDataOut	Reference To BOOL	Ausgabewert Custom Attribute vom Typ Bool
OUTPUT	GetCustomAttribute_bool	BOOL	TRUE, wenn in der Methode ein Fehler aufgetreten ist.

6.1.1.2.3 GetCustomAttribute_fp64

	Parameter	Typ	Beschreibung
INPUT	sCustomAttributeName	T_MaxString	Name des Custom Attribute
INPUT	nDataOut	Reference To LREAL	Ausgabewert des Custom Attribute fp64
OUTPUT	GetCustomAttribute_fp64	BOOL	TRUE, wenn in der Methode ein Fehler aufgetreten ist.

6.1.1.2.4 GetCustomAttribute_int64

	Parameter	Typ	Beschreibung
INPUT	sCustomAttributeName	T_MaxString	Name des Custom Attribute
INPUT	nDataOut	Reference To LINT	Ausgabewert des Custom Attribute int64
OUTPUT	GetCustomAttribute_int64	BOOL	TRUE, wenn in der Methode ein Fehler aufgetreten ist.

6.1.1.2.5 GetCustomAttribute_str

	Parameter	Typ	Beschreibung
INPUT	sCustomAttributeName	T_MaxString	Name des Custom Attribute
INPUT	nDataOut	Reference To T_MaxString	Ausgabewert des Custom Attribute vom Typ String
OUTPUT	pnStringLen	Pointer To UDINT	(Optional) Tatsächliche Länge des String-Attributs
OUTPUT	GetCustomAttribute_string	BOOL	TRUE, wenn in der Methode ein Fehler aufgetreten ist.

6.2 Datentypen

6.2.1 E_ExecutionProvider

Enum beschreibt alle unterstützten Ausführungsmodi des TcMlServer.

Name	Typ	Wert	Beschreibung
CPU	USINT	0	Ausführung auf CPU
CUDA	USINT	1	Ausführung auf CUDA-fähigen NVIDIA GPUs

6.2.2 ST_PredictionParameter

Konfigurationsoptionen für eine Inferenzsitzung auf dem TcMlServer.

Name	Typ	Default	Beschreibung
sMlModelFilePath	STRING(255)		Fullpath zum erstellten JSON-File (siehe Model Manager [▶ 28])
eExecutionProvider	E_ExecutionProvider [▶ 41]	ExecutionProvider.CPU	Unter sMlModelFilePath benanntes KI-Modell soll auf dieser spezifizierten Hardware ausgeführt werden.
nDeviceId	UDINT	0	Index des gewünschten GPU-Geräts bei Verwendung des „CUDA“-ExecutionProviders. Der Index entspricht dem CUDA Compute Index und ist nur für IPCs mit mehreren GPUs relevant.
bExclusiveSession	BOOL	TRUE	Bestimmt die Exklusivität der Inferenzsitzung, die für die FB-Instanz auf dem TcMlServer erzeugt wird. Wenn TRUE, erzeugt der TcMlServer eine exklusive Sitzung, die für zustandsabhängige Modelle (z. B. RNNs) notwendig ist, um Interferenzen zu vermeiden. Bei FALSE kann die Sitzung mit anderen FB-Instanzen, die dieselbe Konfiguration anfordern, geteilt werden, was die Speicherbelastung verringern kann.
nSessionTimeout	ULINT	72	Dauer der Inaktivität in Stunden, nach der die Sitzung des FB beim TcMlServer abläuft. Der Server gibt dann die allokierten Ressourcen des betreffenden Clients wieder frei.
sMlSvrNetId	T_AmsSvrNetId	'127.0.0.1.1.1'	AMS Net Id des Geräts auf dem der TcMlServer Service erreichbar ist. Default ist „lokal“.

7 Beispiele

7.1 KI-basierte Bildverarbeitung

In diesem Beispiel wird demonstriert, wie Sie:

- Mit TwinCAT Vision Bilddaten von der lokalen Festplatte laden und in der SPS verfügbar machen.
- Bilder mit der TwinCAT Vision Bibliothek vorverarbeiten.
- Mit dem FB_MlSvrPrediction eine Session auf dem lokal installierten TwinCAT Machine Learning Server starten und ein KI-Modell (Klassifikationsmodell) laden.
- Die Inferenz auf dem TwinCAT Machine Learning Server ausführen.
- Das Ergebnis in der SPS weiterverwenden.

Download und Übersicht der Dateien

Das Projekt können Sie hier herunterladen: https://infosys.beckhoff.com/content/1031/TF3820_TC3_Machine_Learning_Server/Resources/17328164363.zip

- In dem ZIP befindet sich ein **tnzip**, welches Sie im TwinCAT XAE über *File* > *Open* > *Open Solution from Archive…* öffnen können.
- Im Ordner models befinden sich eine **ONNX** sowie die bereits erstellen **JSON** und PlcOpenXml.
- Im Ordner dataset befinden sich **Beispielbilder**, welche verarbeitet werden sollen.

