User's Manual

e-AIトランスレータ　V2.3.0

ユーザーズマニュアル

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.10.01　2022.12

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

目 次

ページ

用語説明

本書で使用する用語は、以下に示すように定義して使用します。

e-AI
Embedded Artificial Intelligence(組み込み向け人工知能)の略称です。

e-AIトランスレータ
オープンソースのディープラーニングフレームワークである PyTorch、Keras、TensorFlow(tf.keras)、TensorFlow Liteの学習済みニューラルネットワークの推論処理部分をMCU/MPU統合開発環境用ソースファイルへ変換し、インポートするツールです。当社製MCU/MPU統合開発環境”e^2 studio”用のプラグインとして動作します。

ニューラルネットワーク
コンピュータに学習能力を与えるために作られた、人間の脳の神経回路の仕組みを模したモデルです。

ディープラーニングフレームワーク
ニューラルネットワークの定義や学習を行うためのソフトウェアです。

ホストマシン
e-AIトランスレータが動作するパーソナルコンピュータを指します。

ユーザシステム
MCU/MPUを使用したお客様のアプリケーションシステムを指します。

ユーザプログラム
MCU/MPU上で動作するアプリケーションプログラムを指します。

1. 概 要

e-AIトランスレータとは、オープンソースのディープラーニングフレームワークであるPyTorch、Keras、TensorFlow(tf.keras)、TensorFlow Liteの学習済みニューラルネットワークの推論処理部分をMCU/MPU統合開発環境”e² studio”用ソースファイルへ変換し、インポートするツールです。
本書では、e-AIトランスレータのインストール方法、使用方法、注意事項等について説明します。
e-AIトランスレータは、当社製MCU/MPU統合開発環境”e² studio”用のプラグインとして動作します。

1.1. 動作環境

e-AIトランスレータの動作環境、インストールが必要なソフトウェアは以下の通りです。

表 1-1 動作環境

項目	ソフトウェア名/対応バージョンなど
対応済みディープラーニングフレームワーク[注1]	PyTorch[注2], Keras[注3], TensorFlow(tf.keras API)[注4], TensorFlow Lite[注5]
動作環境	Windows 8.1, Windows10 いずれも64bit版のみ対応
インストールが必要なソフトウェア	Python 3.7.7
	Python Packageから以下のパッケージのインストールが必要です[注6] Torch, TorchVision, TorchSummary, onnx, TensorFlow, progressbar33, prettytable, h5py, pycryptodome, configparser, psutil, Keras, tflite
	Microsoft Visual Studio 2015 Visual C++ 再頒布可能パッケージ
	e² studio 2021-10以上 Flexible Software Package 3.5以上 3.71以下(RAファミリの場合)
対応デバイス[注7]	RL78ファミリ、RXファミリ、RAファミリ、REファミリ、RZファミリ、Renesas Synergy™ マイクロコントローラ

注 1：本バージョンでは Caffe、TensorFlow(tf.nn や tf.layers 等の API)はサポートしていません。これらのフレームワークをご使用の方は e-AI トランスレータ V1.6.0 をご使用ください。
注 2：PyTorch 1.5.1 で作成した学習済みモデルで動作確認しています。
注 3：単体版 Keras 2.4.3 で作成した学習済みモデルで動作確認しています。
注 4：TensorFlow 2.4.0 に同梱の TensorFlow-backend 版 Keras 2.4.0-tf で作成した学習済みモデルで動作確認しています。
注 5：TensorFlow 2.4.0 の TFLiteConverter で 8bit 量子化した学習済みモデルで動作確認しています。対応している 8bit 量子化の詳細は「3.6.TensorFlow Lite モデルファイルの作成と保存」を参照してください。
注6：Python Packageのインストールは「2.4. Python Packageのインストール」を参照してください。
注7：ニューラルネットワークの大きさによってROM/RAM使用量が増加します。メモリサイズの大きなデバイスの使用を推奨します。

1.2. 変換可能なニューラルネットワークの種類

本バージョンのe-AIトランスレータで変換可能なニューラルネットワークの種類を表1-2に示します。
対応可能なニューラルネットワークは、今後のバージョンアップで順次追加予定です。
注意：未対応のニューラルネットワークは、正しく変換できません。
変換するニューラルネットワークが対応済みかどうか必ずご確認ください。

表 1-2 変換可能なニューラルネットワークの種類

項目	対応内容
対応済みアルゴリズム[注]	CNN (Convolution Neural Network), Auto Encoder, 他
対応済み畳み込み層	Convolution, Deconvolution (Transposed convolution), Depthwise Convolution
対応済みプーリング層	Max pooling, Average pooling, Global max pooling, Global average pooling
対応済み全結合層	Fully connected (Inner product)
対応済み正規化層	Batch normalization
対応済み活性化関数	ReLU (Rectified Linear Unit), Sigmoid, Tanh, ReLU6, Leaky ReLU, Clip, Hard sigmoid, Softsign, Softplus
対応済み分類関数	Softmax

注：本バージョンではRNN(Recurrent Neural Network)には対応していません。

1.3. サポートAPI一覧

本バージョンのe-AIトランスレータでサポート済みのPyTorch API、Keras/TensorFlow(tf.keras) API、TensorFlow Liteの8bit量子化モデルを表1-3に示します。
未対応のAPIやLayer typesに関しては、今後のバージョンアップで順次対応予定です。

表 1-3 サポートAPI一覧(1/2)

PyTorch API	Keras/TensorFlow(tf.keras) API	TensorFlow Lite 8bit quantized Model
torch.nn.Conv2d[注1]	keras.layers.Conv2D tf.keras.layers.Conv2D	Support
torch.nn.ConvTranspose2d	keras.layers.Conv2DTranspose tf.keras.layers.Conv2DTranspose	Support
torch.nn.Conv2d[注1]	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[注2]	keras.activations.softmax tf.keras.activations.softmax keras.layers.Softmax tf.keras.layers.Softmax keras.layers.Activation(“softmax”) tf.keras.layers.Activation(“softmax”)	Support

注1：引数の設定によりConvolution処理およびDepthwise Convolution処理として使用可能です。ただし、Separable Convolution処理はサポートしていません。

注2：PyTorchのSoftmax関数は対応済みですが、LogSoftmax関数には未対応です。
PyTorchのモデルを使用する場合、このような未対応関数を含むニューラルネットワークを変換してもエラーが発生しない場合があります。このため、入力したニューラルネットワークと出力ファイル”dnn_compute.c”を比較して問題なく変換できているかを確認してください。

表 1-4 サポートAPI一覧(2/2)

PyTorch API	Keras/TensorFlow(tf.keras) API	TensorFlow Lite 8bit quantized Model
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
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注
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	-	-

注：TensorFlow Liteの8bit量子化モデルでは、Convolution - Batch Normalization - ReLUの順序でBacth Normalizationを使用してください。その他の順序で使用した場合、推論誤差が発生することがあるため、組み込み後に推論精度を十分に確認してください。

なお、以下の処理は学習時のみ使用されるものなので、e-AIトランスレータの変換では無視されます。

・Loss関数

・Dropout関数

1.4. e-AIトランスレータV2.2.0からV2.3.0の変更点

e-AIトランスレータV2.2.0からV2.3.0の変更点は次の通りです。

・8ビット量子化されたTensorFlow Lite形式モデルの変換に関して、以下を改善しました。
- 分岐構造を持つニューラルネットワークの変換に対応しました。
- 出力形式「CMSIS_INT8」のRAM使用量を削減し、「C_INT8」と同等のRAM使用量としました。
- 出力形式「C_INT8」の実行速度を高速化しました。

改善後の出力形式「CMSIS_INT8」と「C_INT8」の比較は以下の通りです。
今回の改善によりRA、RXファミリをご使用の場合は出力形式「CMSIS_INT8」の使用を推奨します。

出力形式	対応デバイス	実行速度	RAM使用量	備考
CMSIS_INT8	RA、RXファミリ	高速	標準	RA、RXファミリの推奨設定
C_INT8	全てのファミリ	標準	標準	ー

・ライセンスファイルのフォーマットを変更しました。
(旧バージョンのe-AIトランスレータ用ライセンスファイルは使用できません。ご注意ください。)

2. インストール方法

本章では、e-AIトランスレータのインストール方法について説明します。
e-AIトランスレータは当社製MCU/MPU統合開発環境”e² studio”用プラグインです。e² studioのインストールを行ってから以下の手順でe-AIトランスレータのインストールを行ってください。

2.1. e-AIトランスレータのインストール

1.WEBサイトからダウンロードしたインストーラのzipファイルを解凍します。
2.解凍後のファイル”setup.exe”をダブルクリックします。
3.以降、インストーラの指示に従ってインストールを進めます。
ただし、インストール先のフォルダは、e² studioのインストールフォルダとする必要があります。
※e² studioのインストール先フォルダを変更していなければ、e-AIトランスレータのインストール先フォルダも変更する必要はありません。

