User's Manual

e-AI Translator V2.2.0

User's Manual

Renesas Electronics
www.renesas.com

Rev.9.00 Mar 2022

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

Page

Terminology

Some specific words used in this user's manual are defined below.

e-AI

An abbreviation of Embedded Artificial Intelligence.

e-AI Translator

A tool that converts and imports the inference processing of neural network models which have been trained in an open-source deep-learning framework, PyTorch or Keras or TensorFlow (tf.keras) or TensorFlow Lite, into files of code for the integrated development environment for MCUs and MPUs. The e-AI Translator operates as a plug-in for the e^2 studio integrated development environment for Renesas MCUs and MPUs.

Neural network

A model that has been made to provide computers with training abilities by representing the mechanisms of the neural circuits of a human brain.

Deep-learning framework

Software for defining and training neural network models.

Host machine

The personal computer on which the e-AI Translator runs.

User system

An application system in which an MCU or MPU is used, and that is run by the user.

User program

An application program that runs on the MCU or MPU.

1. Overview

The e-AI Translator is a tool that converts and imports the inference processing of neural network models which have been trained in an open-source deep-learning framework, PyTorch or Keras or TensorFlow(tf.keras) or TensorFlow Lite, into files of source code for the e^2 studio integrated development environment for MCUs and MPUs.

This document describes the methods of installation, usage, and points for caution of the e-AI Translator.

The e-AI Translator operates as a plug-in for the e^2 studio integrated development environment for Renesas MCUs and MPUs.

1.1. Operating Environment

Table 1-1 shows the operating environment of the e-AI Translator and required software to be installed.

Table 1-1 Operating Environment

Item	Name of Software, Supported Versions, etc.
Supported deep-learning frameworks[Note1]	PyTorch[Note2], Keras[Note3], TensorFlow(tf.keras API)[Note4], TensorFlow Lite[Note5]
Operating environment	The 64bit edition of Windows 8.1, Windows 10
Required software to be installed	Python 3.7.7
	The following packages from Python Package need to be installed. [Note6] Torch, TorchVision, TorchSummary, onnx, TensorFlow, progressbar33, prettytable, h5py, pycryptodome, configparser, psutil, Keras, tflite
	Microsoft Visual C++ 2015 Redistributable
	e^2 studio 2020-10 or later version Flexible Software Package 3.5 or later version
Supported devices[Note7]	RL78 family, RX family, RA family, RE family, RZ family, and Renesas Synergy™ microcontrollers

Notes: 1. This version of e-AI Translator does not support Caffe and TensorFlow (tf.nn API and tf.layers API). If these frameworks are required, use e-AI Translator V1.6.0.
2. Translation operation has been checked by using trained model of PyTorch 1.5.1.
3. Translation operation has been checked by using trained model of standalone Keras 2.4.3.
4. Translation operation has been checked by using trained model of TensorFlow-backend Keras 2.4.0-tf which is included TensorFlow 2.4.0.
5. Translation operation has been checked by using 8bit quantized model of TFLiteConverter which is included in TensorFlow 2.4.0. Refer to "3.6. How to create/save TensorFlow Lite model file" for more detail about 8bit quantization.
6. For details on installing from Python Package, refer to section "2.4. Installing the Required Python Packages".
7. The usage of ROM and RAM increases with the size of the neural network. We recommend using a device with a large memory capacity.

1.2. Types of Convertible Neural Networks

Table 1-2 shows the types of neural networks that can be converted by the e-AI Translator in its current version.

Further types of neural network will be gradually added in future updates of the plug-in.

Note: Non-supported neural networks cannot be converted correctly. Be sure to confirm that the neural network to be converted only consists of supported types.

Table 1-2 Types of Convertible Neural Networks

Item	Types to be Supported
Algorithms*	Convolutional neural network (CNN), Auto encoder, etc.
Convolution layer	Convolution, Deconvolution (Transposed convolution), Depthwise Convolution
Pooling layer	Max pooling, Average pooling, Global max pooling, Global average pooling
Fully connected layer	Fully connected (Inner product)
Normalization layer	Batch Normalization
Activation functions	ReLU (Rectified Linear Unit), Sigmoid, Tanh, ReLU6, Leaky ReLU, Clip, Hard sigmoid, Softsign, Softplus
Classification function	Softmax

Note: The current version does not support recurrent neural networks (RNNs) and the neural network which has branch structures such as ResNet and MobileNet.

1.3. Supported API list

Table 1-3 shows the supported API of PyTorch, API of Keras/TensorFlow(tf.keras), TensorFlow Lite 8bit quantized model that can be converted by the e-AI Translator in its current version.
Further APIs and Layer types will be gradually added in future updates of the plug-in.

Table 1-3 List of supported APIs (1/2)

PyTorch API	Keras/TensorFlow(tf.keras) API	TensorFlow Lite 8bit quantized Model
torch.nn.Conv2d[Note]	keras.layers.Conv2D tf.keras.layers.Conv2D	Support
torch.nn.ConvTranspose2d	keras.layers.Conv2DTranspose tf.keras.layers.Conv2DTranspose	Support
torch.nn.Conv2d[Note]	keras.layers.DepthwiseConv2D tf.keras.layers.DepthwiseConv2D	Support
torch.nn.MaxPool2d	keras.layers.MaxPooling2D tf.keras.layers.MaxPool2D	Support
torch.nn.AvgPool2d	keras.layers.AveragePooling2D tf.keras.layers.AveragePooling2D	Support
torch.nn.ReLU	keras.activations.relu tf.keras.activations.relu keras.layers.relu tf.keras.layers.relu keras.layers.Activation("relu") tf.keras.layers.Activation("relu")	Support
torch.nn.TanH	keras.activations.tanh tf.keras.activations.tanh keras.layers.Activation("tanh") tf.keras.layers.Activation("tanh")	Support
torch.nn.Sigmoid	keras.activations.sigmoid tf.keras.activations.sigmoid keras.layers.Activation("sigmoid") tf.keras.layers.Activation("sigmoid")	Support
torch.nn.Softmax	keras.activations.softmax tf.keras.activations.softmax keras.layers.Softmax tf.keras.layers.Softmax keras.layers.Activation("softmax") tf.keras.layers.Activation("softmax")	Support
torch.nn.ReLU6	keras.activations.relu(max_value=6.) tf.keras.activations.relu(max_value=6.) keras.layers.relu(max_value=6.) tf.keras.layers.relu(max_value=6.)	Support

Note: This can be used as convolution operation and depthwise convolution operation by setting of its arguments. But e-AI Translator does not support separable convolution operation.

Table 1-4 List of supported APIs (2/2)

PyTorch API	Keras/TensorFlow(tf.keras) API	TensorFlow Lite 8bit quantized Model
torch.nn.Linear	keras.layers.Dense tf.keras.layers.Dense	Support
torch.add	keras.layers.Add tf.keras.layers.Add	Support
torch.sub torch.Tensor.sub	keras.layers.Subtract tf.keras.layers.Subtract	Support
* (Mutiliplation operator)	keras.layers.Multiply tf. keras.layers.Multiply	Support
torch.view torch.Flatten	keras.layers.Reshape tf.keras.layers.Reshape keras.layers.Flatten tf.keras.layers.Flatten	Support
torch.nn.BatchNorm2D	keras.layers.BatchNormalization tf.keras.layers.BatchNormalization	Support[Note]
torch.cat(axis=1)	keras.layers.concatenate tf.keras.layers.concatenate	Support
torch.clamp	keras.backend.clip tf.keras.backend.clip	-
torch.nn.MaxPool2d torch.AdaptiveMaxPool2d	keras.layers.GlobalMaxPooling2D tf.keras.layers.GlobalMaxPooling2D	Support
torch.nn.AvgPool2d torch.AdaptiveAvgPool2d	keras.layers.GlobalAveragePooling2D tf.keras.layers.GlobalAveragePooling2D	Support
torch.nn.LeakyReLU	keras.layers.LeakyReLU tf.keras.layers.LeakyReLU	Support
torch.nn.Sigmoid	keras.layers.Activation("hard_Sigmoid") tf.keras. Layers.Activation("hard_sigmoid")	-
torch.nn.Softsign	keras.layers.Activation("softsign") tf.keras.layers.Activation("softsign")	-
torch.nn.Softplus	keras. Layers.Activation("softplus") tf.keras. Layers.Activation("softplus")	-
torch.nn.LocalResponseNorm	-	-

Note: Batch normalization layer can be used only when the order of layers is "Convolution - Batch Normalization - ReLU".

Following APIs are only used in the training phase, so these APIs are ignored by translation of e-AI Translator.
- Loss functions
- Dropout functions

1.4. Changes of e-AI Translator from V2.1.0 to V2.2.0

This chapter explains the changes of e-AI Translator from V2.1.0 to V2.2.0.

- Addition of new output format "CMSIS_INT8". This function can be used for RA family and RX family.
  By using this function, following choice of translation is available for TensorFlow Lite 8-bit quantized model.
  - output format "CMSIS_INT8": Translation with prioritizing inference speed.
  - output format "C_INT8": Translation with prioritizing ROM/RAM usage.
  Refer to "3.7. How to use CMSIS_INT8" for more detail, such as usage, difference of "Convert option", and so on.

- Support of TensorFlow Lite model which has a difference between input format and output format. (For example, input: 8-bit signed int, output: 32-bit float)
  Refer to "3.6. How to create/save TensorFlow Lite model file for more detail.

- CC-RX compiler and CC-RL compiler can be used for translated source code from TensorFlow Lite 8-bit quantized model.

- Change of the format of license file.
  (Cannot use the e-AI Translator's license file of older version. Be careful.)

2. Installing the e-AI Translator

This chapter explains how to install the e-AI Translator.

The e-AI Translator is a plug-in for the e^2 studio integrated development environment for Renesas MCUs and MPUs. After installing or if you have already installed the e^2 studio, install the e-AI Translator through the procedure below.

2.1. Installing the e-AI Translator

1. Extract zipped installer file which you have downloaded from our Web site.
2. Double-click on the extracted file "setup.exe".
3. Proceed with installation according to the instructions of the installer.

Note that the destination folder for installation must be the folder where the e^2 studio is installed.

Note: If the installation destination folder for the e^2 studio has not been changed, the folder setting for the e-AI Translator need not be changed from the default.

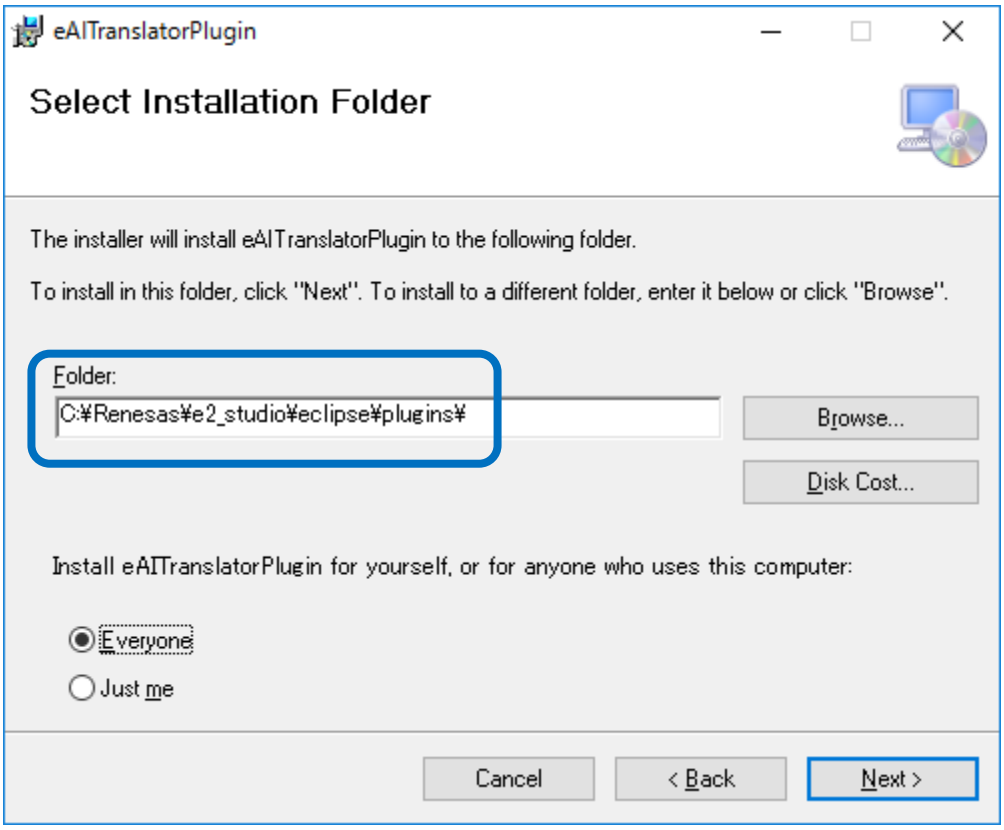

Figure 2.1 Specifying the Destination Folder for Installation

2.2. Copying the License File for the e-AI Translator

Copy the file "license.txt" that you have downloaded from the Web site to the following folder.

C:\Renesas\e2_studio\eclipse\plugins\com.renesas.eaitranslator_2.2.0

Note: If you have changed the destination folder for installation in section 2.1, change the destination folder for copying to match the path you have selected.

2.3. Installing Python 3.7.7

1. Download Python 3.7.7 from the following Web site.
   https://www.python.org/downloads/release/python-377/
   "Windows x86-64 executable installer"
2. Double-click on the installer to start installation.
   Click on "Install Now".
   Proceed with installation according to the instructions of the installer.

   Remark1: Both "checked" and "unchecked" settings are OK about "Add Python 3.7 to PATH".

   Remark2: For the license conditions of Python, read "License.txt" in its installation folder.

   [Installation folder for Python]

   C:\Users*<windows-user-name>*\AppData\Local\Programs\Python\Python37

   Note: "*<windows-user-name>*" is the user name that you use for logging on Windows.