Voraussetzungen

Installieren Sie folgende Workloads:

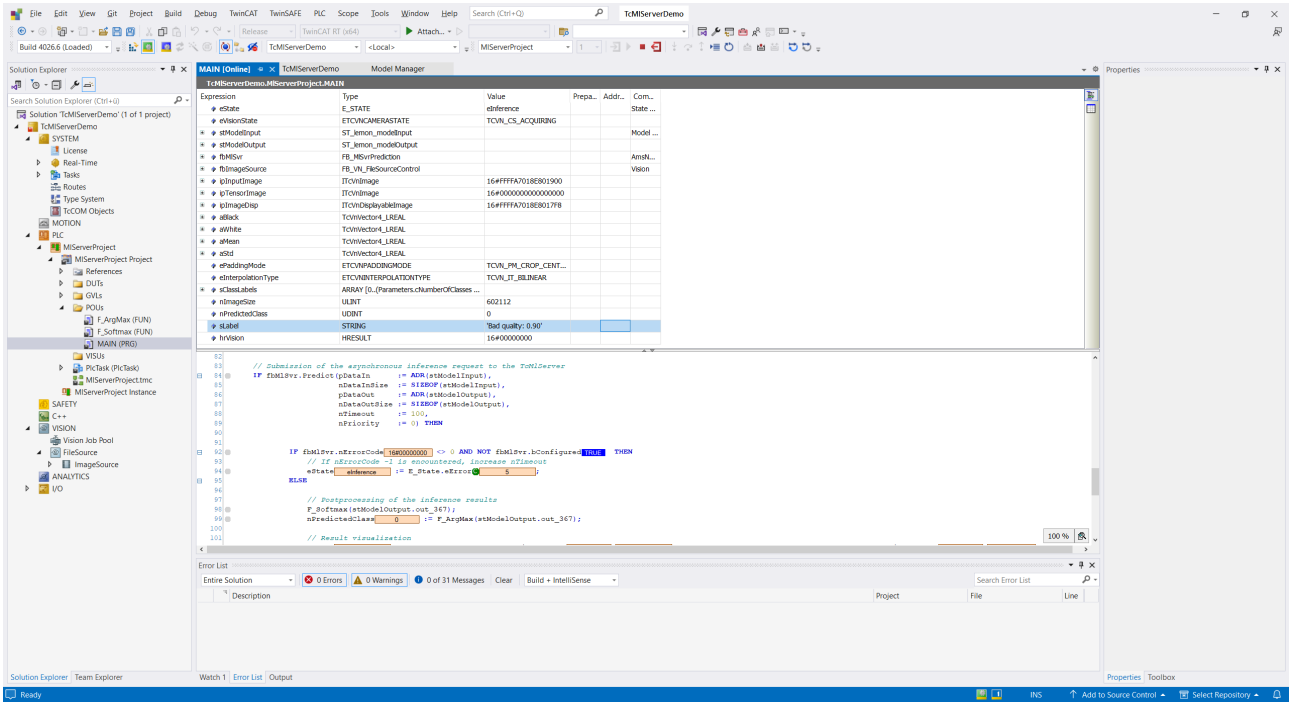
- TwinCAT Standard
- TF3820 | TwinCAT Machine Learning Server
- TF3830 | TwinCAT Machine Learning Server Client
- TF7xxx | TwinCAT 3 Vision

Einrichtung des Projekts

- Öffnen Sie das tnzip und speichern Sie ihr Projekt.
- Richten Sie die FileSpource ein:
  - Selektieren Sie im System Manager das Tree Item *Vision* > *FileSource* > *ImageSource*
  - Fügen Sie die Bilder aus dem Ordner models hinzu.
  - Setzen Sie die Cycle Time der ImageSource auf 100 ms.
- Benennen Sie die FullPath zur Modell-JSON in Zeile 9 der MAIN. Die ONNX muss im selben Ordner liegen.
- Stellen Sie sicher, dass alle Softwarelizenzen zumindest in der 7-Tage-Testlizenz verfügbar sind:
  - TC1200 TwinCAT PLC
  - TF3820 TwinCAT Machine Learning Server
  - TF7100 TwinCAT Vision Base

Ausführen des Projekts

Starten Sie die Anwendung mit Activate Configuration auf Ihrem Zielsystem. Wenn alle Punkte korrekt eingerichtet sind, sollte der `eState` nach kurzer Zeit auf `eInference` stehen und die Variable `sLabel` das Ergebnis der aktuellen Inferenz anzeigen.