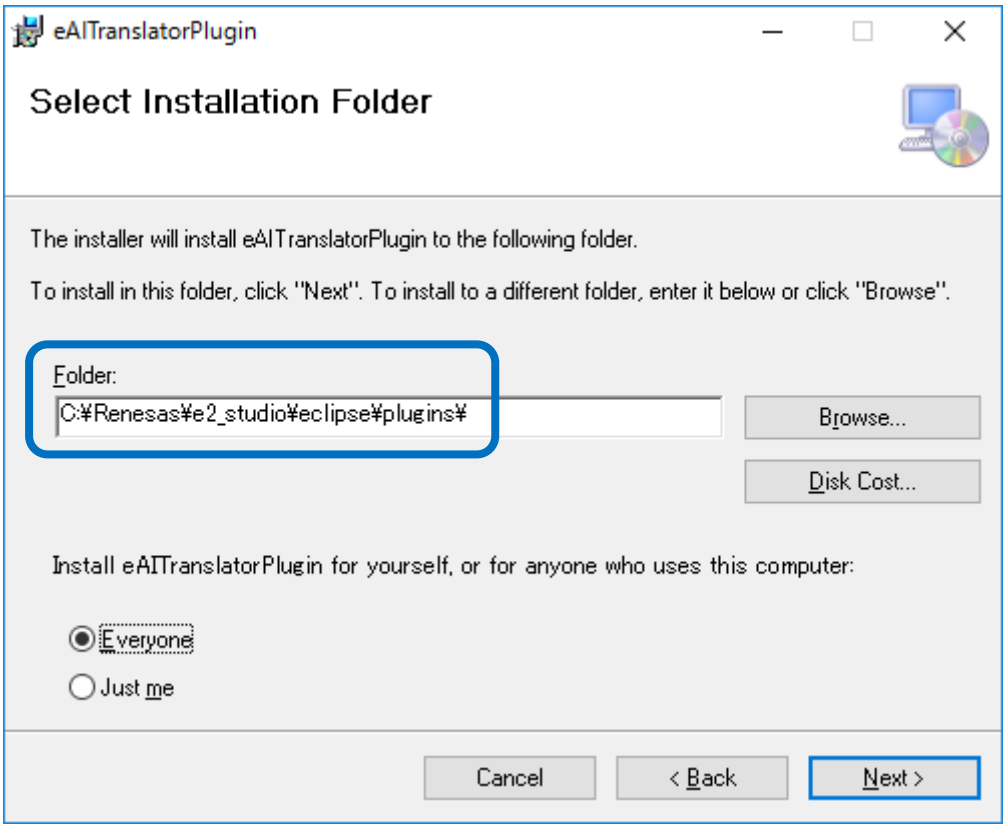

図2.1　インストールフォルダの指定画面

2.2. e-AIトランスレータ　ライセンスファイルのコピー

WEBサイトよりダウンロードした”license.txt”を以下のフォルダへ上書きコピーします。
C:¥Renesas¥e2_studio¥eclipse¥plugins¥com.renesas.eaitranslator_2.3.0
※手順2.1でインストールフォルダを変更している場合、コピー先フォルダをインストールフォルダに合わせて変更してください。

2.3. Python3.7.7のインストール

1.以下のWEBサイトよりPython 3.7.7をダウンロードします。
https://www.python.org/downloads/release/python-377/
”Windows x86-64 executable installer”

2.インストーラをダブルクリックして、インストールを開始します。
”Install Now”をクリックします。
以降、インストーラの指示に従ってインストールを進めます。
備考1：”Add Python 3.7 to PATH”の設定はチェックを付けていても、チェックを付けていなくてもどちらでも問題ありません。
備考2：Pythonのライセンスについては、インストールフォルダにある”Lisence.txt”をご一読ください。
[Pythonのインストールフォルダ]
C:¥Users¥*<windows-user-name>*¥AppData¥Local¥Programs¥Python¥Python37
※”*<windows-user-name>*”にはWindowsにログオン時のユーザ名

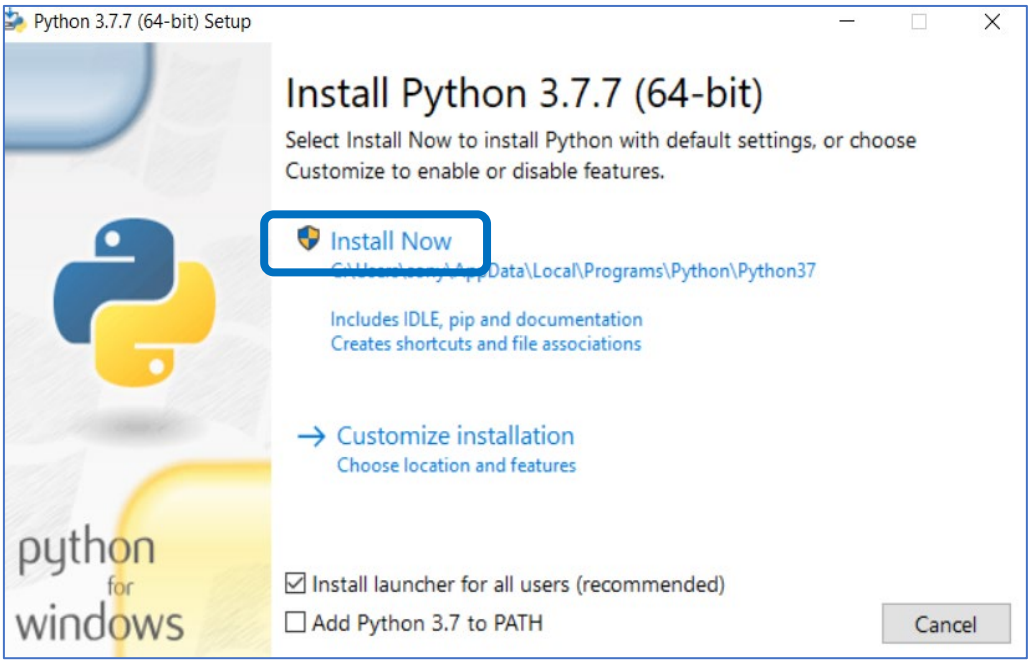

図2.2　Pythonのインストール画面

3.インストール完了後、以下の手順で正常にインストールされていることを確認します。
・Windowsのコマンドプロンプトを開きます。
・コマンド”py -3.7 -V”を入力し、実行します。
・コマンドの実行結果が”Python 3.7.7”となった場合、正常にインストールされています。
(手順4は不要です。)

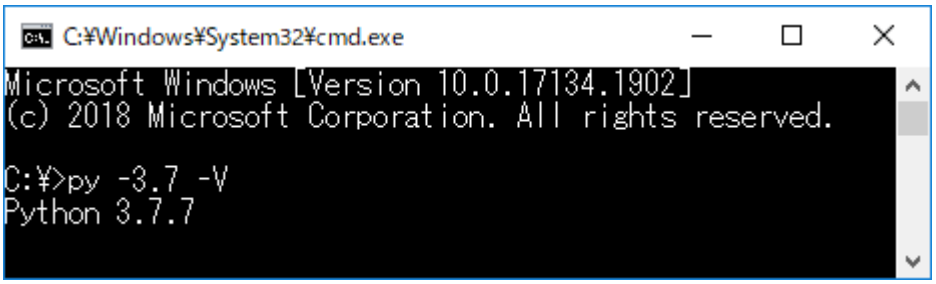

図2.3 Pythonのインストール確認

4.手順3でのコマンド実行結果がエラーとなった場合、以下の手順でシステム環境変数を編集してください。ここでは、Windows 10の画面イメージを使用して説明しています。
・Windowsのシステムのプロパティを開き、”詳細設定”タブの”環境変数”ボタンをクリックします。