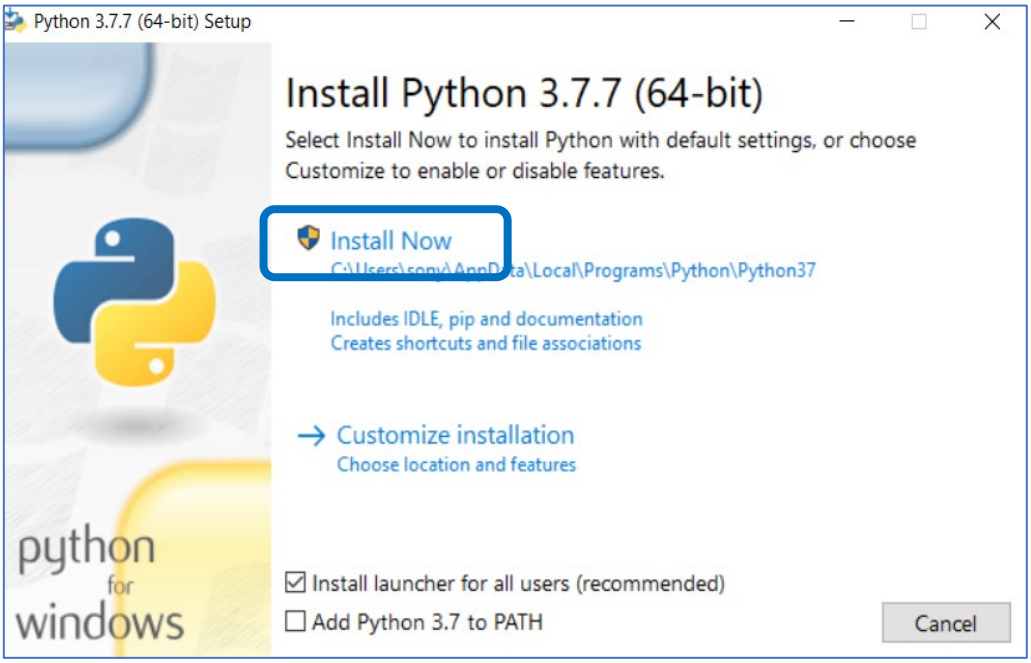

Figure 2.2 Installing Python

3. Confirm the installation by the following steps.
   - Open command prompt of Windows
   - Execute the command "py -3.7 -V"
   - When the result is "Python 3.7.7", the installation is normally finished.
     (No need to do the procedure 4.)

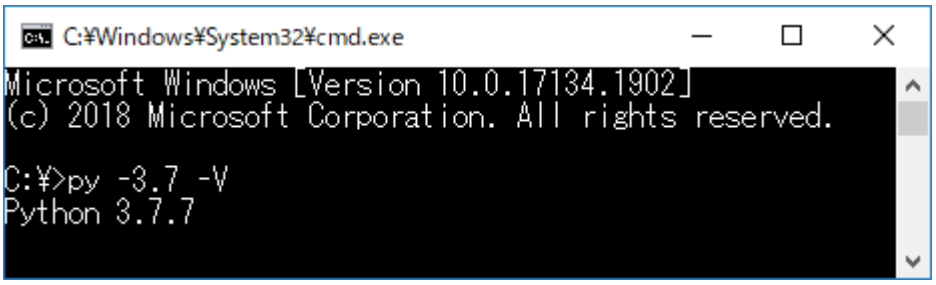

Figure 2.3 Confirmation of Installation Python

4. When the result is error in procedure 3, edit environment variables of Windows by the following steps. These window images are Windows 10 as an example.
   - Open "System Properties, then click "Environment Variables" button in the "Advanced" tab.

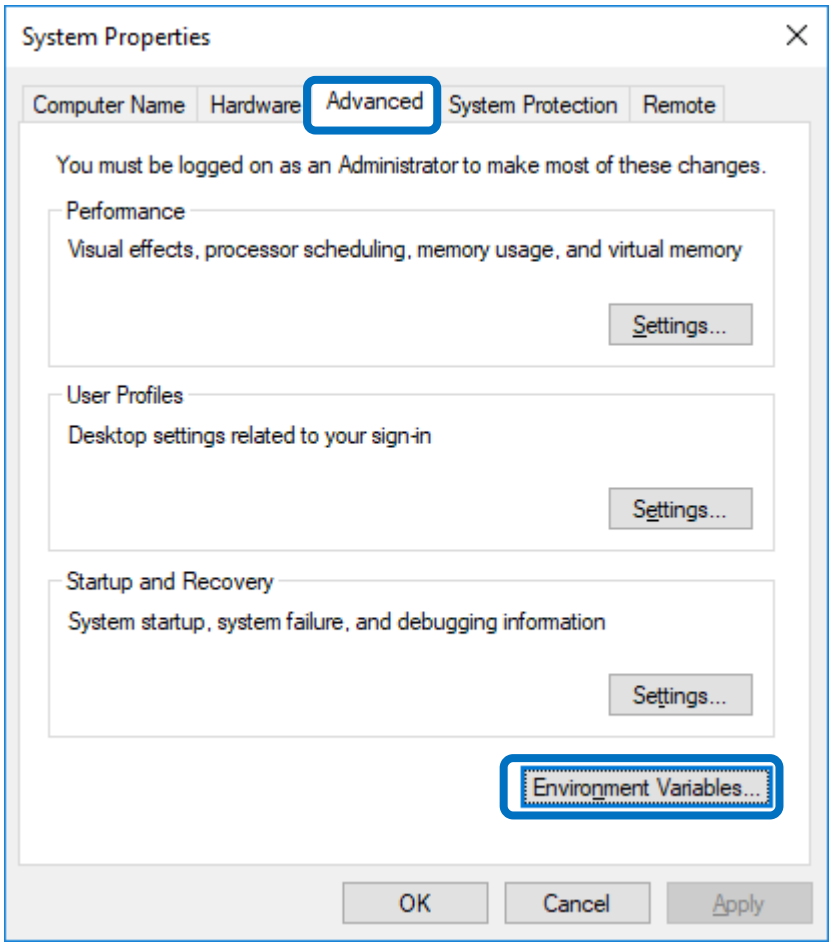

Figure 2.4 System Properties

- Select "Path" from Open "System variables", then click "Edit" button.

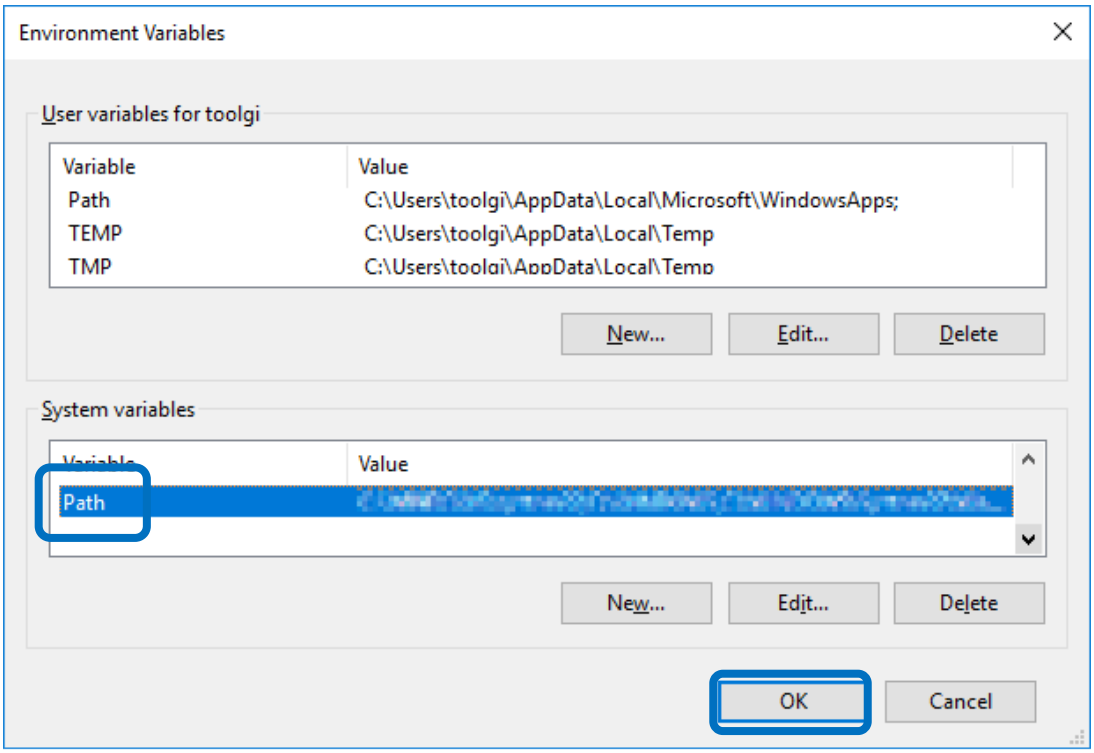

Figure 2.5 Setting of Environment Variables 1

- Click "New" button, then add "C:\Windows". Click "OK" button to close each dialog, then confirm procedure 3 again.

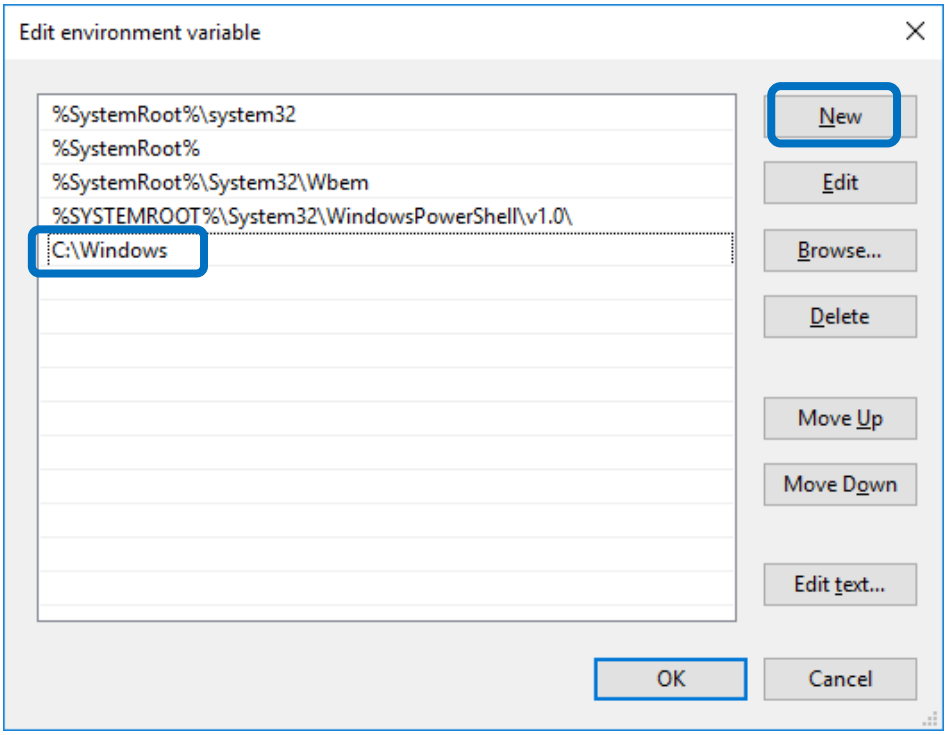

Figure 2.6 Setting of Environment Variables 2

2.4. Installing the Required Python Packages

- Installing TensorFlow
  1. Start the command prompt of Windows.
  2. Run the following command.
     cd C:\Users\<*windows-user-name*>\AppData\Local\Programs\Python\Python37\Scripts
     Note1: "<*windows-user-name*>" is the user name that you use for logging on Windows.
     Note2: Specify "Python" as the installation folder of Python3.7.7.
     Especially when multiple versions of Python are installed, be careful to specify the correct folder.
  3. Run the following command.
     pip3 install --upgrade tensorflow==2.4.0
     Note: Up to 10 minutes will be required until the command ends.

After you have finished installing above, run the following commands to install each package.

- Installing Torch, TorchVision
  pip3 install torch==1.5.1+cpu torchvision==0.6.1+cpu -f https://download.pytorch.org/whl/torch_stable.html

- Installing TorchSummary
  pip3 install torchsummary

- Installing Onnx
  pip3 install onnx==1.7.0

- Installing ProgressBar
  pip3 install progressbar33==2.4

- Installing Prettytable
  pip3 install prettytable==1.0.1

- Installing h5py
  pip3 install h5py==2.10.0

- Installing pycryptodome
  pip3 install pycryptodome==3.7.2

- Installing configparser
  pip3 install configparser==5.0.1

- Installing psutil
  pip3 install psutil==5.7.2

- Installing Keras (only when using standalone Keras)
  pip3 install keras==2.4.3

- Installing TensorFlow Lite
  pip3 install tflite==2.4.0

2.5. Installing Microsoft Visual C++ 2015 Redistributable

1. Download 64bit version installer of Microsoft Visual C++ 2015 Redistributable "vc_redist.x64.exe" from the following Web site.
   https://www.microsoft.com/en-US/download/details.aspx?id=52685

2. Double-click on the installer to start installation.
   Proceed with installation according to the instructions of the installer.

3. Using the e-AI Translator

This chapter explains how to use the e-AI Translator.

3.1. Basic Procedure for Using the e-AI Translator

1. Start the e² studio.
   Create a new project as the project or open an existing project.
2. Start the e-AI Translator.
   Click on the [TR] button or select [Configure and Translate] from the [Translator] menu.