Sollte ein Fehler aufgetreten sein, steht der `eState` auf Error. Sie können dann die Instanz fbMlSvr aufklappen und den Error Code auslesen. Nutzen Sie die Tabelle der Error Codes [▶ 49], um das Problem einzugrenzen.

Ausschnitte aus dem SPS-Programm

Deklaration

In der Deklaration sind die wesentlichen Punkte, die den Umgang mit dem TwinCAT Machine Learning Server betreffen, die Eingangs- und Ausgangsdatentypen des KI-Modells sowie die Instanz der Clients zum Machine Learning Server.

```
stModelInput : ST_lemon_modelInput;      //model input datatype, imported via PlcOpenXml
stModelOutput : ST_lemon_modelOutput;    //model output datatype, imported via PlcOpenXml
fbMlSvr : FB_MlSvrPrediction();   // Instance of Client to TcMlServer
```

Die Datentypen sind bereits über die PlcOpenXml (siehe Ordner models) in das TwinCAT Projekt eingelesen. Die Beschreibung finden Sie im Ordner DUTs.

Konfiguration der Session

In diesem Beispiel eröffnet der Client im State E_State.eMlSvrConfiguration eine Session auf dem TwinCAT Maschine Learning Server. Dabei wird spezifiziert, auf welchem System der Server erreichbar ist, welches Modell in der Session geladen und auf welcher Hardware das Modell ausgeführt werden soll.

```
fbMlSvr.stPredictionParameter.sMlModelFilePath := 'C:
\models\lemon_model.json';                              // fullpath to model
fbMlSvr.stPredictionParameter.sMlSvrNetId := '127.0.0.1.1.1';                                    //
Server on local system
fbMlSvr.stPredictionParameter.eExecutionProvider := E_EXECUTIONPROVIDER.CPU;
       // CPU execution

// Submit configuration request to the TcMlServer
// Provide a generous nTimeout, as the configuration can take a substantial amount of time
IF fbMlSvr.Configure(nTimeout := 1000, nPriority:=0) THEN
   IF fbMlSvr.nErrorCode <> 0 THEN
      // If nErrorCode -1 is encountered, increase nTimeout
      eState := E_State.eError;
   ELSE
      eState := E_State.eImageAcquisition;
   END_IF
END_IF
```

Der Aufruf der Methode `Configure()` sendet die Anfrage zur Eröffnung einer Session an den Server ab. Der Aufruf ist asynchron zur SPS-Task und wird mit einem `TRUE` quittiert, wenn die Einrichtung der Session erfolgreich abgeschlossen ist.

Ausführen des Modells

Im State `E_State.eInference` wird der `Predict`-Aufruf an den Machine Learning Server gesendet. Auch dieser Aufruf ist asynchron zur SPS-Task. Die Methode liefert ein TRUE zurück, wenn das Ergebnis vorliegt.

In diesem Beispiel wird vor dem Inferenz-Aufruf das Bild vom Typ `ITcVnImage` mit der Funktion `F_VN_ExportImage` in den Modelleingangsdatentyp kopiert.

```
F_VN_ExportImage(ipTensorImage, ADR(stModelInput.in_input1), nImageSize, hrVision);
// Submission of the asynchronous inference request to the TcMlServer
IF fbMlSvr.Predict(pDataIn       := ADR(stModelInput),
                nDataInSize   := SIZEOF(stModelInput),
                pDataOut    := ADR(stModelOutput),
                nDataOutSize := SIZEOF(stModelOutput),
                nTimeout    := 100,
                nPriority    := 0) THEN
      IF fbMlSvr.nErrorCode <> 0 AND NOT fbMlSvr.bConfigured  THEN
         // If nErrorCode -1 is encountered, increase nTimeout
          eState := E_State.eError;
      ELSE

         // Postprocessing of the inference results
         F_Softmax(stModelOutput.out_367);
         nPredictedClass := F_ArgMax(stModelOutput.out_367);
```

Das Ergebnis der Inferenz kann nach erfolgreicher Fehlerprüfung verwendet werden.

7.2 Custom Attributes

Mit den Custom Attributes ist es möglich, dem KI-Modell Meta-Informationen mitzugeben, welche zur Laufzeit in der SPS berücksichtigt werden sollen. Das Metadaten-Konzept existiert ähnlich auch in der ONNX, diese können jedoch nicht zur Laufzeit in der SPS ausgewertet werden.

Prinzipiell können Sie beide Metadatenbereiche gleichzeitig nutzen. Unterscheiden Sie dabei in Informationen, die zur Laufzeit in der SPS benötigt werden (Custom Attributes Bereich) und Informationen, die der SPS-Programmierer nur zur Engineering Phase benötigt (ONNX Metadaten oder Custom Attributes).