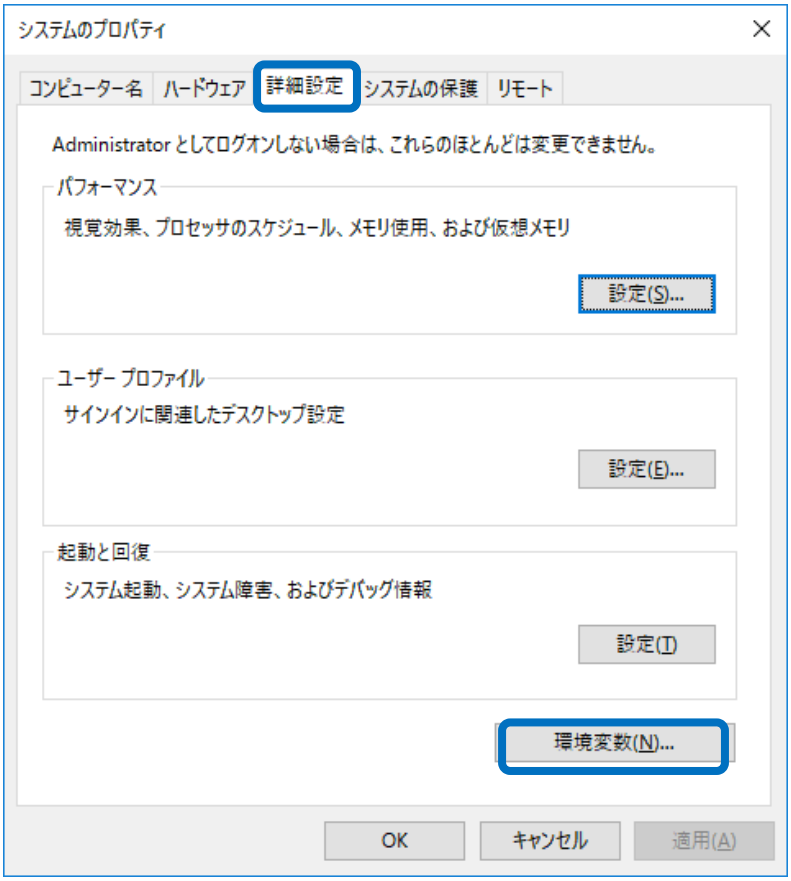

図2.4 システムのプロパティ

・システム環境変数の一覧から”Path”を選択し、”編集”ボタンをクリックします。

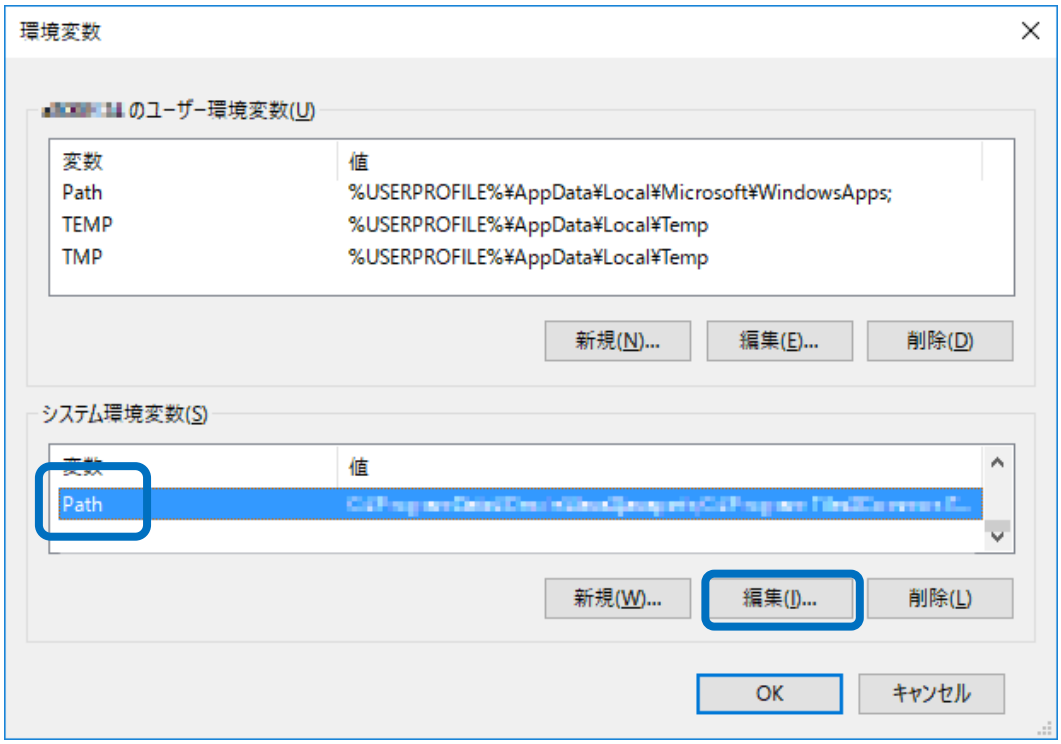

図 2.5　環境変数の設定 1

・”新規”ボタンを押して、環境変数名に”C:¥Windows”を追加します。”OK”ボタンを押して各ダイアログを閉じた後で、再度手順3の確認を行います。

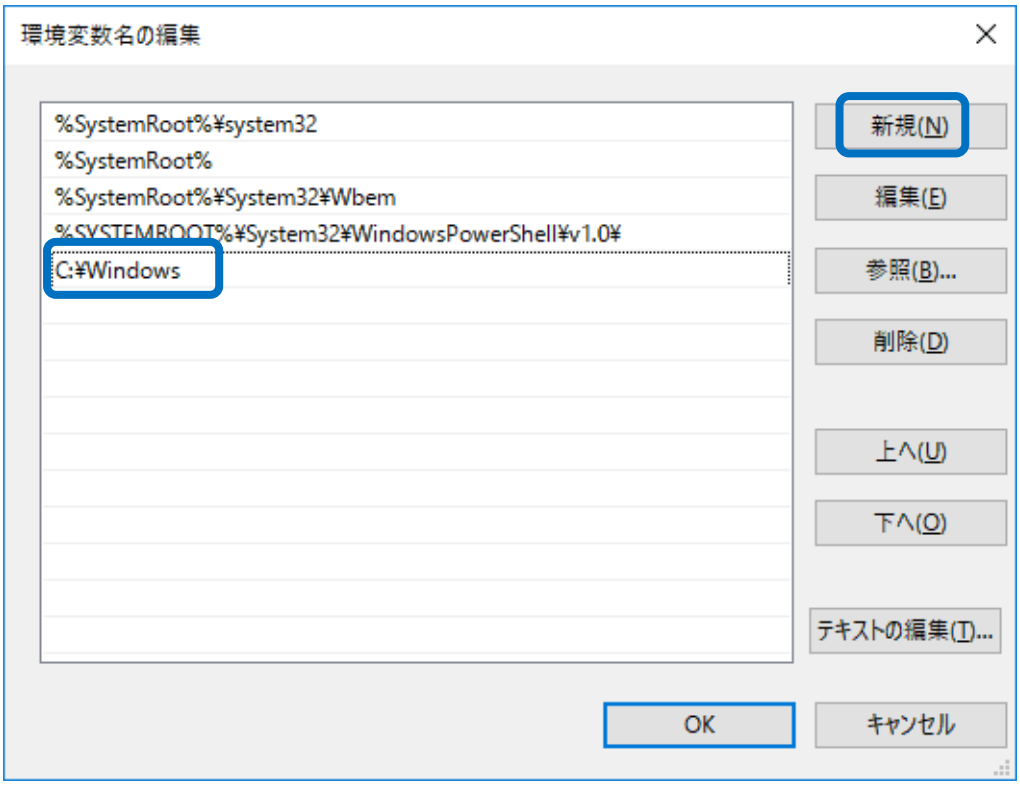

図 2.6　環境変数の設定 2

2.4. Python Packageのインストール

・TensorFlowのインストール

1.Windowsのコマンドプロンプトを起動します。

2.次のコマンドを実行します。

cd C:¥Users¥*<windows-user-name>*¥AppData¥Local¥Programs¥Python¥Python37¥Scripts

※"*<windows-user-name>*"にはWindowsにログオン時のユーザ名を入力します。

※Python3.7.7のインストールフォルダ「Python37」を指定します。

特に複数のバージョンのPythonがインストールされている場合、フォルダの指定を間違えないようご注意ください。

3.次のコマンドを実行します。

pip3 install --upgrade tensorflow==2.4.0

※コマンドが完了するまで最大10分ほどかかります。

上記のインストール完了後に、引き続き次のコマンドを実行して各Packageをインストールします。

・Torch, TorchVisionのインストール

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

・TorchSummaryのインストール

pip3 install torchsummary

・Onnxのインストール

pip3 install onnx==1.7.0

・ProgressBarのインストール

pip3 install progressbar33==2.4

・Prettytableのインストール

pip3 install prettytable==1.0.1

・h5pyのインストール

pip3 install h5py==2.10.0

・pycryptodomeのインストール

pip3 install pycryptodome==3.7.2

・configparserのインストール

pip3 install configparser==5.0.1

・psutilのインストール

pip3 install psutil==5.7.2

・Kerasのインストール(単体版Kerasの使用時のみ)
pip3 install keras==2.4.3

・TensorFlow Liteのインストール
pip3 install tflite==2.4.0

2.5. Microsoft Visual Studio 2015 Visual C++ 再頒布可能パッケージのインストール

1.以下のWEBサイトより、Microsoft Visual Studio 2015 Visual C++ 再頒布可能パッケージの64ビット版用のインストーラ” vc_redist.x64.exe”をダウンロードします。
https://www.microsoft.com/ja-JP/download/details.aspx?id=52685

2.インストーラをダブルクリックして、インストールを開始します。
以降、インストーラの指示に従ってインストールを進めます。

3. 使用方法

本章では、e-AIトランスレータの使用方法について説明します。

3.1. 基本的な使用手順

1. e² studioを起動します。
   プロジェクトとして新規プロジェクトを作成します。もしくは既存のプロジェクトを開きます。

2. e-AIトランスレータを起動します。
   “TR”と書かれたボタンを押します。もしくは”Translator”メニューより”Configure and Translate”を選択します。