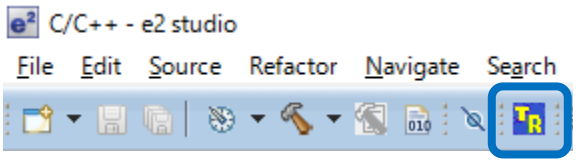

Figure 3.1 Start Button for the e-AI Translator

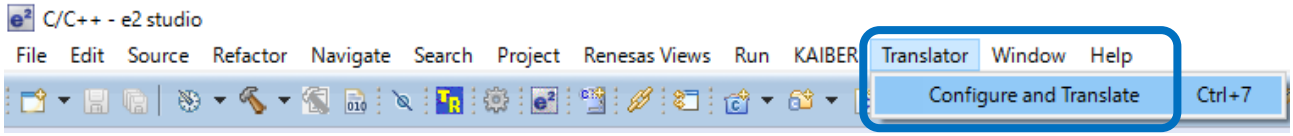

Figure 3.2 Start Menu of the e-AI Translator

3. The e-AI Translator is activated as shown below.

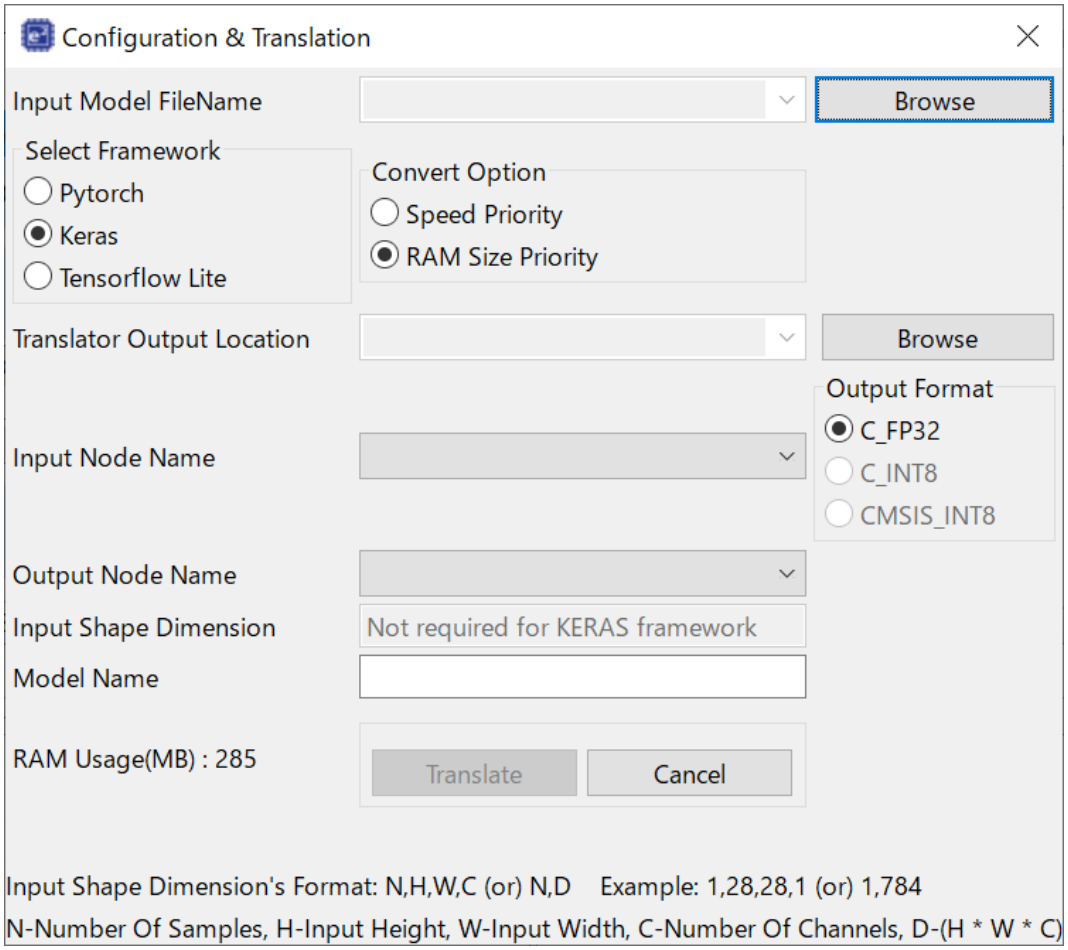

Figure 3.3 e-AI Translator

Table 3-1 Descriptions of Each Area of the e-AI Translator

Name	Description
Input Model File Name	Specifies the trained model file which is created by PyTorch or Keras or TensorFlow(tf.keras) or TensorFlow Lite.
Select Framework	Specifies the type of trained model. Select "Keras" when using TensorFlow(tf.keras).
Convert option	Specifies where to allocate weight data, ROM or RAM. When RAM size is prioritized, weight data is allocated to ROM. When speed is prioritized, weight data is allocated to RAM.
Translator Output Location	Specifies the output destination folder for the source files and header files that are output as the results of conversion. Specify the source folder of e^2 studio project.
Input Node Name Output Node Name	The input / output node name is specified automatically when using the trained model of Keras or TensorFlow(tf.keras).
Output Format	Specifies output source file format from the following three choices. - C_FP32: When translating PyTorch or TensorFlow(tf.keras) models to source files, "C_FP32" is selected automatically. - C_INT8: When translating TensorFlow Lite models to source files, prioritize ROM/RAM usage. - CMSIS_INT8: When translating TensorFlow Lite models to source files, prioritize inference speed by utilizing CMSIS library.
Input Shape Dimensions	Specifies the size of the input layer of the neural network. This is only required when the framework is PyTorch.
Model Name	This area is used only when using multiple neural networks. Refer to "3.2. Translation of multiple neural networks" for the detail.
RAM Usage (MB)	Displays PC RAM usage as a reference information when translating.
Translate button	Converts the trained model of the AI to source files and header files.

4. Set [Select Framework].
   Specify the type of trained model.
   - Pytorch: Trained model by using PyTorch.
   - Keras: Keras or TensorFlow(tf.keras) was used for training by the AI.
   - Tensorflow Lite: 8bit quantized model by using TensorFlowLite,

   Note: Specify [Select Framework] setting before specifying [Input Model File Name].

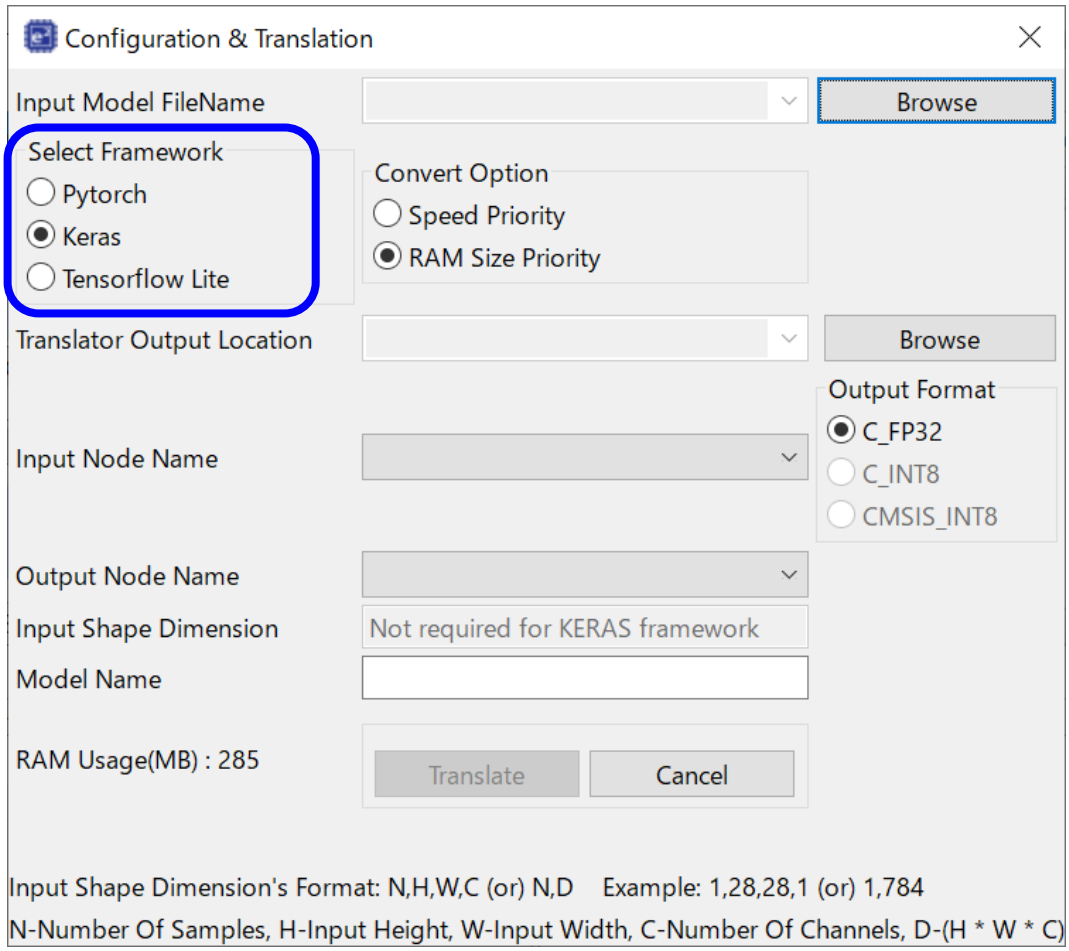

Figure 3.4 Setting [Select Framework]

5. Click on the [Browse] button to set [Input Model File Name].
   Specify the folder containing the trained model created by PyTorch or Keras or TensorFlow(tf.keras) or TensorFlow Lite.
   One-byte characters can be used for the folder name and path name. Do not use two-byte characters.
   The files listed following the screenshot must have been input in the cases of PyTorch, Keras, TensorFlow(tf.keras).

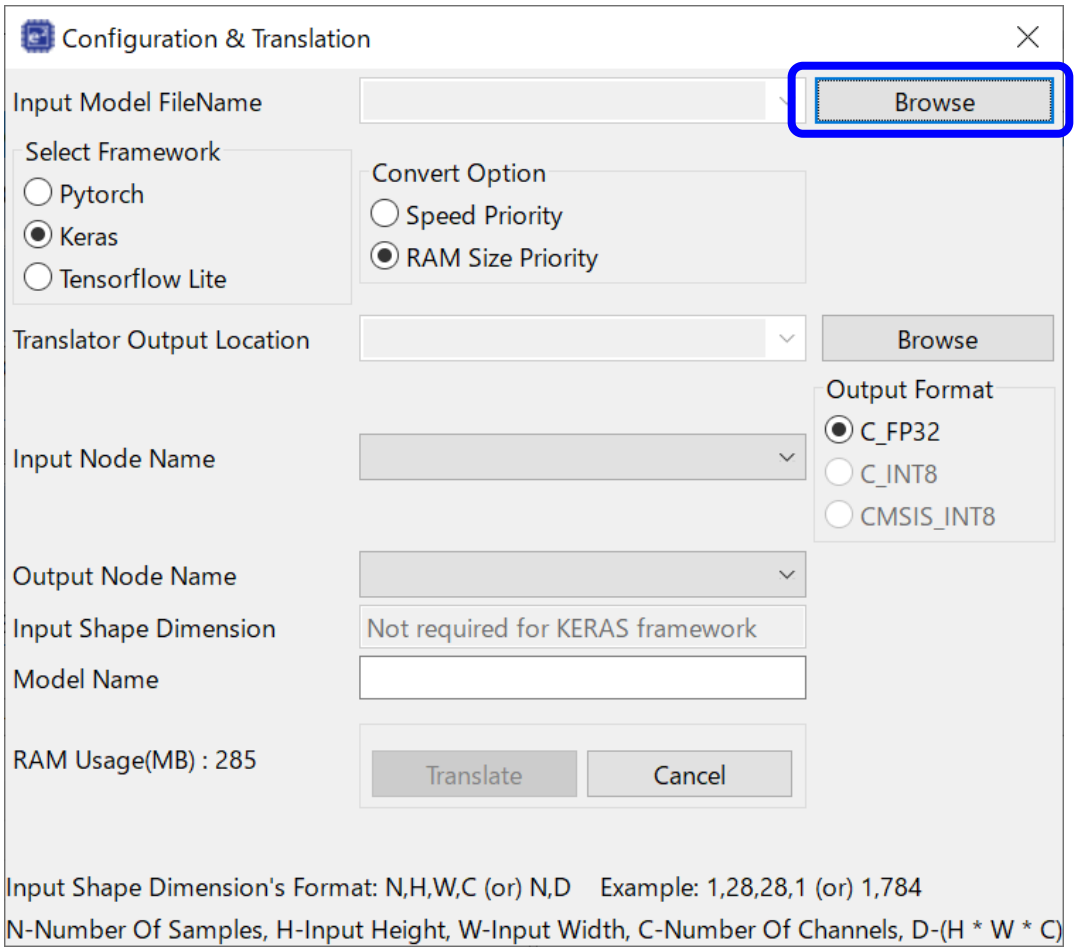

Figure 3.5 Setting [Input Model File Name]

In the case of PyTorch

- Put following 2 files into the same folder, then specify the file which has the extension "*.pth" as the input model file. Refer to "3.5. How to create/save PyTorch model file" for the detail to prepare these 2 files.
- - The trained model file of PyTorch which has the extension "*.pth".
- - The neural network structure definition file which has the extension "*.py".

In the case of Keras, TensorFlow(tf.keras):

- Specify the trained model file saved as HDF5 format (file extension: *.h5) as the input model file.

In the case of TensorFlow Lite:

- Specify the trained model file (file extension: *.tflite) as the input model file.

6. Set [Convert option].
   Specify the quality to which priority is to be given when the trained model of the AI is converted.
   - Speed priority: Prioritize execution speed by allocating weight data to RAM.
   - RAM size priority (Recommended Note): Prioritize RAM usage by allocating weight data to ROM.
   Note: "RAM size priority" is recommended because the size of weight data is huge, generally.

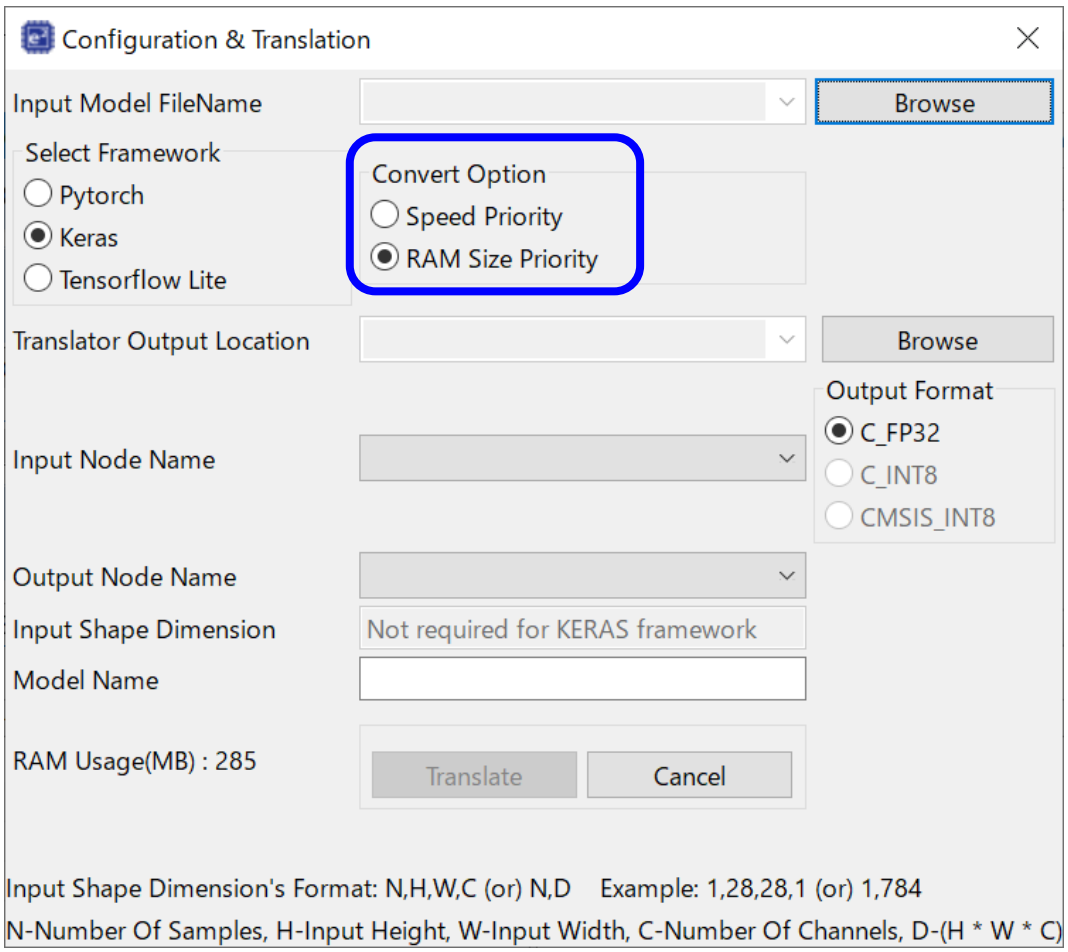

Figure 3.6 Specifying [Convert option]

7. Click on the [Browse] button to set [Translator Output Location].
   Specify the destination folder for output of the source and header files from conversion.
   One-byte characters can be used for the folder name and path name. Do not use two-byte characters.
   Select the folder where the source files of the e^2 studio have been registered (e.g. the "src" folder in the project folder).