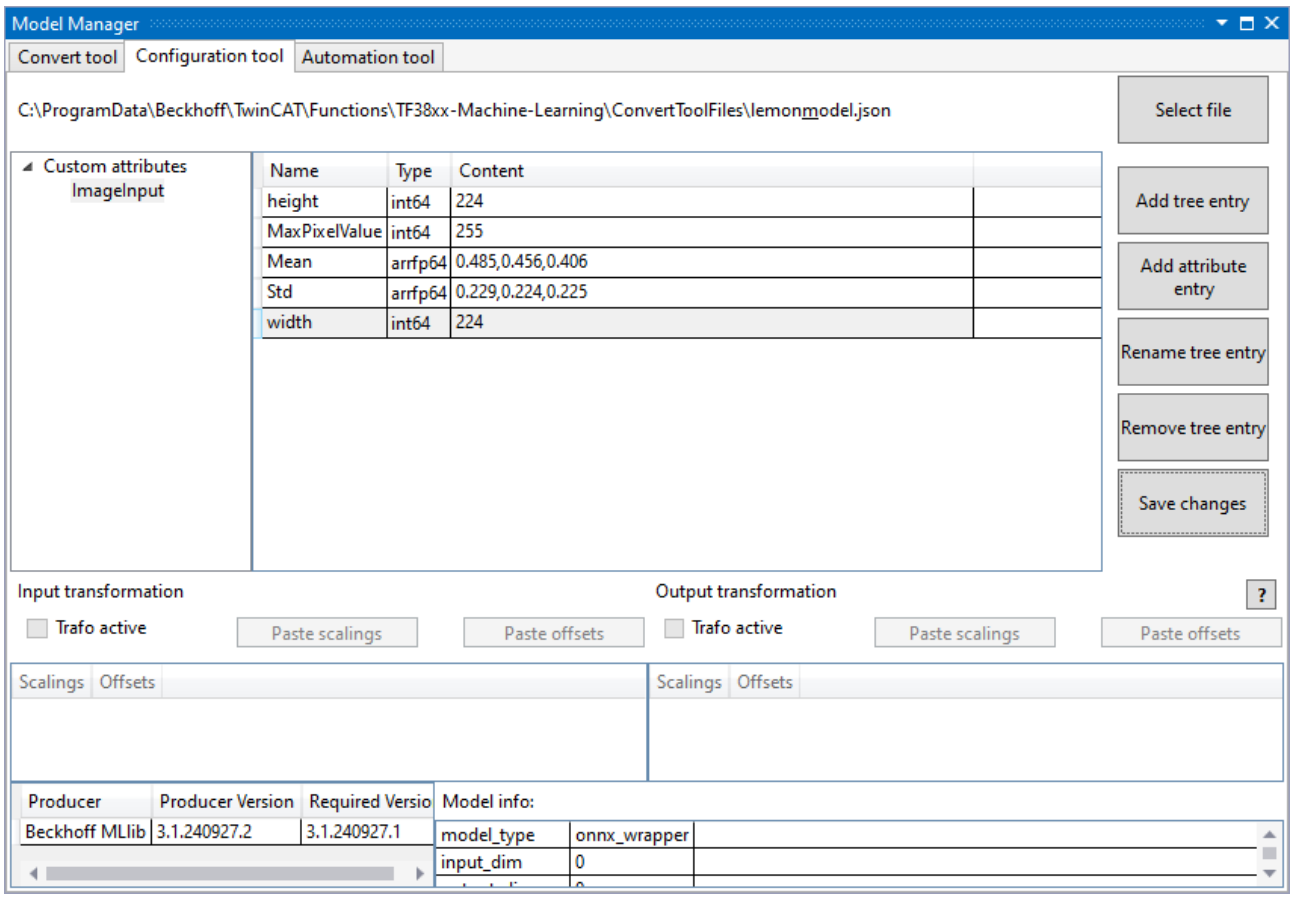
Ziel ist es in jedem Fall, dass der Ersteller einer ONNX genügend Informationen an den Konsumenten der ONNX weitergeben kann, damit der Konsument das KI-Modell korrekt und effizient nutzen kann.

Custom Attributes zur Bildvorverarbeitung

Im Folgenden wird das Beispiel KI-basierte Bildverarbeitung [▶ 43] erweitert. Es werden Informationen über das Eingangsbild hinzugefügt. Ziel ist es, die Bildvorverarbeitung entsprechend den Metainformationen anzupassen.

Umsetzung im TwinCAT Machine Learning Model Manager

In der graphischen Umgebung kann das Configuration Tool genutzt werden, um die Custom Attributes einzupflegen.