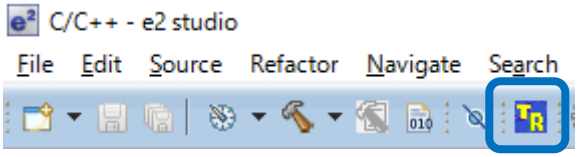

図3.1 e-AIトランスレータの起動ボタン

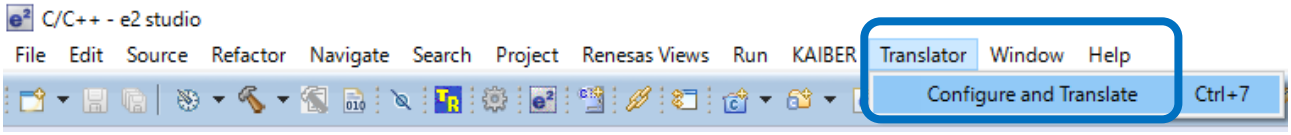

図3.2 e-AIトランスレータの起動メニュー

3. 下図のようにe-AIトランスレータが起動します。

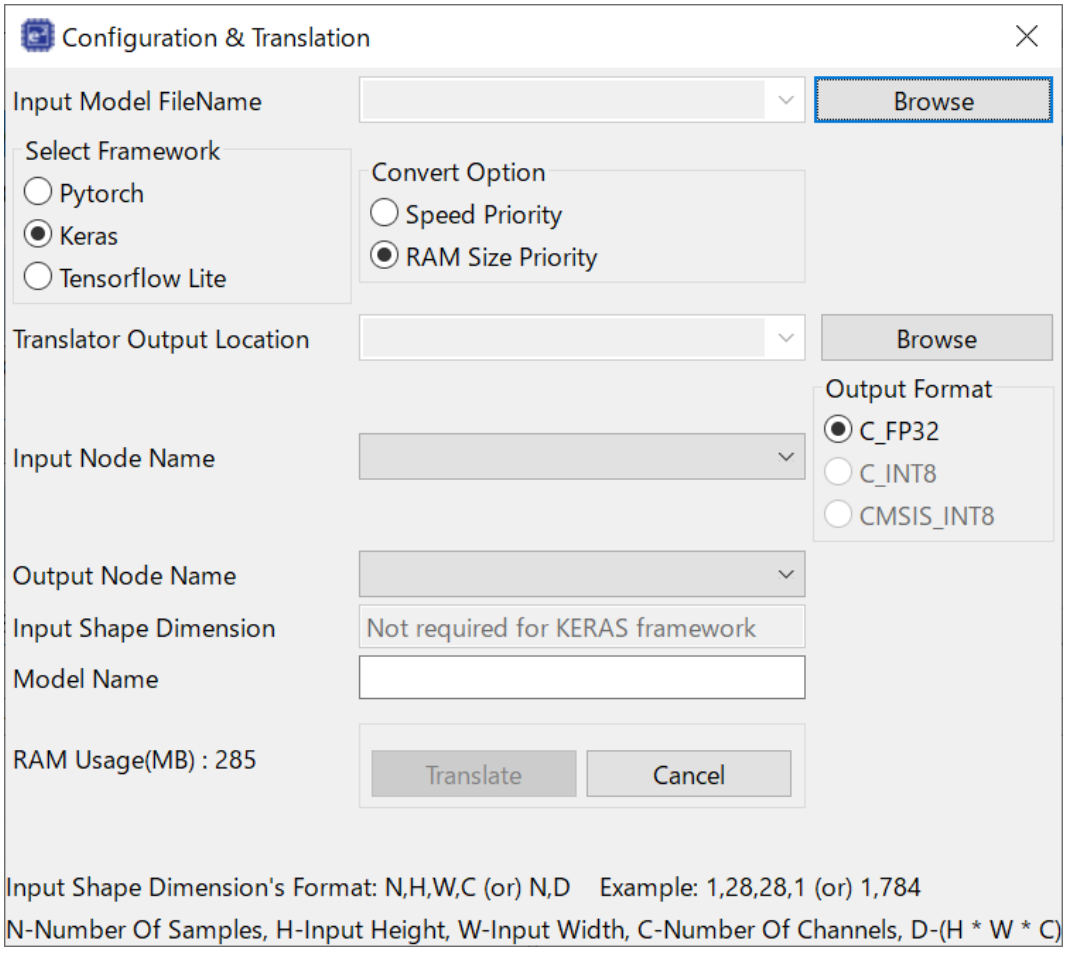

図3.3 e-AIトランスレータ

表 3-1 e-AIトランスレータ 各エリアの説明

項目名	説明
Input Model File Name	PyTorch、Keras、TensorFlow(tf.keras), TensorFlow Liteで作成した学習済みモデルが存在するフォルダを指定します。
Select Framework	学習済みモデルの種類を指定します。 TensorFlow(tf.keras)を使用している場合は”Keras”を選択します。
Convert option	重みデータの配置場所をROMとするかRAMとするかを変更します。RAMサイズを優先する場合、重みデータがROMに配置されます。実行速度を優先する場合、重みデータがRAMに配置されます。
Translator Output Location	変換結果として出力されるソースファイル、ヘッダファイルの出力先フォルダを指定します。e^2 studioプロジェクトのソースフォルダを指定してください。
Input Node Name	Keras、TensorFlow(tf.keras)の学習済みモデルを指定した場合、入力ノード名と出力ノード名が自動で表示されます。
Output Node Name	
Output Format	以下の3つからソースファイルの形式が選択できます。 ・C_FP32： PyTorch、TensorFlow(tf.keras)形式のモデルをソースファイルへ変換する際、”C_FP32”が自動で選択されます。 ・C_INT8： TensorFlow Lite形式のモデルをソースファイルに変換します。全てのMPU/MCUで使用可能です。 ・CMSIS_INT8： TensorFlow Lite形式のモデルをソースファイルに変換する際、CMSISライブラリを活用して推論速度を高速化します。RA、RXファミリのみ使用可能です。
Input Shape Dimension	ニューラルネットワークの入力層のサイズを設定します。フレームワークにPyTorchを使用した場合のみ設定が必要です。
Model Name	複数のニューラルネットワークを使用する場合のみ設定するエリアです。詳細は「3.2.複数ニューラルネットワークの変換」を参照してください。
RAM Usage (MB)	トランスレート時、PCのRAM使用量を参考情報として表示します。
Translateボタン	AIの学習済みモデルをソースファイル、ヘッダファイルに変換するためのボタンです。

4. “Select Framework”の設定を行います。
学習済みモデルの種類を指定します。
・Pytorch：PyTorchの学習済みモデルを使用
・Keras：Keras、もしくはTensorFlow(tf.keras)の学習済みモデルを使用
・Tensorflow Lite：Keras、もしくはTensorFlow(tf.keras)の学習済みモデルを
TensorFlow Liteで8bit量子化した学習済みモデルを使用