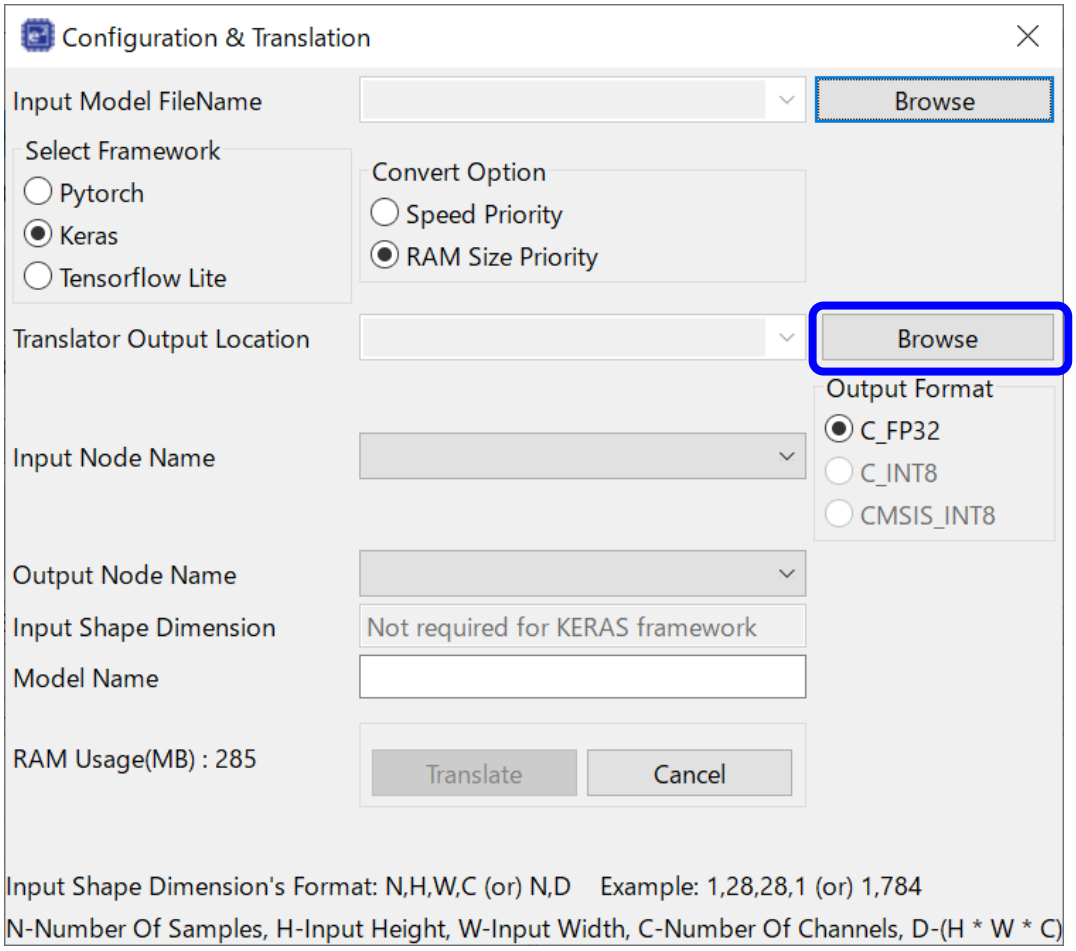

Figure 3.7 Specifying [Translator Output Location]

8. Confirm [Input Node Name], [Output Node Name], then specify [Output Format].

- [Input Node Name] and [Output Node Name]:
  The input / output node name is specified automatically when using the trained model of Keras or TensorFlow(tf.keras). Basically, the change of this setting is not required. But if the setting value is not correct, specify the setting value manually.

- [Output Format]:
  Specify the output source file format by the trained model format as "Input Model File Name". When using "CMSIS_INT8", refer to "3.7 How to use CMSIS_INT8" also.
  - When specifying the model of PyTorch, Keras, TensorFlow(tf.keras):
    "C_FP32" is specified automatically.
  - When specifying the 8bit quantized model of TensorFlow Lite:
    Specify "C_INT8" or "CMSIS_INT8".
    - C_INT8: Translate source files which ROM/RAM usage is prioritized.
    - CMSIS_INT8: Translate source files which inference speed is prioritized.
      (only for RA family and RX family)

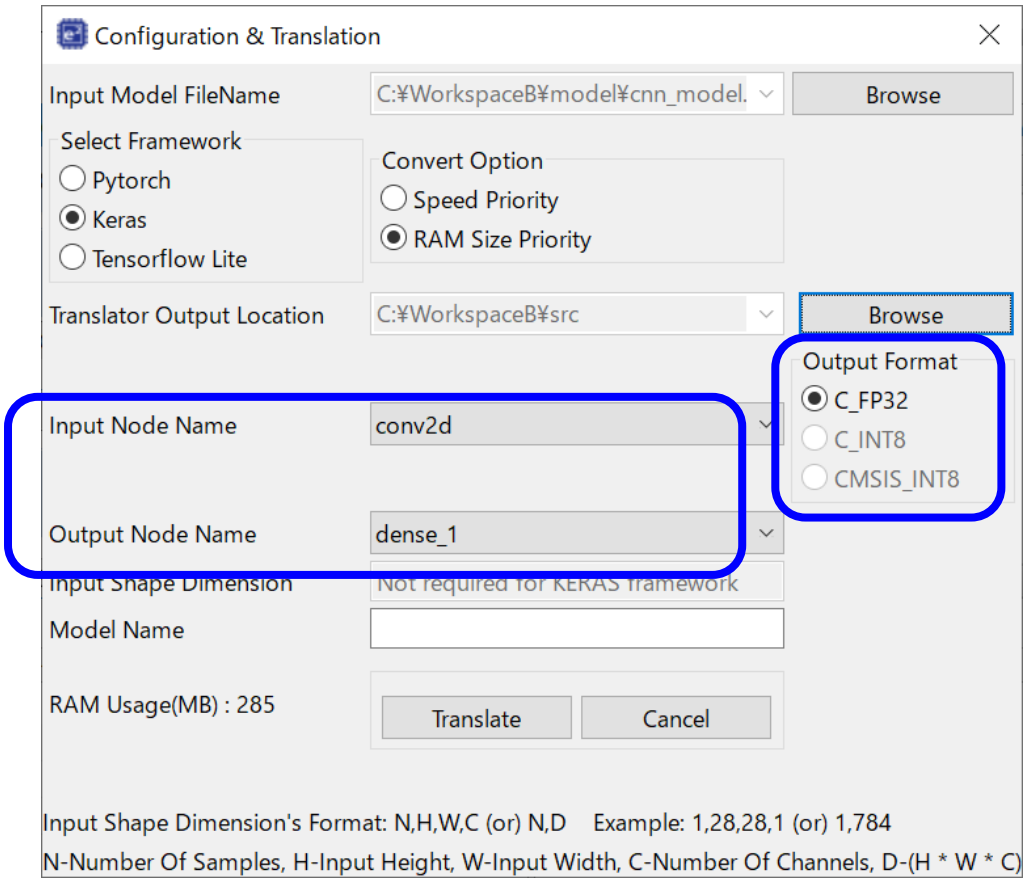

Figure 3.8 Confirming [Input Node Name], [Output Node Name], [Output Format]

9. Set [Input Shape Dimensions] (only when using PyTorch).

Specify the size of the input layer of the neural network in the order of "C,H,W" or "D". This is only required when PyTorch is used as the framework.

Only numerical characters and commas (,) can be used for this input area, and input them as one-byte characters.

C: Number of Channels; H: Input Height; W: Input Width; D: C*H*W

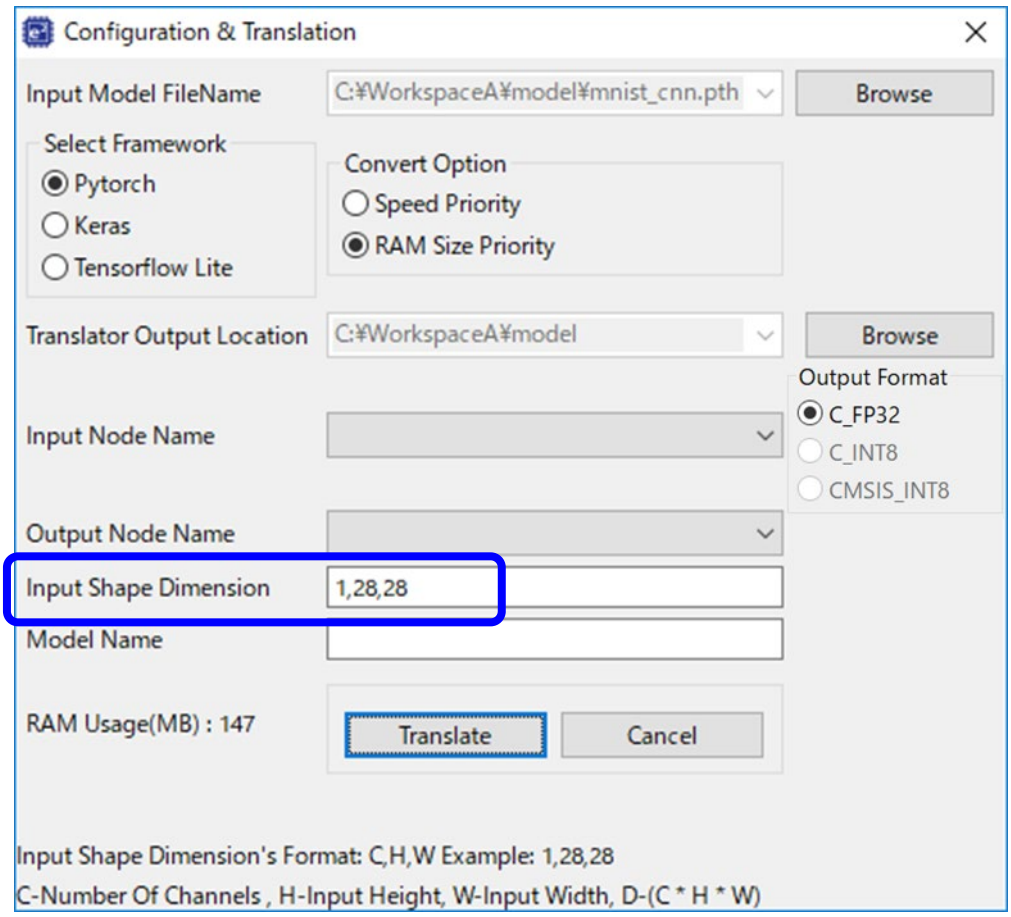

Figure 3.9 Specifying [Input Shape Dimensions]

10. Click on the [Translate] button to convert the trained model into C-source and header files.

Note: Need to set [Model Name] area only when using multiple neural networks.

Be blank when using single neural network. Refer to "3.2 Translation of multiple neural networks" for the detail.

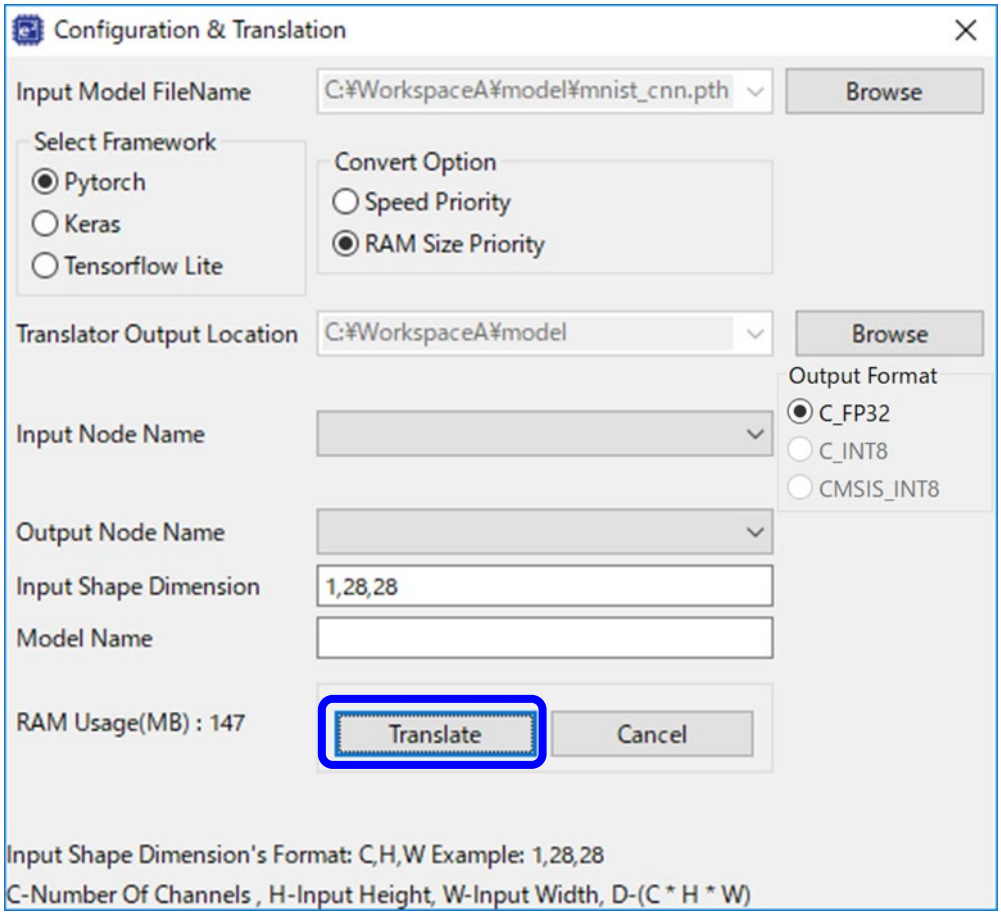

Figure 3.10 [Translate] Button

11. The converted C-source and header files are output to the "Translator" folder under the "src" folder.
    To build from the header files, add this folder to the include file path in the settings of the build tool.