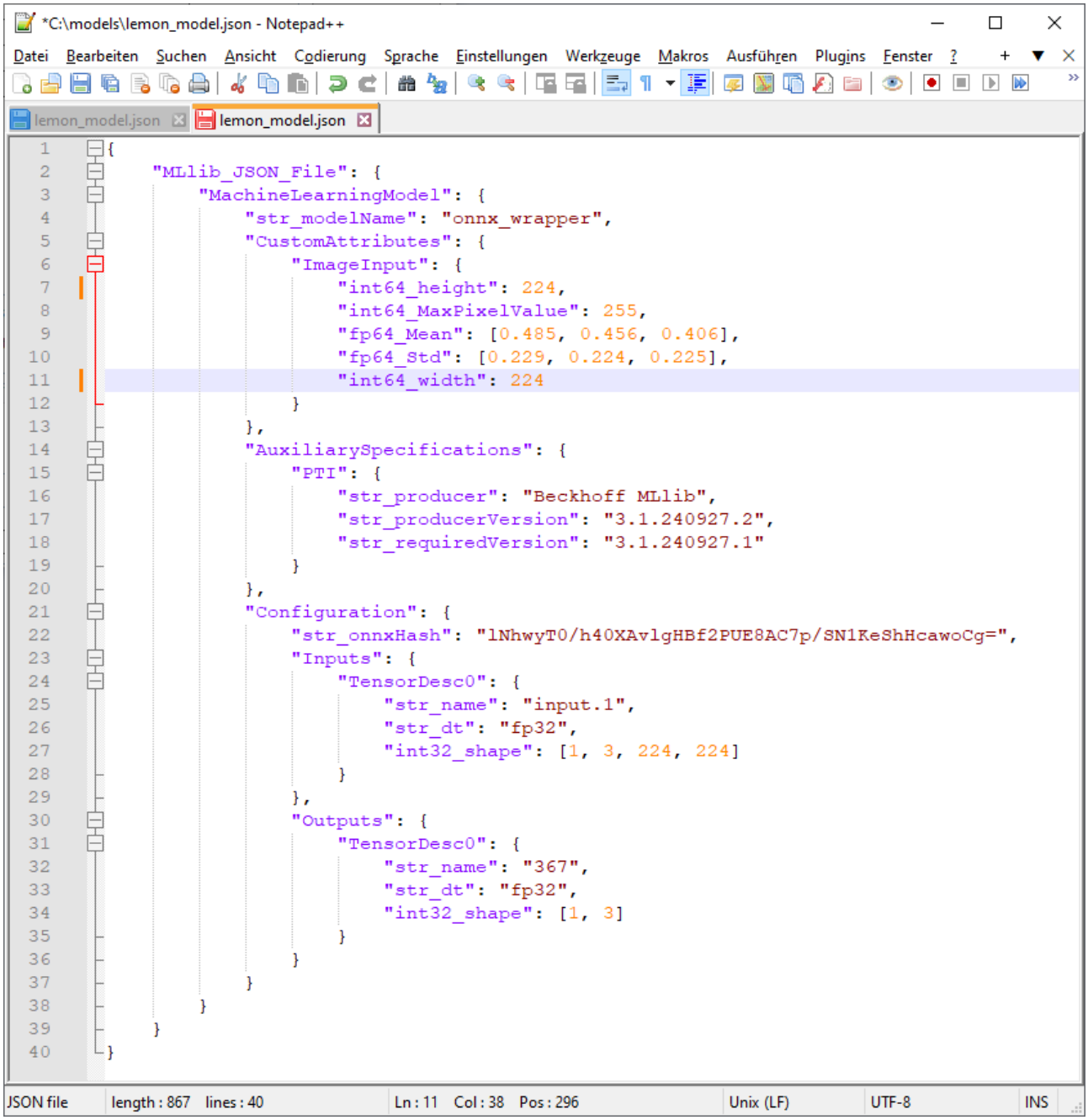


Umsetzung in Python

Oft ist es vom Workflow her geschickter, alle notwendigen Custom Attributes direkt in Python im Anschluss an das Modelltraining einzutragen.

```
new_ca = { 'ImageInput' : {'nwidth' : 244, 'nheight' : 244, 'nMaxPixelValue' : 255, 'fMean' : [0.485
, 0.456, 0.40], 'fStd' : [0.229, 0.224, 0.225]} }
tb.modify_ca("C:\\models\\lemon_model.json", "C:\\models\\lemon_model.json", new_ca)
```

Das Ergebnis kann in der JSON im Klartext nachvollzogen werden.



Auslesen der Custom Attributes in TwinCAT

Das SPS-Projekt aus dem Beispiel wird im Folgenden erweitert.

Nach erfolgreicher Konfiguration einer Session auf dem TwinCAT Machine Learning Server werden die abgelegten Custom Attributes ausgelesen.