注意：”Input Moel File Name”で学習済みモデルを指定する前に、“Select Framework”を設定してください。

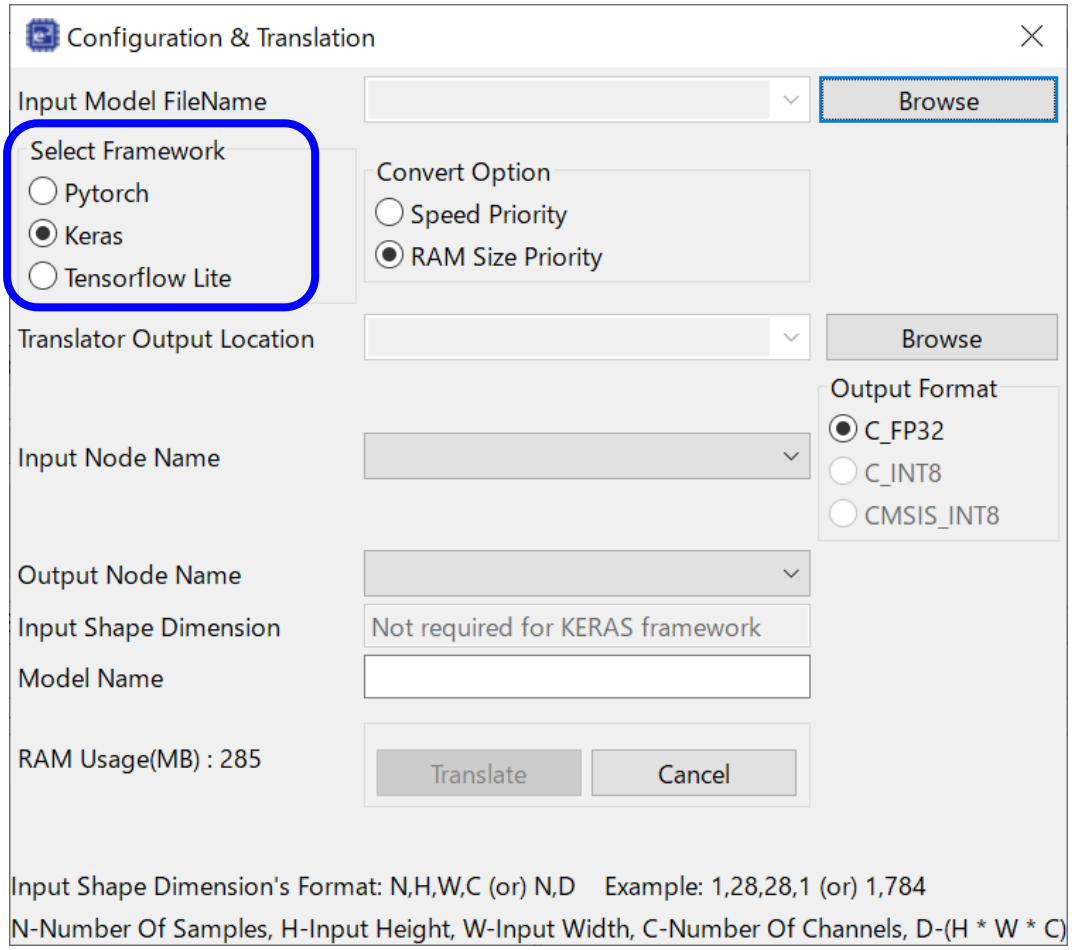

図3.4 “Select Framework”の設定

5. “Browse”ボタンを押して、”Input Model File Name”を設定します。
PyTorch、Keras、TensorFlow(tf.keras)、TensorFlow Liteで作成した学習済みモデルが存在するフォルダを指定します。
フォルダ名、パス名には半角英数文字のみ使用可能です。全角文字は使用しないでください。
PyTorch、Keras、TensorFlow(tf.keras)のそれぞれで入力が必要なファイルは以下の通りです。

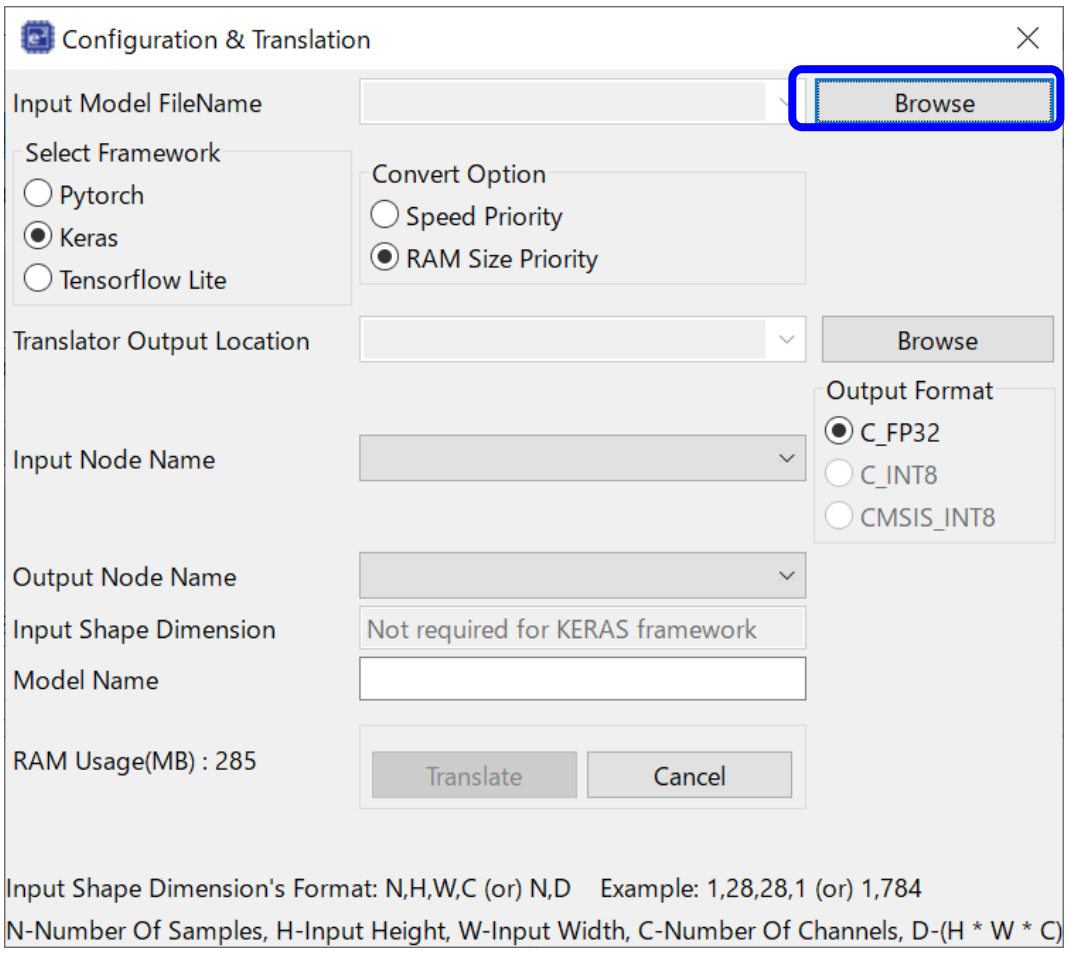

図3.5 “Input Model File Name”の設定

[PyTorchの場合]
以下の2つのファイルを同じフォルダへ格納し、拡張子が”*.pth”のファイルを指定します。
2つのファイルの準備方法の詳細については、「3.5 PyTorchのモデルファイル作成と保存」を参照してください。
- PyTorchで学習結果を保存した際に生成される、拡張子が”*.pth”のファイル
- ニューラルネットワークの構造を定義した、拡張子が”*.py”

[Keras、TensorFlow(tf.keras)の場合]
HDF5形式で保存した学習済みモデル(拡張子が”*.h5”)のファイルを指定します。

[TensorFlow Liteの場合]
TensorFlow Lite形式の学習済みモデル(拡張子が”*.tflite”)のファイルを指定します。

6. “Convert option”の設定を行います。
   AIの学習済みモデルを変換する際に何を優先するかを指定します。
   ・Speed priority：
   重みデータをRAMに配置することで、変換後のプログラムの実行速度を優先します。
   ・RAM size priority(推奨[注])：
   重みデータをROMに配置することで、変換後のプログラムのRAM使用量を優先します。

   注：一般的に重みデータのサイズは大きいため、”RAM size priority”の使用を推奨します。

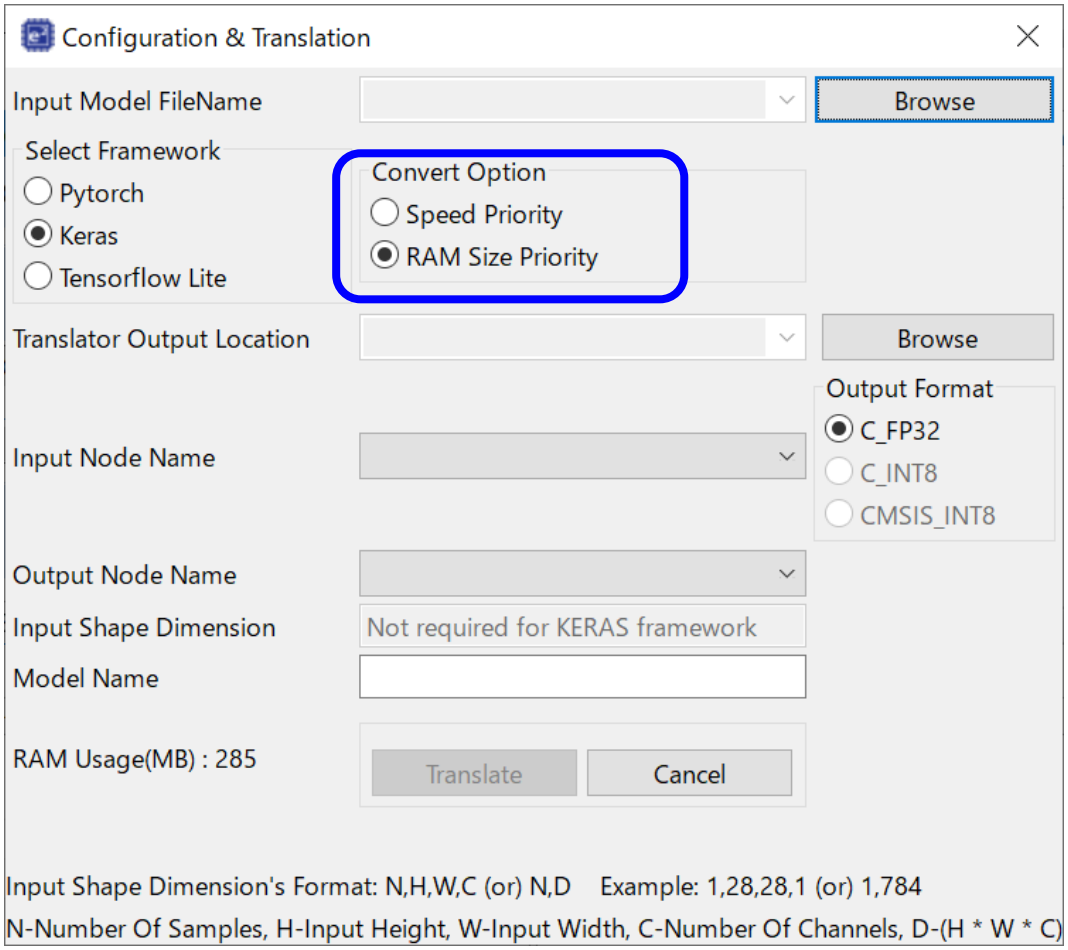

図3.6 ”Convert option”の設定

7. “Browse”ボタンを押して、”Translator Output Location”を設定します。
変換結果として出力されるソースファイル、ヘッダファイルの出力先フォルダを指定します。
フォルダ名、パス名には半角英数文字のみ使用可能です。全角文字は使用しないでください。
e^2 studioのソースファイルを登録しているフォルダ(例：プロジェクトフォルダ内にある”src”フォルダ)を指定してください。