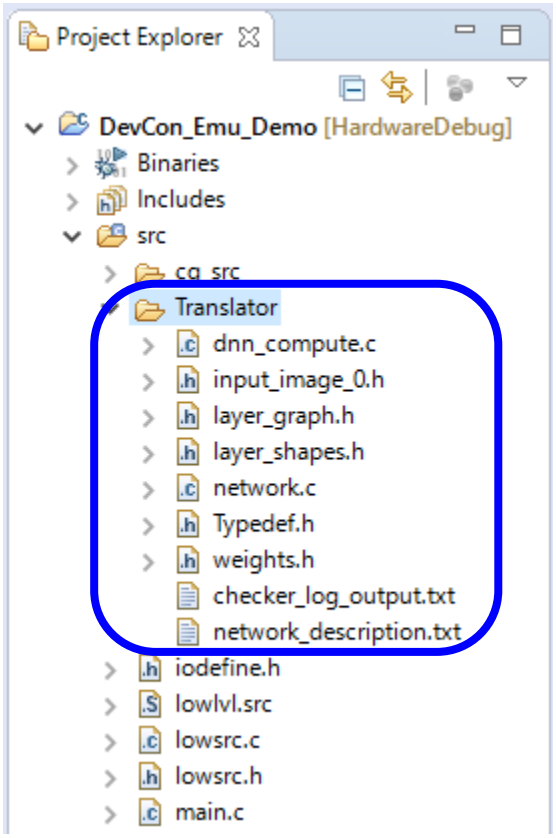

Figure 3.11 Output Folder and Files from Conversion

12. By calling "dnn_compute" function in "dnn_compute.c" file, AI inference operation can be used. The specification of "dnn_compute" function is as follows.

Table 3-2 Specification of "dnn_compute" funcction

Function name	dnn_compute	
Outline	Execution of inference operation.	
Function interface Note	TsOUT* dnn_compute(TsIN* input, TsInt *errorcode) Note: Refer to "Typedef.h" about definitions of TsOUT, TsIN, TsInt	
Return Value	Result of inference	
Arguments	input	Input data of inference operation.
	errorcode	Variable to detect memory allocation error caused by malloc instruction. - 0: Memory allocation error did not occur. - 1: Memory allocation error occurred. Initialization of this variable is not required before calling "dnn_compute" function because it is first initialized to zero in "dnn_compute". Therefore, initialization is not required before calling this function. Do not use inference result when the value of "errorcode" is 1 after "dnn_compute" function, because memory allocation error occurred. Build the program again after increasing heap memory for malloc instruction. The memory allocation by malloc instruction is executed only when CC-RX compiler and CC-RL compiler is used. If other compilers are used, the memory allocation error does not occur.

Note: When using this function for RL78 family, Warning occurs. Because the address of pointer is 16-bit in the case of RL78 family. The warning can be resolved by utilizing __far qualifier as follows.

[Before handling: CC-RL compiler code example]

```
TsOUT *prediction;
TsInt errorcode;
prediction = (TsOUT*) (intptr_t) dnn_compute (predict_data_preprocessed, &errorcode);
```

[After handling: CC-RL compiler code example]

```
__far TsOUT *prediction;
TsInt errorcode;
prediction = (__far TsOUT*) (intptr_t) dnn_compute (predict_data_preprocessed, &errorcode);
```

13. An outline of the results of conversion can be checked by reading the following two files in the "Translator" folder.
    - checker_log_output.txt: The ROM and RAM sizes to be used can be checked. note
      Also, the number of multiply-and-accumulate operations can be checked as information related to the speed of execution.
      Refer to "3.4. Confirmation of ROM/RAM usage" for the detail.
    - network_description.txt: Information on the converted neural network structure can be checked.

    Note: Estimation function of ROM size and RAM size are used by neural network.
    This file shows ROM size and RAM size when specifying "RAM size priority" as "convert option" in its current version. Even if "convert option" is changed, the estimation result is not changed.

14. Build the program which includes AI processing.
    Subsequently, the procedure for debugging the program is the same as that for programs in general.
    For the procedure for debugging the program through the e^2 studio, refer to the documents as follows.

Target device	Document title	Document number
RE Family	e^2 studio Quick Start Guide	R20UT5034EJ
RA Family	e^2 studio Quick Start Guide	R20UT4989EJ
RZ Family	e^2 studio Getting Started Guide	R20UT4535EJ
Other MCU Family	e^2 studio Getting Started Guide	R20UT4819EJ

3.2. Translation of multiple neural networks

When using the multiple neural networks in single user program, the translation procedure is as follows.

In this chapter, these 2 neural networks are used as an example.

- Neural network for abnormal signal detection.
- Neural network for abnormal sound detection.

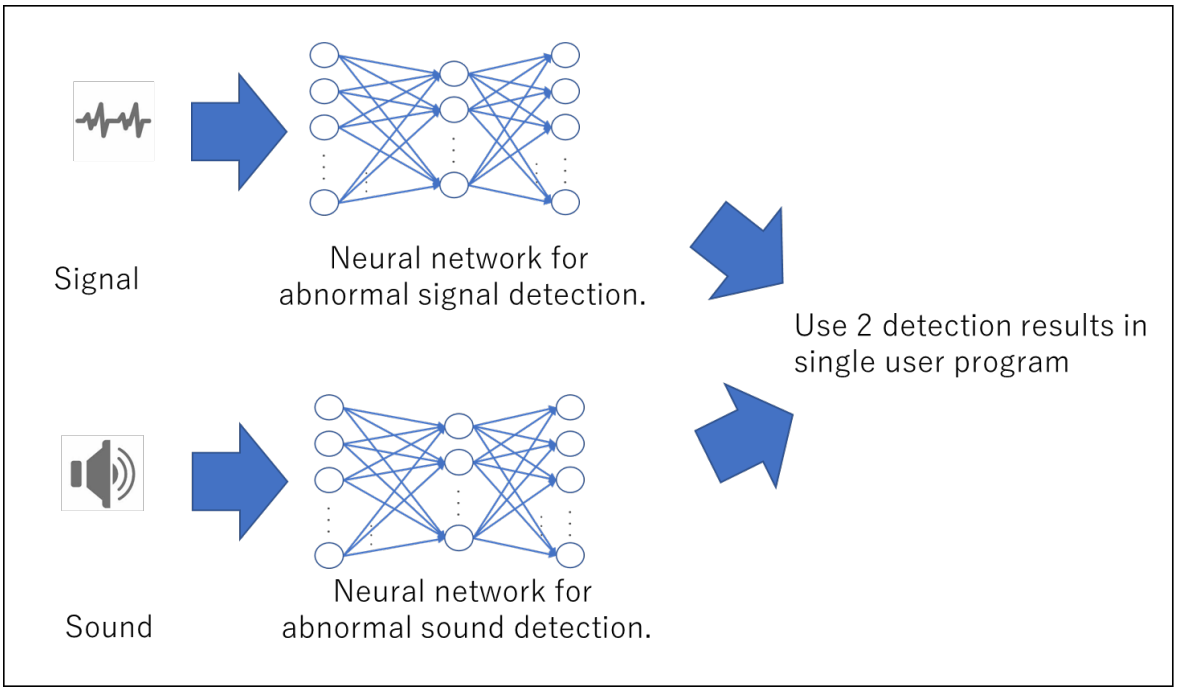

Figure 3.12 Translation example of 2 neural networks

1. Translate the 1st neural network for abnormal signal detection.
   - Launch e-AI Translator, then specify the trained model of 1st neural network to [Input Model File Name].
   - Specify different folder to [Translator Output Location] between 1st neural network and 2nd neural network.
   - Specify the 1st neural network name to [Model Name] for distinction of each neural network.
     In this case, "Signal" is used as the model name.
     Note: Only one-byte alphanumeric characters can be used, the maximum length is 8 characters.
   - After above settings, click [Translate] button for the translation.

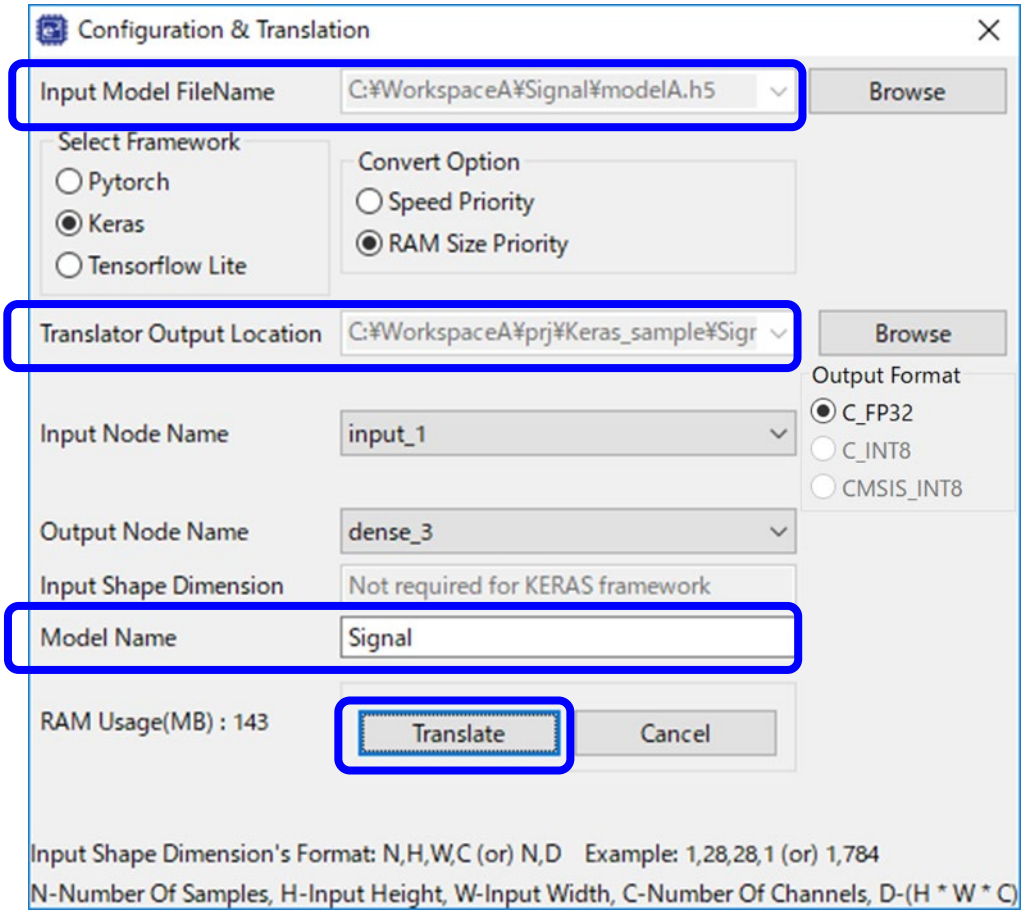

Figure 3.13 Translation of 1st neural network for abnormal signal detection

2. Translate the 2nd neural network for abnormal sound detection.
   - Launch e-AI Translator, then specify the trained model of 2nd neural network to [Input Model File Name].
   - Specify different folder to [Translator Output Location] between 1st neural network and 2nd neural network.
   - Specify the 2nd neural network name to [Model Name] for distinction of each neural network.
     In this case, "Sound" is used as the model name.
     Note: Only one-byte alphanumeric characters can be used, the maximum length is 8 characters.
   - After above settings, click [Translate] button for the translation.

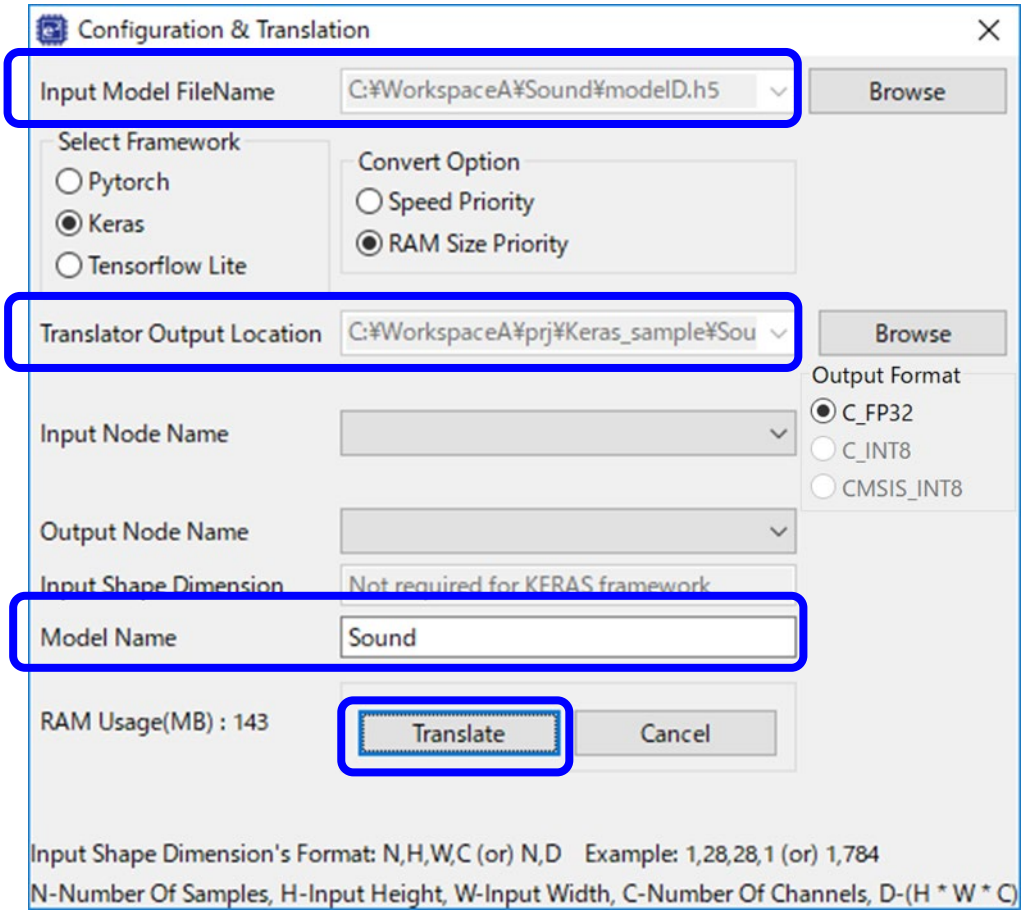

Figure 3.14 Translation of 2nd neural network for abnormal sound detection

3. Confirm the translated source files.
   Source files and header files are translated with the Model Name. However, "typedef.h" and "imput_image_0.h" don't have the Model Name, because they are same files.
   For using the inference function, call the following functions/
   - Inference function for signal: dnn_compute_Signal() function in "dnn_compute_Signal.c"
   - Inference function for sound: dnn_compute_Sound() function in "dnn_compute_Sound.c"

Table 3-3 Translated source files and header files

	1st neural network for abnormal signal detection	2nd neural network for abnormal sound detection
Translation Result	dnn_compute_Signal.c	dnn_compute_Sound.c
	layer_graph_Signal.h	layer_graph_Sound.h
	layer_shapes_Signal.h	layer_shapes_Sound.h
	network_Signal.c	network_Sound.c
	weights_Signal.h	weights_Sound.h
	Typedef.h	Typedef.h
	input_image_0.h	input_image_0.h

4. Marge "network_Signal.c" and "network_Sound.c".
   The functions and variables for each neural network layers are defined in these 2 files.
   When using the same layer in these 2 neural networks, build error occurs because of multiple definition of functions and variables.
   Therefore, these multiple definitions need to be commented out.