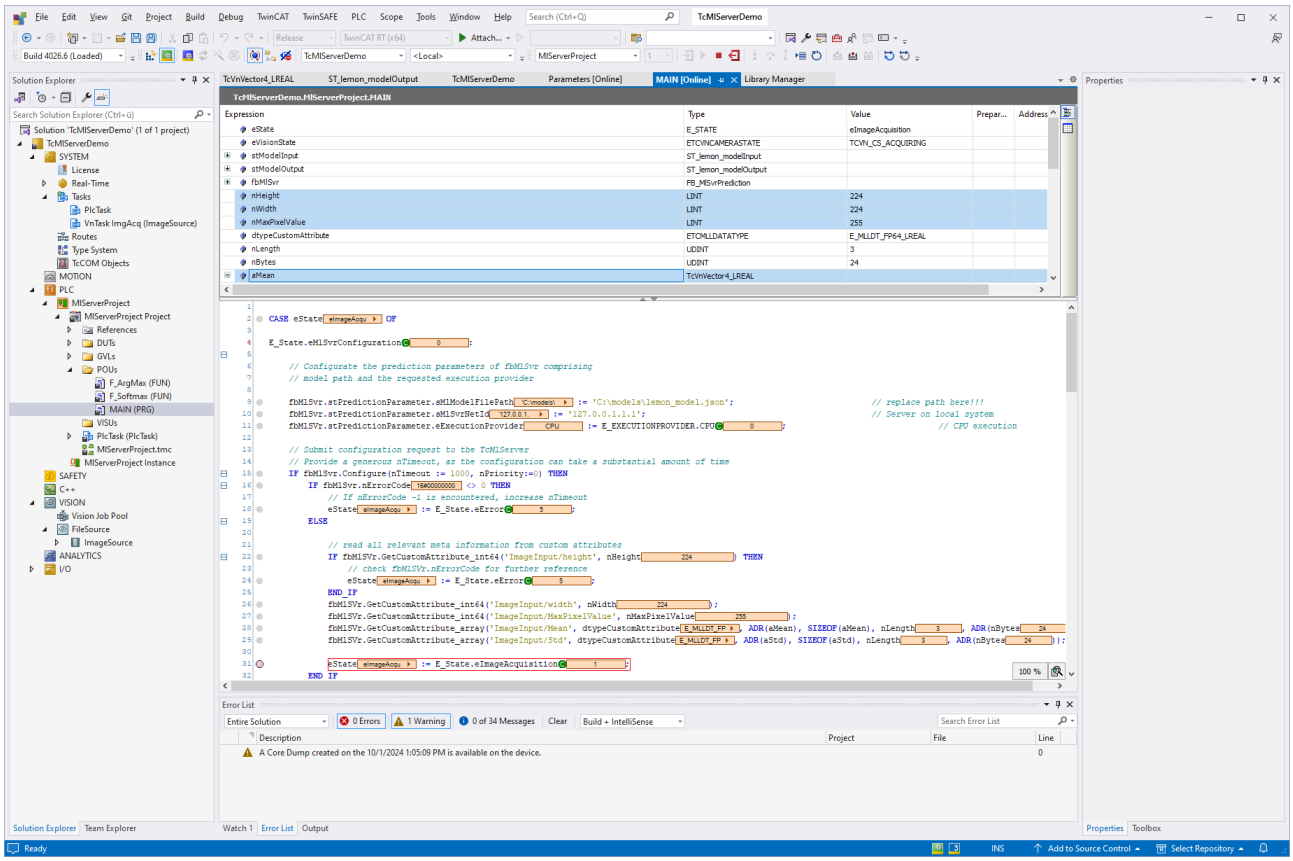
```
IF fbMlSvr.Configure(nTimeout := 1000, nPriority:=0) THEN
   IF fbMlSvr.nErrorCode <> 0 THEN
      // If nErrorCode -1 is encountered, increase nTimeout
      eState := E_State.eError;
   ELSE

      // read all relevant meta information from custom attributes
      IF fbMlSVr.GetCustomAttribute_int64('ImageInput/height', nHeight) THEN
         // check fbMlSVr.nErrorCode for further reference
         eState := E_State.eError;
      END_IF
      fbMlSVr.GetCustomAttribute_int64('ImageInput/width', nWidth);
      fbMlSVr.GetCustomAttribute_int64('ImageInput/MaxPixelValue', nMaxPixelValue);
      fbMlSVr.GetCustomAttribute_array('ImageInput/
Mean', dtypeCustomAttribute, ADR(aMean), SIZEOF(aMean), nLength, ADR(nBytes));
      fbMlSVr.GetCustomAttribute_array('ImageInput/
```

```
Std', dtypeCustomAttribute, ADR(aStd), SIZEOF(aStd), nLength, ADR(nBytes));

      eState := E_State.eImageAcquisition;
   END_IF
END_IF
```