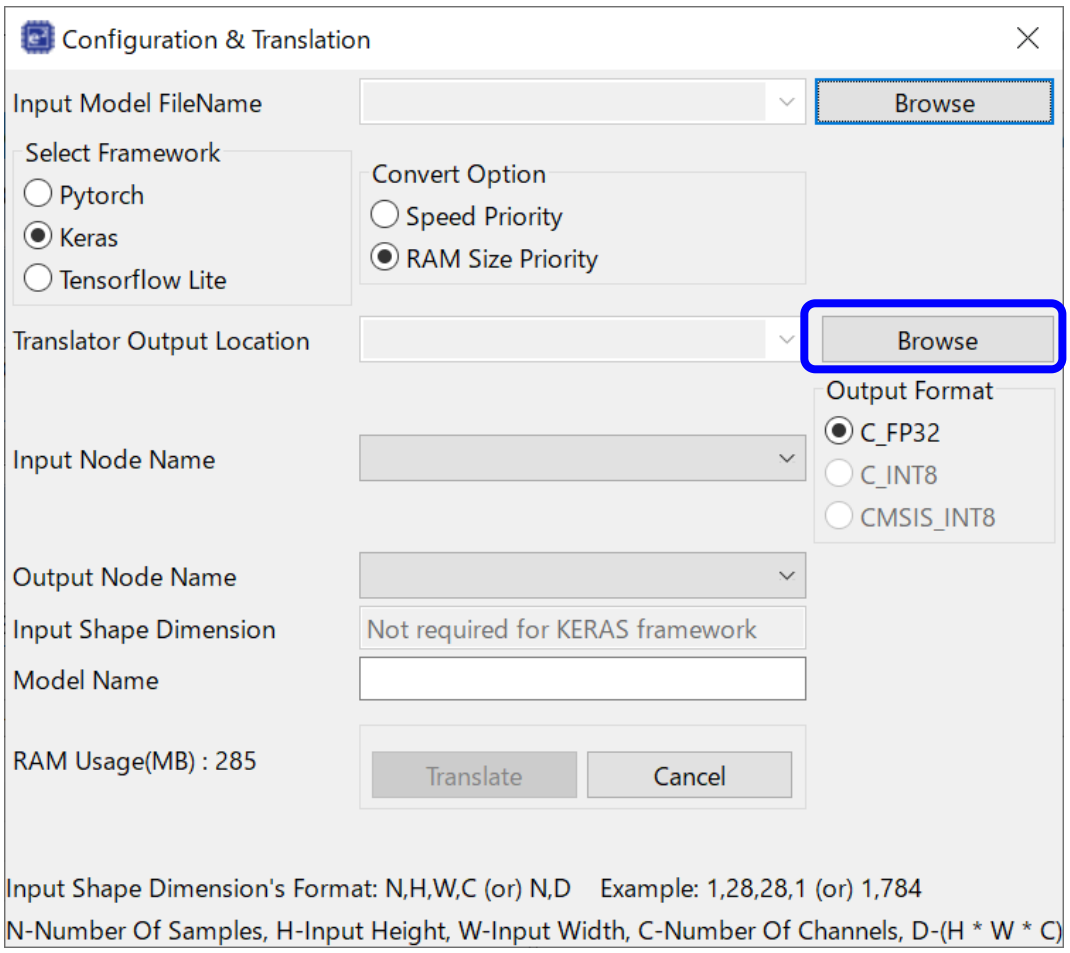

図3.7 “Translator Output Location”の設定

8. ”Input Node Name”、 ”Output Node Name”の設定を確認し、”Output Format”を設定します。

・”Input Node Name”および”Output Node Name”：
Keras、TensorFlow(tf.keras)の学習済みモデルを指定した場合、入力ノード名と出力ノード名が自動で表示されます。基本的に設定の変更は不要です。ただし、自動で選択された設定値に間違いがある場合は、手動で設定値を変更してください

・”Output Format”：
“Input Model File Name”で指定した学習済みモデルの形式に応じて、出力ソースファイルの種類を選択します。CMSIS_INT8 を選択する場合は「3.7 CMSIS_INT8 の使用方法」も参照してください。
・PyTorch、Keras、TensorFlow(tf.keras)のモデルを指定した場合：
”C_FP32”が自動で選択されます。
・TensorFlow Lite で 8bit 量子化したモデルを指定した場合：
以下の表を参照して“CMSIS_INT8”もしくは”C_INT8”を選択します。

出力形式	対応デバイス	実行速度	RAM使用量	備考
CMSIS_INT8	RA、RXファミリ	高速	標準	RA、RXファミリの推奨設定
C_INT8	全てのファミリ	標準	標準	ー

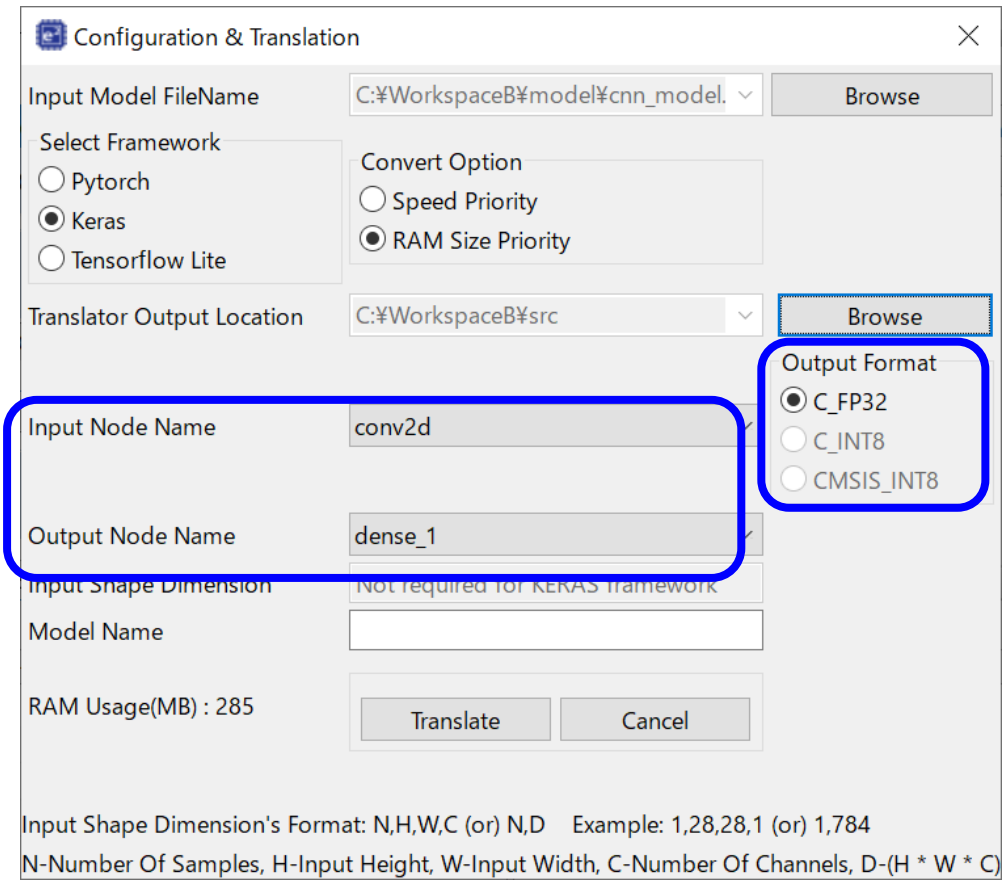

図3.8 ”Input Node Name”, ”Output Node Name”の確認と”Output Format”の設定

9. ”Input Shape Dimensions”を設定します。(PyTorchの場合のみ)
   ニューラルネットワークの入力層のサイズを設定します。フレームワークにPyTorchを使用した場合のみ設定が必要です。”C, H,W”、もしくは”D”の形式で入力します。
   入力可能な文字は数値とカンマ(,)のみです。数値およびカンマは半角文字で入力してください。
   C：Number of Channels 、H：Input Height、W：Input Width
   D：C×H×W