5. Build user program.
   Multiple neural networks can be used in single user program by this procedure.

3.3. Sample of a Trained Model and Sample Code of the Main Function

The installation folder of the e-AI Translator contains samples of a trained model and the main function. Use these samples as required.

[Stored directory]

C:\Renesas\e2_studio\eclipse\plugins\com.renasas.eaitranslator_2.2.0\Translator

|

|--- main.c: Sample code of a main function (calling of the functions for inference processing)

|--- input: Samples of input characters 0 to 9; header file format

|

--- model

|

|--- Keras: Trained model of the AI using Keras

|--- tflite: Trained model of TensorFlow Lite

---- PyTorch: Trained model of the AI using PyTorch

[Trained model of Keras]

Handwriting recognition processing for Modified National Institute of Standards and Technology (MNIST) is provided as the sample program.

- A handwritten character from 0 to 9 is entered to the neural network and infer which number has been entered.
- The size of the input is 28 × 28 pixels, and they are grayscale.
- The e-AI Translator outputs the probabilities of which number has been entered.

The inference operation on MCU or MPU can be checked by translating the trained model and building with "main.c" and header files mentioned above.

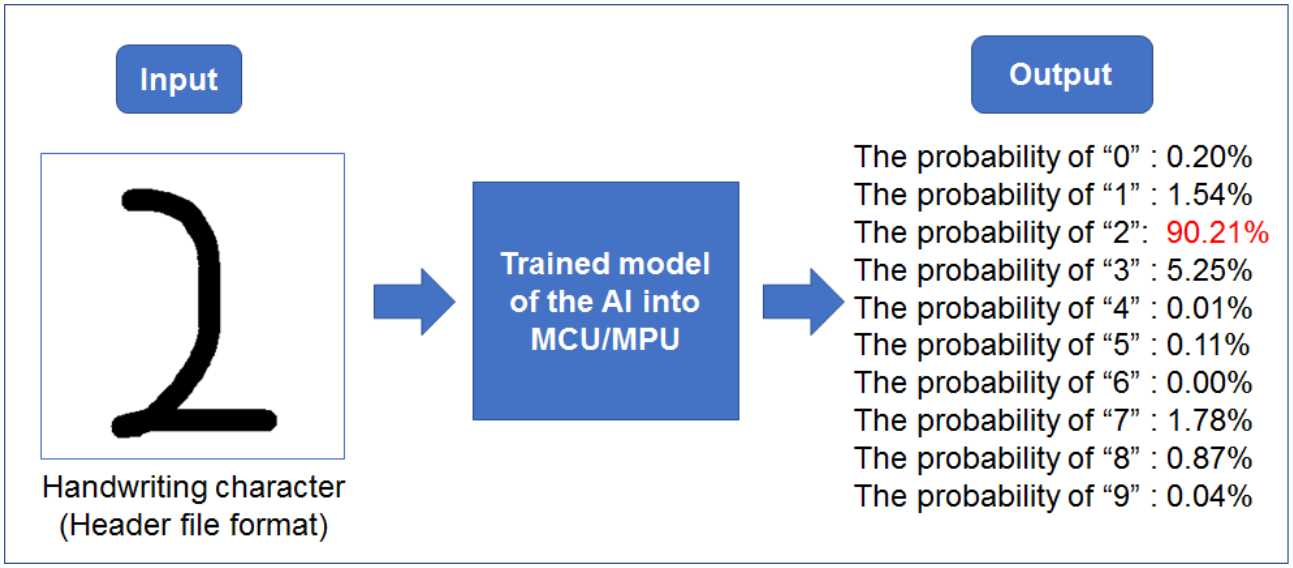

Figure 3.15 Image of Behavior of the Keras Sample

[Trained model of TensorFlow Lite]

There are some models as sample model files. These cannot be used with "main.c" and header files mentioned above, but the translation operation can be checked by using these model files.

[Trained model of PyTorch]

The trained model as a sample is for classification of flowers.

- Flower images like Tulip, Sunflower are entered to the neural network and infer which flower has been entered.
- The input data is channel number:3(RGB), size:100 × 100 pixels.
  (The "Input Shape Dimension" setting of e-AI Translator should be "3,100,100".)

This cannot be used with "main.c" and header files mentioned above, but the translation operation can be checked by using this model file.

3.4. Confirmation of ROM/RAM usage

ROM/RAM usage for inference operation can be checked by using "checker_log_output.txt".

The confirmation method is as follows.

Layer Information		Size Information		Speed Information
Layer Output No.	Layer Name	ROM(Byte)	RAM(Byte)	MAC Operations(times)
1	Input	-	3,136	-
2	Convolution	160	16,144	28,224
3	ReLU	-	12,544	-
4	Max Pooling	-	3,136	-
5	Convolution	592	7,232	28,224
6	ReLU	-	3,136	-
7	Max Pooling	-	784	-
8	Full Connect	19,700	100	4,900
9	ReLU	-	100	-
10	Full Connect	1,040	40	250
11	Softmax	-	40	-
12	Output	-	40	-
		①	②	③
	TOTAL	21,492	46,432 19,816	61,598

1 : ROM usage of inference operation.

2(upper) : Total RAM size of each layer.
(RAM usage of inference operation when using e-AI Translator V1.4.0 or lower version.)

2(lower) : Actual RAM usage of inference operation.

3 : The number of multiply-and-accumulate operations.
If the number is bigger, the inference speed is slower.

3.5. How to create/save PyTorch model file

Be careful about 3 points as follows when creating the trained model of PyTorch.
The error occurs if these points are not satisfied.

1. Configuration of Python files
   Create Python file for model definition and for training as the separated files.

2. Save method of the model file
   Do not specify "state_dict" when saving the model file.

3. Translation by e-AI Translator
   Put Python file for model definition(*.py) and trained model file (*.pth) into the same folder.
   Specify trained model file (*.pth) as "Input Model File Name" of e-AI Translator.

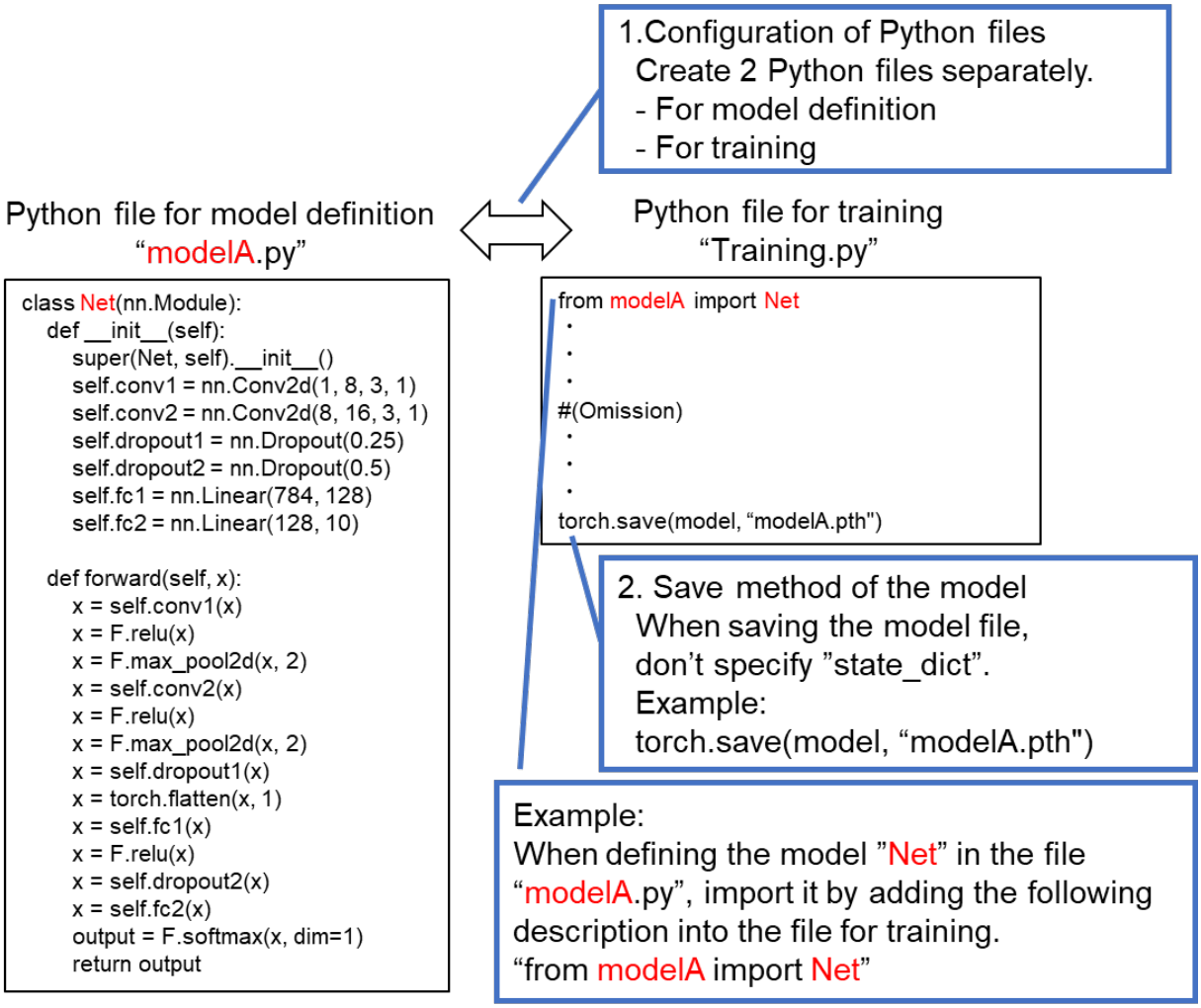

Figure 3.16 How to create/save PyTorch model file

3.6. How to create/save TensorFlow Lite model file

The trained model file Keras or TensorFlow(tf.keras) can be 8bit quantized by using TensorFlow Lite. As following figure, the weight data and activation data can be quantized from 32bit float format to 8bit signed int format. 8bit quantization is very useful technique because ROM usage and RAM usage can be reduced.

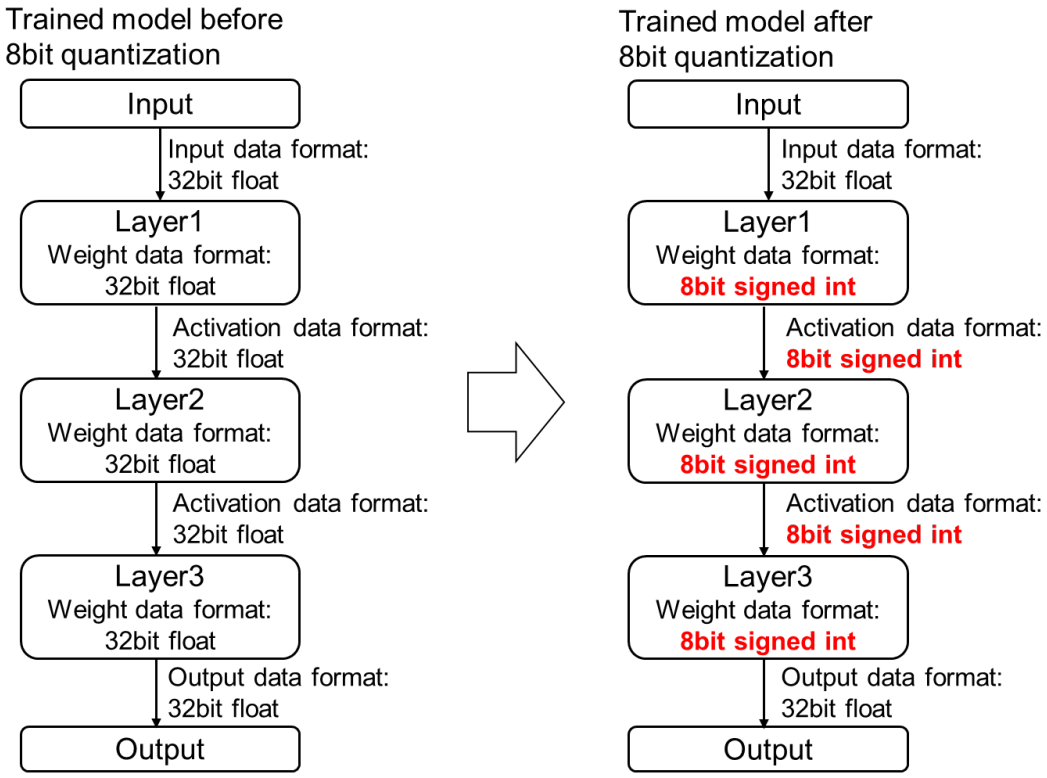

Figure 3.17 Compaction example by 8bit quantization of TensorFlow Lite

・Support model of TensorFlow Lite by e-AI Translator

e-AI Translator supports 8bit quantized model of TensorFlow Lite in the following table.
When using quantization, match the input/output data format between training and quantization.

Table 3-4 Support model of TensorFlow Lite by e-AI Translator

Quantization method	Input data format	Output data format	Weight data format	Activation data format
Quantization both weight and activation data Note	32bit float	32bit float	8bit signed int	8bit signed int
	32bit float	8bit signed int	8bit signed int	8bit signed int
	32bit float	8bit unsigned int	8bit signed int	8bit signed int
	8bit signed int	32bit float	8bit signed int	8bit signed int
	8bit signed int	8bit signed int	8bit signed int	8bit signed int
	8bit signed int	8bit unsigned int	8bit signed int	8bit signed int
	8bit unsigned int	32bit float	8bit signed int	8bit signed int
	8bit unsigned int	8bit signed int	8bit signed int	8bit signed int
	8bit unsigned int	8bit unsigned int	8bit signed int	8bit signed int

Note: The TensorFlow Lite model which is quantized only weight data is not supported.