Danach werden die in SPS-Variablen gespeicherten Custom Attributes in der Vorverarbeitungs-Pipeline genutzt. Hier sind im Wesentlichen nur die notwendigen Typ-Konvertierungen zu beachten.

```
// Adjust the dimensions of the input image to match the model input requirements
hrVision := F_VN_ResizeImageExp(ipInputImage, ipTensorImage, LINT_TO_UDINT(nWidth), LINT_TO_UDINT(nH
eight), eInterpolationType, ePaddingMode, aBlack, hrVision);
// Convert the image to type REAL and scale to the range [0.0, 1.0]
hrVision := F_VN_ConvertElementTypeExp(ipTensorImage, ipTensorImage, TCVN_ET_REAL, 1.0 / LINT_TO_REA
L(nMaxPixelValue), 0, hrVision);
// Normalization
hrVision := F_VN_SubtractVectorFromImage(ipTensorImage, aMean, ipTensorImage,hrVision);
hrVision := F_VN_DivideImageByVector(ipTensorImage, aStd, ipTensorImage, hrVision);
```

Nach Aktivierung der Konfiguration kann im Online View kontrolliert werden, ob die Werte richtig übernommen wurden.



8 Anhang

8.1 Error Codes

Error Code (dec)	Error Code (hex)	Description	Hints
0	0000	Operation successful	-
-1	FFFF	ML server timeout	Increase the timeout argument of your request. Check, that the TcMlServer service is running.
-3	FFFD	ML server malformed data	-
-5	FFFB	PLC protocol error	Make sure to call the methods of the FB in a canonical order, i.e. make sure the FB is configured before making inference calls.
-8	FFF8	Invalid AMS Net ID	Check the syntactical correctness of the provided AmsNetId.
-9	FFF7	Input pointer is null	Check the validity of the input data pointer provided to the predict methods.
-11	FFF5	Output pointer is null	Check the validity of the output data pointer provided to the predict methods.
-13	FFF3	Batch size is zero	Provide a valid batch size (>= 1).
-15	FFF1	Input size is zero	Provide a valid input data size (>0).
-17	FFEF	Output size is zero	Check your installation of TF3830.
-18	FFEE	Driver instantiation failed	Check your installation of TF3830.
-20	FFEC	Driver state propagation to state OP failed	Check the validity and integrity of your model file (existing path, valid file format).
-22	FFEA	Driver loading failed	Check your installation of TF3830.
-24	FFE8	Driver state depropagation failed	Check your installation of TF3830.
-26	FFE6	Driver parameter configuration failed	TwinCAT-internal error
-28	FFE4	Could not instantiate file accessor	TwinCAT-internal error
-30	FFE2	Could not propagate file accessor state	TwinCAT-internal error
-32	FFE0	Could not access model file	Check the validity of your model file (existing path, valid file format).
-34	FFDE	Mismatch in read model file bytes	TwinCAT-internal error
-36	FFDC	Could not parse model file	Check the integrity of your model file.
-38	FFDA	String buffer overflow	Check the integrity of your model file, especially the length of the contained model hash.
-40	FFD8	Could not find model config in file	Check the integrity of your model file, especially the defined model configuration.
-42	FFD6	Could not retrieve model attributes	Check the integrity of your model file, especially the availability of a model hash.
-43	FFD5	ADS server connection error	ADS messages could not be sent. Check integrity of AmsRouter.

Error Code (dec)	Error Code (hex)	Description	Hints
-44	FFD4	Engine mismatch	The model file you intended to load is not designated for the TcMlServer.
-45	FFD3	Invalid CST attribute name	Check the name of the attribute in your model file or your query. The name must not be empty.
-47	FFD1	CST attribute retrieval failed	The queried attribute could not be found. Check the queried attribute name in query and model file.
-49	FFCF	CST attribute invalid buffer	Check the destination pointer provided to the retrieval function. Must not be a null pointer.
-50	FFCE	Versioned model could not be resolved	Check the valditiy of the model version you are querying.
-51	FFCD	If a PLC online change is detected, a running predict is discarded with the error	
-101	FF9B	Invalid PLC request variant	Internal error
-102	FF9A	Invalid device index	Verify the device index provided in prediction parameters of the FB.
-103	FF99	Invalid request data	Internal error
-201	FF37	Invalid data	Internal error
-202	FF36	Invalid model type	Requested tensor of unsupported datatype. See log files.
-206	FF32	Invalid model	The provided model file is not supported. check especially, that no dimension parameters are left except for an optional batch dimension.
-300	FED4	Invalid execution provider	The requested execution provider is invalid. Make sure to select an EP from the enum E_ExecultionProvider.
-302	FED2	Unsupported execution provider	The requested execution provder is valid but not supported on your system (for instance due to a lack of GPU support).
-400	FE70	Session config file not found	Internal error
-402	FE6E	Environment config file not found	Internal error
-404	FE6C	Run config file not found	Internal error
-406	FE6A	Server config file not found	Internal error
-408	FE68	Device config files not found	Internal error
-500	FE0C	Invalid session config file	Internal error
-502	FE0A	Invalid environment config file	Internal error
-504	FE08	Invalid run config file	Internal error
-506	FE06	Invalid server config file	Internal error
-508	FE04	Invalid device config files	Internal error
-600	FDA8	Targeted inference provider unavailable	Internal error
-602	FDA6	Inference provider session timeout	Increase the session timeout in the prediction parameters.
-701	FD43	Dynamic loaded library not found	Internal error