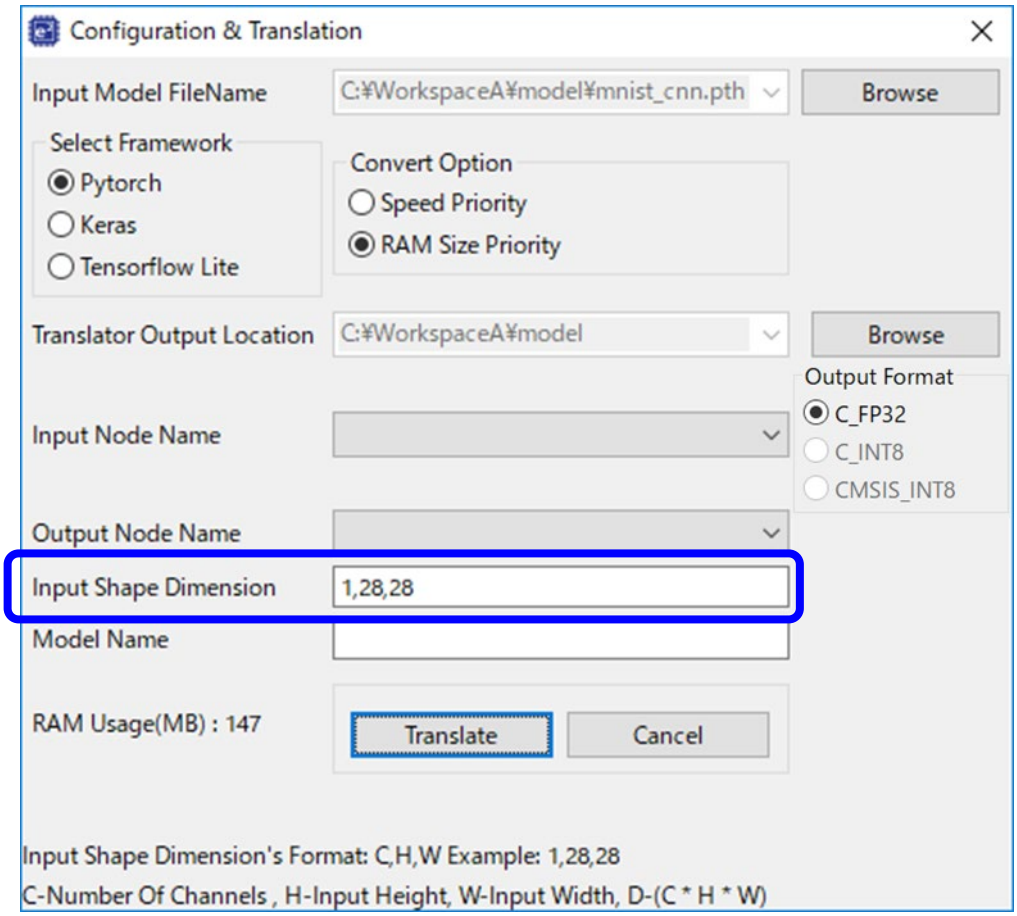

図3.9 “Input Shape Dimensions”の設定

10. ”Translate”ボタンを押して、Cソースファイルとヘッダファイルへの変換を行います。
※”Model Name”は、複数のニューラルネットワークを使用する場合に設定が必要なエリアです。
複数使用しない場合は空欄としてください。また、実際の使用方法については「3.2.複数ニューラルネットワークの変換」を参照してください。

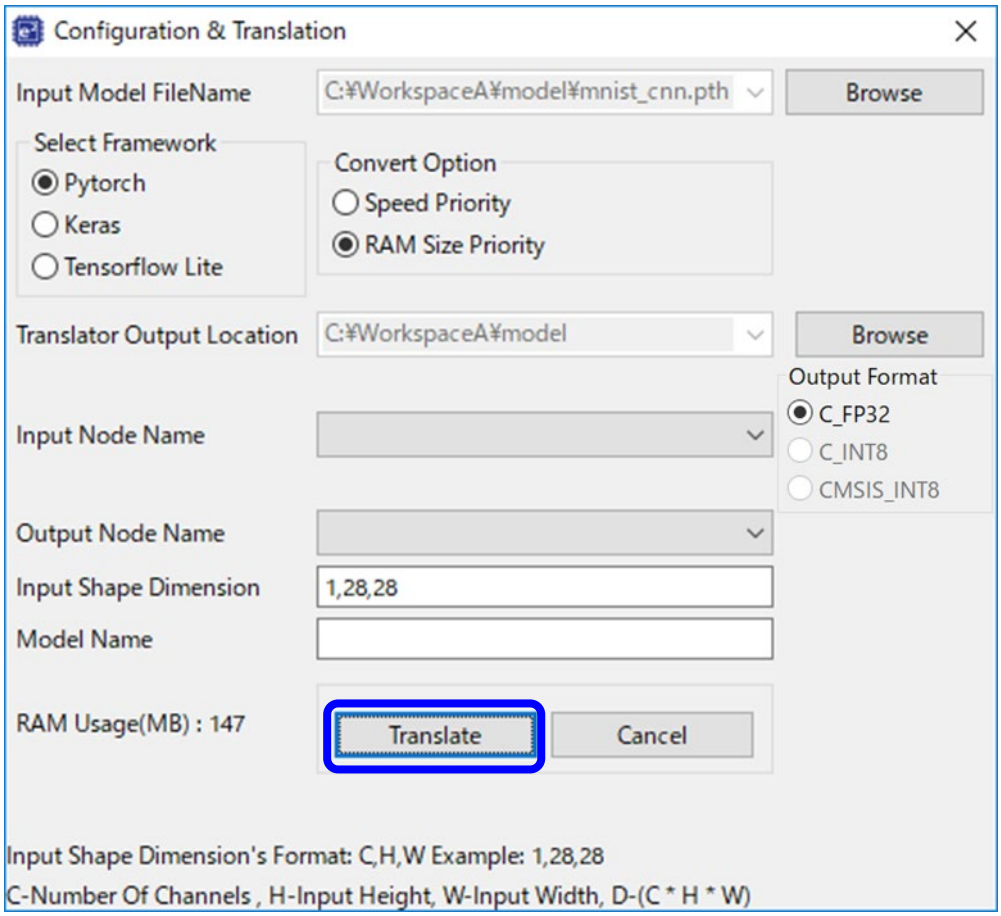

図3.10 “Translate”ボタン

11. 変換されたCソースファイルとヘッダファイルは、”src”フォルダ内の”Translator”フォルダに出力されます。
    ヘッダファイルをビルドに含めるために、ビルドツールの設定でこのフォルダをインクルードファイルパスに追加してください。

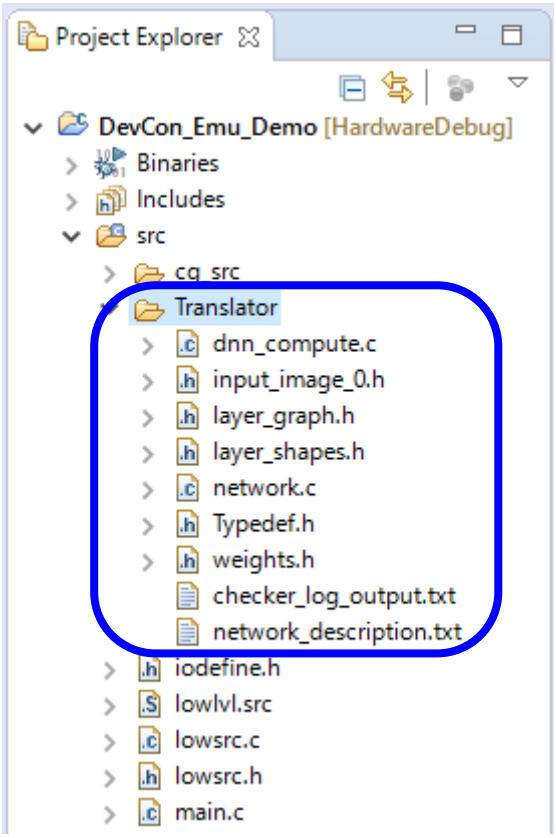

図3.11　変換後の出力フォルダとファイル

12. AIの推論処理を使用するには、”dnn_compute.c”ファイル内の”dnn_compute”関数を呼び出してください。
”dnn_compute”関数の仕様は以下の通りです。

表3-2. dnn_compute関数の仕様

関数名	dnn_compute	
概要	AIの推論処理を実行	
関数インターフェース注	TsOUT* dnn_compute(TsIN* input, TsInt *errorcode) ※TsOUT、TsIN、TsIntの定義については”Typedef.h”を参照してください。	
戻り値	推論結果	
引数	input	推論するデータを入力します。
	errorcode	malloc命令によるメモリ確保エラーの発生を判断する変数です。 ・0：メモリ確保エラーが発生していない ・1：メモリ確保エラーが発生 "dnn_compute"関数の先頭で0に初期化されます。 このため、関数呼び出し時の初期値設定は不要です。 "dnn_compute"関数終了後に"errorcode"が1となっている場合、エラーが発生しているため推論結果を使用しないでください。malloc命令のメモリ確保に必要なヒープメモリを増やした上で、再度プログラムをビルドしてください。 なお、malloc命令によるメモリ確保が実施されるのは、CC-RXコンパイラ、もしくはCC-RLコンパイラ使用時のみです。その他のコンパイラを使用している場合、メモリ確保エラーは発生しません。