・How to create TensorFlow Lite model

TensorFlow Lite model can be created by the following steps.
1.Create the trained model (file extension: *.h5) by Keras or TensorFlow(tf.keras)
2.Quantize the trained model to 8bit quantized model by TFLiteConverter
3.Save the 8bit quantized model. (file extension: *.tflite)

Refer to the example script on the next page for 8bit quantization.
[Explanation of example script]
- Dataset for training: MNIST
- Trained model file name: cnn_model.h5
- Output file1: 8bit_cnn_model_fp32io.tflite
  (Quantization weight and activation data, Input/Output format: 32bit float)
- Output file2: 8bit_cnn_model_uint8io.tflite
  (Quantization both weight and activation data, Input/Output format: 8bit unsigned int)

Note: By changing from "tf.uint8" to "tf.int8" in the script, 8bit unsigned int input/output model also can be created.

In addition, refer to following TensorFlow Lite Web page for more detail of 8bit quantization.
https://www.tensorflow.org/lite/performance/post_training_integer_quant

```
import tensorflow as tf
import numpy as np
import os

# Load MNIST dataset
mnist = tf.keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Normalize the input image so that each pixel value is between 0 to 1.
train_images = train_images.astype(np.float32) / 255.0
test_images = test_images.astype(np.float32) / 255.0

# Representative data generation for calibration
def representative_data_gen():
  for input_value in tf.data.Dataset.from_tensor_slices(train_images).batch(1).take(100):
    # Model has only one input so each data point has one element.
    yield [input_value]

#Convert to 8bit quantized model with FP32 input/output
load_model = tf.keras.models.load_model("cnn_model.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(load_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_data_gen
tflite_model = converter.convert()
open("8bit_cnn_model_fp32io.tflite", "wb").write(tflite_model)

#Convert to 8bit quantized model with 8bit input/output
load_model = tf.keras.models.load_model("cnn_model.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(load_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_data_gen
# Ensure that if any ops can't be quantized, the converter throws an error
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
# Set the input and output tensors to uint8 or int8
converter.inference_input_type = tf.uint8  # can be changed to tf.int8
converter.inference_output_type = tf.uint8 # can be changed to tf.int8
tflite_model = converter.convert()
open("8bit_cnn_model_uint8io.tflite", "wb").write(tflite_model)
```

Figure 3.18 Example script for 8bit quantization of TensorFlow Lite

3.7. How to use CMSIS_INT8

The inference speed can be faster by combining translated source code and CMSIS NN, CMSIS DSP libraries when selecting "CMSIS_INT8" from "Output format".
After translating source files by e-AI Translator, add CMSIS NN, CMSIS DSP libraries by following steps.

- Support devices of CMSIS_INT8 library
  RA family, RX family

- Usage for RA family
  1.Double-click "configuration.xml" on project explorer after creating project on integrated development environment e² studio. Then open "Stacks" tab of "FSP Configuration" view.

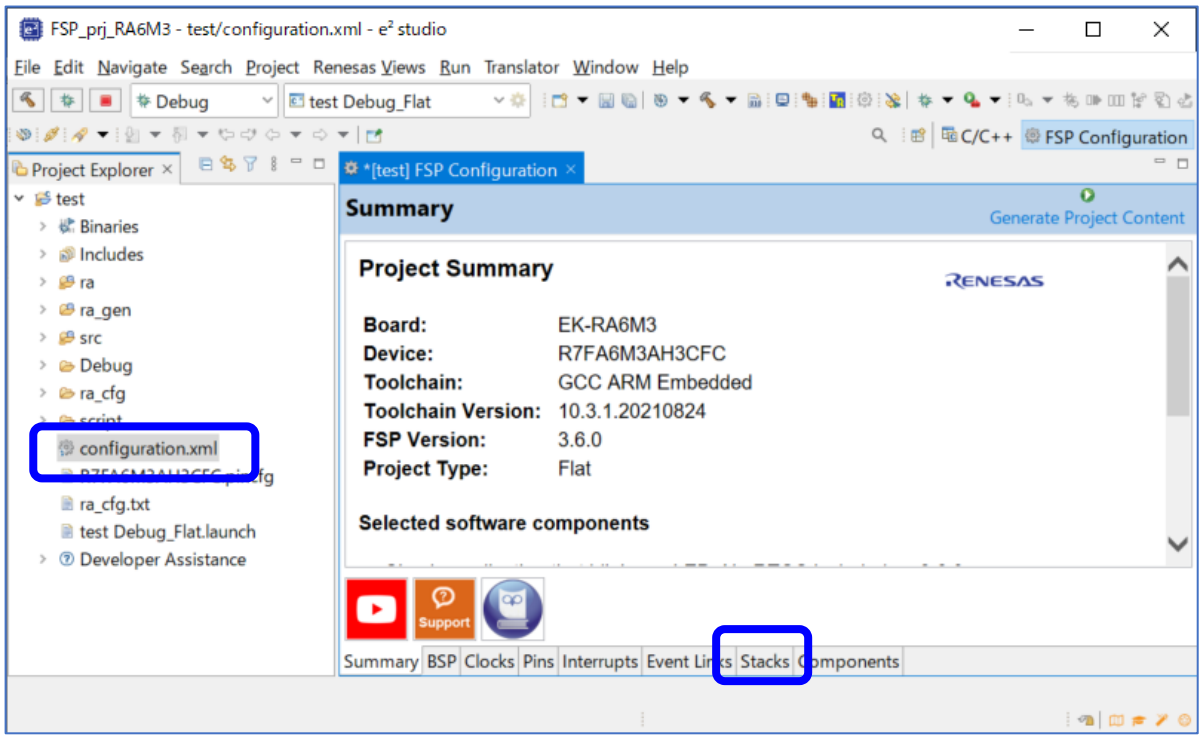

Figure 3.19 "Stacks" tab of "FSP Configuration" view

2.Click "New Stack" tab on the upper side of "Stacks" tab, then specify "Arm CMSIS5 NN Library Source" in "Artificial Intelligence".

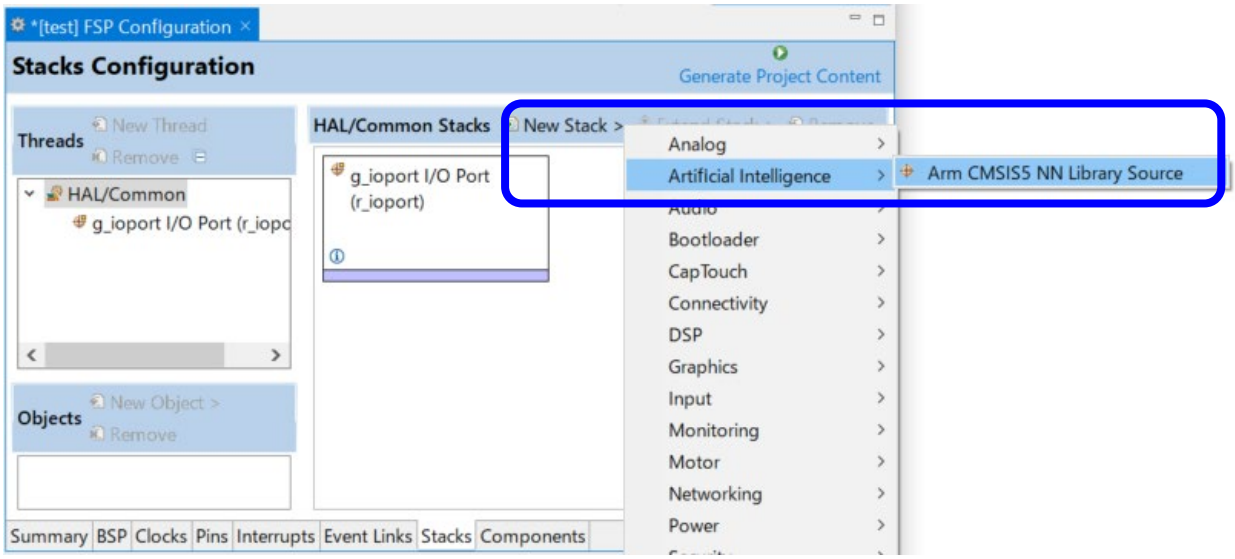

Figure 3.20 Addition of CMSIS libraries

3.Confirm the addition of "Arm CMSIS NN Library Source" and "Arm CMSIS DSP Library Source", then push "Generate Project Content" button.

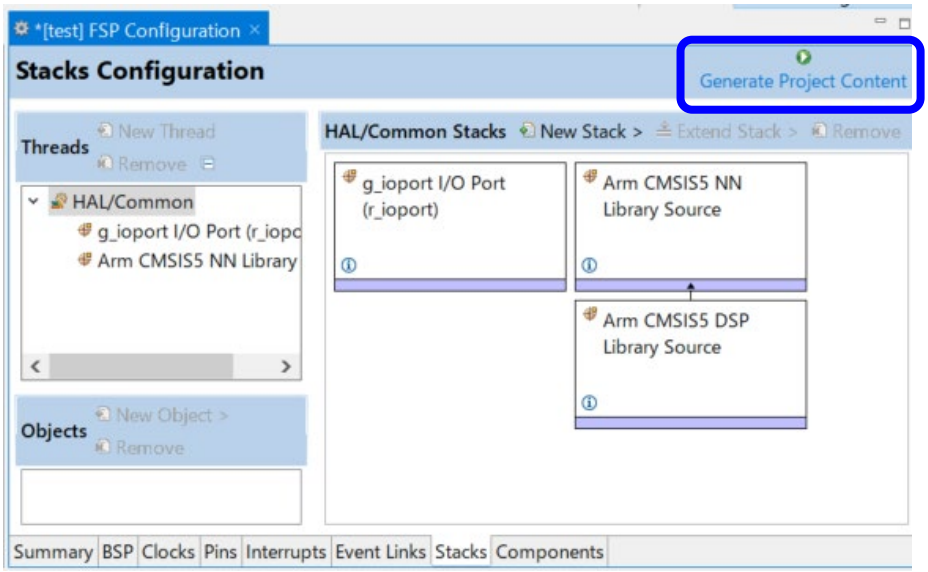

Figure 3.21 Generation of CMSIS libraries

- Usage for RA family
  Getting CMSIS for RX library is required.
  Refer to the attached document of CMSIS for RX library for the detail.

- Supplement about "Convert option" and "Output format" options
  Both "Convert option" and "Output format" are options regarding inference speed and ROM/RAM usage. Refer to following table about the operation of each option.

Table 3-4 Descriptions of Each Area of the e-AI Translator

Option name		Content and operation of setting
Convert option		Option for allocating weight data
	RAM Size Priority (Recommended Note)	Prioritize RAM usage of translated program by allocating weight data to ROM.
	Speed Priority	Prioritize execution speed of translated program by allocating weight data to RAM.
Output Format		Option for specifying inference program
	C_INT8	Translate ROM/RAM prioritized program.
	CMSIS_INT8	Translate inference speed prioritized program by utilizing CMSIS libraries.

Note: "RAM size priority" is recommended because the size of weight data is huge, generally.

4. Points for Caution

- The non-supported types of neural network cannot be correctly converted.
  For the supported types of neural network, refer to section 1.2, Types of Convertible Neural Networks, and section 1.3, Supported API list.

- When using preprocessing such as Reshape and so on for the input layer of neural network, this preprocessing is removed in the inference function of e-AI Translator. Therefore, use the data shape after preprocessing for the input of the inference function.
  Additionally, when using the 3D shape input data for the inference function and using "CMSIS_INT8" as "Output format", translation error may occur without removing preprocessing. Use trained model which is saved without preprocessing in this case.

- When using the 4D shape input data for the inference function, the input data order of the inference function is different by the setting of "Output format".
  - "C_FP32" or "C_INT8": input data [N, C, H, W] order
  - "CMSIS_INT8": input data [N, H, W, C] order
  Note that the standard specification of input order when training the model is also different between frameworks. The input order of PyTorch is [N, C, H, W], the input order of Keras and TensorFlow(tf.keras) is [N, H, W, C].
  N: Number of Samples, H: Input Height, W: Input Width, C: Number of Channels

- After pre-processing the input data, convolutional layer or fully connected layer can be used as the first data processing layer in neural network.
  Activation function, clip function, normalization layer and pooling layer cannot be used.
  When using them, eAI-311 error occurs.

- If the PC used for training the neural network and the PC using the e-AI translator are different, use same version of deep learning framework on both PC.
  And when using Lambda layer of Keras or TensorFlow (keras.layers.Lambda or tf.keras.layers.Lambda), use same version of Python on both PC. When detecting the version mismatch, eAI-310 error occurs.

- When using data shape conversion with Reshape function, e-AI Translator supports the conversion only from 4D shape to 2D shape or from 2D shape to 4D shape. The Reshape from 4D shape to 4D shape and 2D shape to 2D shape are not supported. Do not use these Reshape operations.

- Be careful about 3 points as follows when creating the trained model of PyTorch.
  The error occurs if these points are not satisfied.
  1. Configuration of Python files
     Create Python file for model definition and for training as the separated files.
  2. Save method of the model file
     Do not specify "state_dict" when saving the model file.
  3. Translation by e-AI Translator
     Put Python file for model definition(*.py) and trained model file (*.pth) into the same folder.
     Specify trained model file (*.pth) as "Input Model File Name" of e-AI Translator.
  Refer to "3.5. How to create/save PyTorch model file" for the detail.

- When using CC-RX compiler or CC-RL compiler made by Renesas, the changes of settings may be required.

  [Addition of standard library]

  Build error may occur because the default setting of standard library "math.h" is disable. In such case, change the standard library "math.h" setting to enable.

  [Change setting to "C99"]

  Build error may occur because the default setting of C standard is "C89" or "C90". In such case, change the setting to "C99".

  [Allocation of heap memory]

  If the neural network has the following operations, e-AI Translator generates C source files including "malloc" operation, therefore allocate heap memory area.

  Max pooling with padding, Sigmoid, Softsign, Softplus, TanH

- When 8bit quantized model of TensorFlow Lite is used, there are cautions as follows.
  - Batch normalization layer can be used only when the order of layers is "Convolution - Batch Normalization - ReLU". When using Batch Normalization in other orders, evaluate the accuracy of inference enough after embedding because the accuracy may be changed.
  - Do not define input data variable as "const" variable, because input data variable is calculated in neural network processing.

- When "CMSIS_INT8" is used as "Output format", the model file which has branch structures in the neural network is not supported.

- When "CMSIS_INT8" is used as "Output format", the model file which has layer name by using "name" argument in the neural network is not supported. Therefore, do not use "name" argument when defining each layer.

  Example of "name" argument: tf.keras.layers.Dense(units = 16, activation = 'relu', name = 'Dense_1')

- When clicking "Browse" button of "Input Model File Name" again after specifying model file, different folder from model file folder may be opened. (Actually, correct model file is specified for translation.)

5. Error message

This chapter explains error message of e-AI Translator and how to solve each error.