Error Code (dec)	Error Code (hex)	Description	Hints
-703	FD41	Symbol not loadable from dynamic library	Internal error
-801	FCDF	Unknown job execution error	Internal error
-803	FCDD	Unknown error	Internal error
-901	FC7B	Failed backend execution	Internal error
-1100	FBB4	License server connection error	Internal licensing error
-1102	FBB2	License server ADS error	Internal licensing error
-1104	FBB0	License ADS error	Internal licensing error
-1106	FBAE	License invalid	Make sure, that a license TF3820 is available.
-1108	FBAC	License invalid unknown	Internal licensing error
-1201	FB4F	Expected tensor collection type	Make sure, that only types defined in the PLCopen files generated by the Beckhoff model management products is used. Do not manipulate the contained header
-1203	FB4D	Mismatching tensor collection hashes	Make sure, that only types defined in the PLCopen files generated by the Beckhoff model management products is used. Do not manipulate the contained header
-1205	FB4B	Invalid tensor collection header	Make sure, that only types defined in the PLCopen files generated by the Beckhoff model management products is used. Do not manipulate the contained header
-1207	FB49	Tensor collection offset feature not supported	Make sure, that only types defined in the PLCopen files generated by the Beckhoff model management products is used. Do not manipulate the contained header
-1209	FB47	Internal allocation special type header	Internal error
-1300	FAEC	Server startup cannot build log directory	No Logs directory available and could not be generated.
-1801	F8F7	NVIDIA NVML library function failure	Make sure the Nvidia nvml.dll is avilable.
-1803	F8F5	NVIDIA NVML library extraction failure	Make sure the Nvidia nvml.dll is avilable.
-1901	F893	NVIDIA CUDA library function failure	Make sure the Nvidia cuda.dll is available.
-1903	F891	NVIDIA CUDA library extraction failure	Make sure the Nvidia cuda.dll is available.
-2000	F830	Model registry file storage error	Internal error
-2002	F82E	Model registry file deletion error	Internal error
-2003	F82D	Model registry invalid reference	Internal error
-2004	F82C	Model registry entry failed to load model	Internal error
-2006	F82A	Model registry inconsistent model hashes	Internal error
-2008	F828	ADS could not open remote file	Make sure, that an .onnx file with the same name as your .json model description file is located in the same directory.

Error Code (dec)	Error Code (hex)	Description	Hints
-2010	F826	ADS could not retrieve size of remote file	Internal error
-2012	F824	ADS could not load remote file	Make sure, that your AMS router has sufficient memory for your .onnx file.
-2014	F822	ADS could not close remote file	Internal error
-2018	F81E	ADS could not open client port	Make sure your TwinCAT installation is valid.
-2201	F767	Buffer base is null	Internal error
-2203	F765	Not enough memory for head	Internal error
-2205	F763	Buffer out of memory	Internal error
-2207	F761	Buffer cannot read from null	Internal error
-2209	F75F	Buffer cannot write to null	Internal error
-2211	F75D	Buffer corrupted with invalid string	Internal error
-2020	F81C	Insufficient AMS Router Memory on Client	Increase Router Memory on the Client system
-2022	F81A	Insufficient AMS Router Memory on Server	Increase Router Memory on the Server system
-2024	F818	Could not read router memory on client	Internal error
-2026	F816	Could not read router memory on server	Internal error
-2301	F703	Invalid CUDA setup detected	Check your CUDA and cuDNN installation.

8.2 Log files

TwinCAT Machine Learing Model Manager Logs: C:\ProgramData\Beckhoff\TwinCAT\Functions\TF38xx-Machine-Learning\Logs

TwinCAT Machine Learning Server Logs: C:\ProgramData\Beckhoff\TwinCAT\Functions\TF38xx-Machine-Learning\TcMlServer\Logs

8.3 Third-party components

This software contains third-party components.
Please refer to the license file provided in the following folder for further information:
C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAR-MlServer
C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAE-ModelManagerCore

8.4 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/TF3820

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com