When e-AI Translator generates the error, following dialog is displayed.

The log file shows error number "eAI-xxx", and is generated in the folder which is displayed in error dialog.

When the error occurs, confirm the solution from this error number.

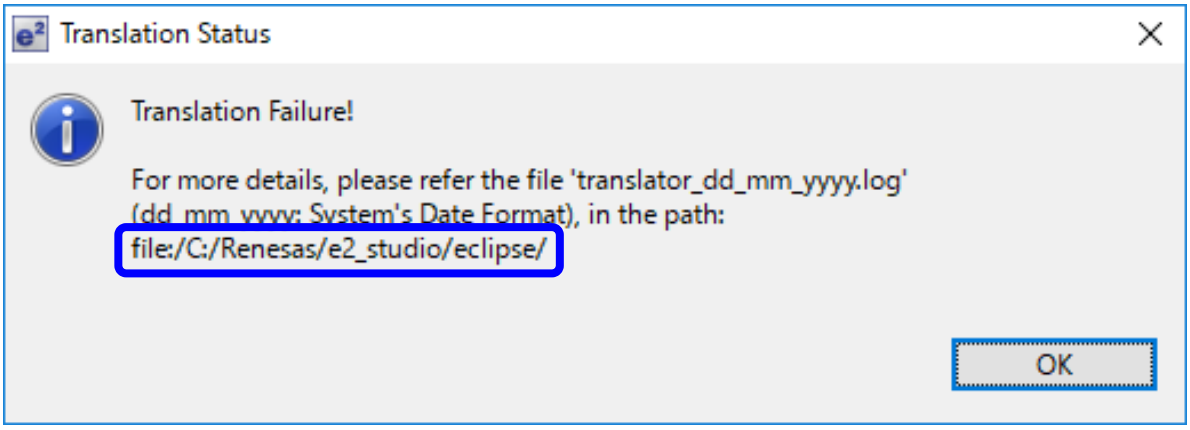

Figure 5.1 Folder of log file in error dialog

```
[2017-10-27 09:59:01,268 ERROR requirements.py:81 - log_exception() ]
eAI-301 : Framework Dependent File Not Found
Please check the contents of the Input Model Location. Ensure to have
required framework dependent files.
[2017-10-27 09:59:01,268    INFO requirements.py:82 - log_exception() ]
Translation FAILED!!
[2017-10-27 09:59:01,268    INFO requirements.py:51 - remove_object() ]
Uninitializing and cleaning up the translator object..
```

Figure 5.2 Error number in the log file

Table 5-1 List of error numbers and solutions (1/6)

Error category	Error number	Error message and the solution
About setup	eAI-101	Python not installed. The error when Python 3.7.7 is not installed. Install it by referring to "2.3. Installing Python 3.7.7".
	eAI-102	64bits Python version is not installed. The error when Python is not 64bit version but 32bit version. Install 64bit version of Python, confirm whether Windows on PC is 64bit or not.
	eAI-103	"tensorflow" package is not installed. The error when tensorflow package is not installed. Install tensorflow package by referring to "2.4. Installing the required Python Packages".
	eAI-104	"progressbar" package is not installed. The error when progressbar package is not installed. Install progressbar package by referring to "2.4. Installing the required Python Packages".
	eAI-105	"prettytable" package is not installed. The error when prettytable package is not installed. Install prettytable package by referring to "2.4. Installing the required Python Packages".
	eAI-106	"pytorch" package is not installed. The error when pytorch package is not installed. Install pytorch package by referring to "2.4. Installing the required Python Packages".

Table 5-2 List of error numbers and solutions (2/6)

Error category	Error number	Error message and the solution
About setup	eAI-107	”h5py” package is not installed. The error when h5py package is not installed. Install h5py package by referring to “2.4. Installing the required Python Packages”.
	eAI-108	“pycrypto” package is not installed. The error when pycryptodome package is not installed. Install pycryptdome package by referring to “2.4. Installing the required Python Packages”.
	eAI-110	”configparser” package is not installed. The error when configparser package is not installed. Install configparser package by referring to “2.4. Installing the required Python Packages”.
	eAI-111	”psutil” package is not installed. The error when psutil package is not installed. Install psutil package by referring to “2.4. Installing the required Python Packages”.
	eAI-112	"Keras" package is not installed. The error when Keras package is not installed. Install Keras package by referring to “2.4. Installing the required Python Packages”. However, when using TensorFlow-backend Keras, confirm the installation of TensorFlow correctly.
	eAI-113	"onnx" package is not installed. The error when onnx package is not installed. Install onnx package by referring to “2.4. Installing the required Python Packages”.

Table 5-3 List of error numbers and solutions (3/6)

Error category	Error number	Error message and the solution
About setting value	eAI-208	Invalid Input Shape Dimensions. The error when the specified characters to "Input Shape Dimensions" cannot be used. Only numerical characters and comma (,) can be used for this input area, and input them as one-byte characters.
	eAI-209	Input Shape Dimensions Not Found. The error when "Input Shape Dimensions" is not specified. Specify the input layer size of neural network to "Input Shape Dimensions".
	eAI-210	Incorrect Input Shape Dimensions. The error when the setting value of "Input Shape Dimensions" is not correct. Specify the size of the input layer of the neural network in the order of "C,H,W" or "D". C: Number of Channels; H: Input Height; W: Input Width; D: C*H*W;
	eAI-213	Model name is either an empty string or contains invalid characters. The error when specifying incorrect "Model Name". Confirm to fulfill these 2 conditions for the model name. - Specify only one-byte alphanumeric characters. - Specify within 8 characters.
	eAI-215	Input model filename is missing. The error when the file specified by "Input Model File Name" is not found. Confirm to specify the correct file.
	eAI-217	Unsupported library provided with Incorrect input_model_filename. The error when the setting is not matched between "Input Model File Name" and "Output Format". Confirm whether the model file of "Input Model File Name" is correct or not, and whether the setting of "Output Format" is correct or not.

Table 5-4 List of error numbers and solutions (4/6)

Error category	Error number	Error message and the solution
About framework	eAI-301	Framework Dependent File Not Found. The error when the files specified by "Input Model File Name" is not found. Confirm whether the specified file is correct or not, and whether all required files are existing or not by referring to procedure 5 of "3.1. Basic Procedure for Using the e-AI Translator". When using PyTorch, refer to "3.5 How to create/save PyTorch model file" also.
	eAI-302	pytorch Network Configuration Mismatch. The error when file extensions are not "*.pth" and "*.py" as trained model of PyTorch. Confirm whether the specified files are correct or not.
	eAI-303	Memory Error in Tensorflow. The error when the memory size of PC is not enough. Close PC application software, then try to use e-AI Translator again.
	eAI-304	Error in importing frozen graph. The error when e-AI Translator cannot get required parameters from trained model. If this error occurs, contact Renesas.
	eAI-305	Unsupported network. The error when using unsupported neural network. Confirm whether unsupported neural network is used or not by referring to "1.2. Types of Convertible Neural Networks" and "1.3. Supported API / unsupported API list".

Table 5-5 List of error numbers and solutions (5/6)

Error category	Error number	Error message and the solution
About framework	eAI-306	Unsupported Algorithm/Model The error when using unsupported model. Confirm whether unsupported layers are included or not in the trained model.
	eAI-307	Tensorflow Version mismatch The error when using the trained model which is created by unsupported version of TensorFlow. Confirm whether the version of TensorFlow is 2.4.0 or not.
	eAI-308	Keras model file versions mismatch The error when using the trained model which is created by unsupported version of Keras. Confirm the version of Keras as follows. - Standalone Keras: 2.4.3 - TensorFlow-backend Keras: 2.4.0-tf (Included in TensorFlow 2.4.0)
	eAI-309	Multiple Bias Error The error when using multiple bias addition in the trained model. Confirm whether multiple bias addition is existed or not in the training script. Example: The case to add bias to the output of "tf.keras.layers.conv2d" API. ("tf.keras.layers.conv2d" API has bias addition operation, therefore bias addition after the API is not required.)
	eAI-310	Unable to load keras/tf.keras model file The error when loading the model which use Lambda layer of Keras. If the PC used for training the neural network and the PC using the e-AI translator are different, use same version of Keras and Python on both PC. Refer to "4. Points of caution" for the detail.
	eAI-311	Input restricted layer The error when using activation function, clip function, normalization layer and pooling layer as the first data processing layer in neural network. Refer to "4. Points of caution" for the detail.

Table 5-6 List of error numbers and solutions (6/6)

Error category	Error number	Error message and the solution
About accessing files	eAI-401	File Open Error. The error when the files in the folder specified by "Input Model Location" cannot be opened. Confirm whether the folder access setting is inhibited or not, and whether other PC application software is using these files or not.
	eAI-402	File Creation Error. The error when the file creation is failed to the folder specified by "Translator Output Location". Confirm whether the folder write access setting is inhibited or not.
	eAI-403	File Overwrite Error. The error when overwriting the files in the folder specified by "Translator Output Location" is failed. Confirm whether the folder write access setting is inhibited or not, and whether other PC application software is using these files or not.
	eAI-404	File Not Found. The error when the files are not found in the folder specified by "Input Model Location". Confirm whether specified folder is correct or not.
Others	eAI-501	Uncaught Exception. The error which cannot be classified. When specifying PyTorch model, confirm whether the method to save the model file is correct or not. Refer to "3.5. How to create/save PyTorch model file" for the detail. If this error occurs, contact Renesas.

Revision History

Table 6-1 Revision History (1/3)

Rev.	Date	Description	
		Page	Summary
1.01	Sep 8, 2017	—	First edition issued
2.00	Nov 6, 2017	Overall	Change of the version of e-AI Translator from V1.0.0 to V1.0.1. Change of the GUI images and the descriptions of installation folder by this version up.
		P5	Addition of “Microsoft Visual C++ 2015 Redistributable” as required software to be installed.
		P7, P8	Addition of “1.3. Supported API / unsupported API list”.
		P9	Addition of “1.4. Changes of e-AI Translator from V1.0.0 to V1.0.1”
		P12, P13	Addition of the confirmation after installing Python.
		P15, P16, P21	Modification of Python version from 2.7 to 2.7.6 for installing Ubuntu environment.
		P19	Addition of character information which can be used for the folder name and path name as “Input Model Location”.
		P20, P21	Change of the procedure to convert the trained model from Ubuntu environment to Windows environment.
		P23	Addition of character information which can be used for the folder name and path name as “Translator Output Location”.
		P24	Addition of character information which can be used for “Input Shape Dimensions”. Addition of input format expansion for “Input Shape Dimensions”.
		P27	Addition of 1 caution.
		P28 - P31	Addition of “5. Error message”.
2.01	May 15, 2018	P5, P14	Addition of description about TensorFlow version to install.
		P20	Addition of description about “.prototxt” file when using Caffe.
3.00	July 24, 2018	Overall	Change of the version of e-AI Translator from V1.0.1 to V1.0.2. Change of the GUI images and the descriptions of installation folder by this version up.
		P5, P9, P14	Modification about supporting of TensorFlow version 1.8.0.
		P6, P7, P8	Modification about supporting of LRN (Local Response Normalization).
		P9	Addition about changes of e-AI Translator V1.0.2.
		P18	Addition of “RAM Usage(MB)” explanation.
		P21, P22	Modification about specification change of Python script files which are used when converting file format from Ubuntu environment to Windows environment.

Table 6-2 Revision History (2/3)

Rev.	Date	Description	
		Page	Summary
4.00	Feb 3, 2020	Overall	Change of the version of e-AI Translator from V1.0.2 to V1.4.0. Change of the GUI images and the descriptions of installation folder by this version up.
		P5, P7, P8, P21	Addition about supporting Keras framework.
		P6, P8	Addition about supporting of Batch Normalization.
		P14	Addition of required software, modification about the version of software.
		P9	Addition about changes of e-AI Translator V1.4.0.
		P15-P16	Modification of the explanation about Caffe installation, because Ubuntu14.04 is end of life.
		P31-P34	Addition about the procedure to use multiple neural networks.
		P36-P37	Addition / remove of cautions.
		P38-P44	Addition of error numbers and messages.
4.01	Mar 10, 2020	P5, P14	Correction of errors
6.00	Oct 27, 2020	Overall	Change of the version of e-AI Translator from V1.4.0 to V1.6.0. Change of the descriptions of installation folder by this version up.
		P9	Addition about supporting branch structures of neural networks
		P9, P36	Addition about RAM usage reduction.
		P5, P9	Addition about supporting e^2 studio 64bit version.
		P6-P9	Addition about supporting new layers.
		P37, P38	Addition and modification of cautions.
		P44	Addition and modification of error numbers and messages.
7.00	Feb 26, 2021	Overall	Change of the version of e-AI Translator from V1.6.0 to V2.0.0. Change of the descriptions of installation folder by this version up.
		P5-P9	Addition of PyTorch description. Update the support version of Keras and TensorFlow(tf.keras). Delete the description about Caffe and TensorFlow(tf.nn and tf.layers).
		P10-P15	Update the installation method.
		P16-P33	Update the usage.
		P34-P35	Addition/delete of points for caution.
		P37-P42	Addition/delete of error messages.
7.01	Mar 26, 2021	P14	Addition of version information to install “onnx”, “h5py”, “prettytable”, “progressbar33”, “configparser” and “psutil”.
8.00	Sep 21, 2021	Overall	Change of the version of e-AI Translator from V2.0.0 to V2.1.0. Change of the descriptions of installation folder by this version up.
		P5-P9	Addition of TensorFlow Lite 8bit quantization description. Update the support version of Keras and TensorFlow(tf.keras).
		P14, P15	Update the installation method.
		P17-P37	Update the usage. Addition of “3.6. How to create/save TensorFlow Lite model file”.
		P41, P44	Update/addition of error message.
8.01	Oct 8, 2021	P15	Modify the version of TensorFlow Lite and Keras.

Table 6-3 Revision History (3/3)

Rev.	Date	Description	
		Page	Summary
9.00	2022.3.18	Overall	Change of the version of e-AI Translator from V2.1.0 to V2.2.0. Change of the descriptions of installation folder by this version up.
		P5	Change of the version of e^2 studio.
		P7	Addition of support API (torch.nn.LocalResponseNorm).
		P9, P19, P24, P41, P42, P43	Addition of the descriptions about “CMSIS_INT8” as source file output format.
		P9, P39	Addition of the descriptions about support model of TensorFlow Lite.
		P44, P45	Update/remove of points for caution.

e-AI Translator V2.2.0 User's Manual

Publication Date: Rev.1.01 Sep. 08, 17
Rev.9.00 Mar. 18, 22

Published by: Renesas Electronics Corporation

e-AI Translator V2.2.0

Renesas Electronics Corporation

R20UT4135EJ0900