注：RL78で本関数を使用する場合はワーニングが出ます。RL78ではポインタ型のアドレスが16bitであるためです。以下のように__far修飾子を使用することでワーニングを解決できます。

[CC-RLコンパイラでのコード例：対処前]

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

[CC-RLコンパイラでのコード例：対処後]

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

13. ”Translator”フォルダにある以下の2つのファイルから変換結果の概要を確認可能です。
    ・checker_log_output.txt：使用するROMサイズ、RAMサイズを確認できます。注
    また、実行速度関連の情報として積和演算の回数を確認できます。
    詳細は「3.4.ROM使用量/RAM使用量の確認」を参照してください。
    ・network_description.txt：変換したニューラルネットワークの構成情報を確認できます。
    注：ニューラルネットワークが使用するROMサイズ/RAMサイズの概算値として見積もる機能です。
    本バージョンでは”Convert Option”に”RAM Size Priority”を設定した場合のROMサイズ/RAMサイズを表示します。”Convert Option”を変更しても見積もり結果は変わりません。

14. AI処理を含むプログラムをビルドします。
    以降、プログラムのデバッグ手順は通常のプログラムと同じです。
    e² studioを使用したプログラムのデバッグ手順は以下のドキュメントでご確認ください。

対象デバイス	ドキュメントタイトル	資料番号
REファミリ	統合開発環境e² studio クイックスタートガイド	R20UT5034JJ
RAファミリ	統合開発環境e² studio クイックスタートガイド	R20UT4989JJ
RZファミリ	統合開発環境e² studio 入門ガイド	R20UT4535JJ
上記以外のMCU	統合開発環境e² studio 入門ガイド	R20UT4819JJ

3.2. 複数ニューラルネットワークの変換

1つのプログラム内で複数のニューラルネットワークを動作させるには、以下のような手順で変換を行います。
ここでは、以下の2つのニューラルネットワークを1つのプログラムで動作させる手順を例として説明します。

・信号波形から異常を検出するニューラルネットワーク
・音から異常を検出するニューラルネットワーク

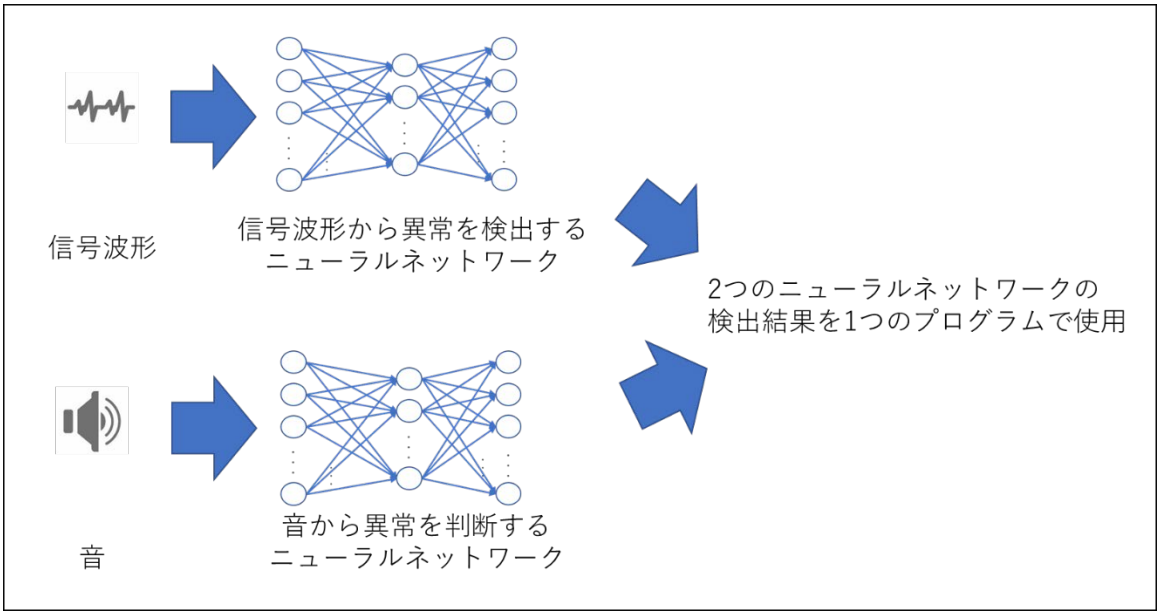

図3.12　2つのネットワークの変換例

1. 1つ目のネットワーク(異常信号検出用)を変換します。
   ・e-AIトランスレータを起動して、”Input Model File Name”に1つ目のネットワークの学習済みモデルを指定します。
   ・”Translator Output Location”は、1つ目のネットワークと2つ目のネットワークで別々のフォルダを指定します。
   ・2つ目のネットワークと区別するため、”Model Name”には1つ目のネットワークの名前を設定します。
   ここでは”Signal”と設定しています。
   なお、指定可能な文字は半角英数文字で、最大8文字です。
   ・上記の設定後、”Translate”ボタンを押して変換します。

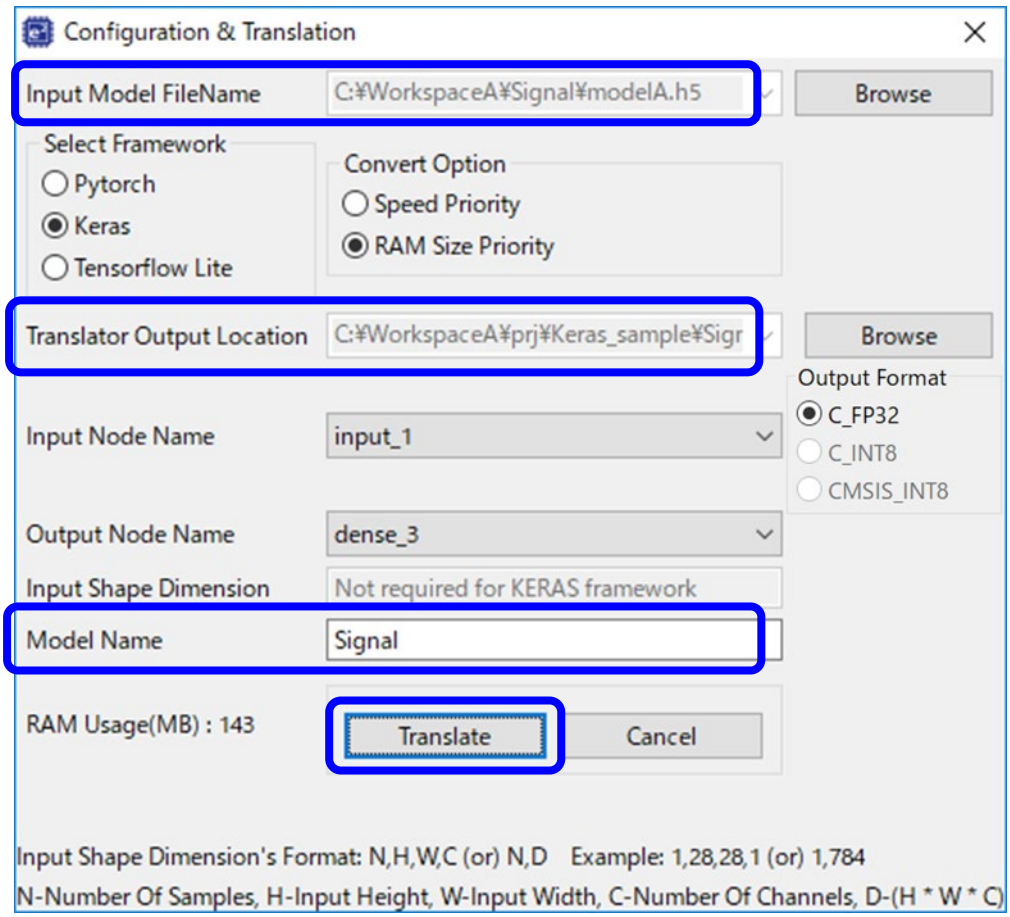

図3.13 1つ目のネットワークの変換(異常信号検出用)

2. 2つ目のネットワーク(異音検出用)を変換します。
   ・再度e-AIトランスレータを起動して、”Input Model File Name”に2つ目のネットワークの学習済みモデルが存在するフォルダを指定します。
   ・”Translator Output Location”は、1つ目のネットワークと2つ目のネットワークで別々のフォルダを指定します。
   ・1つ目のネットワークと区別するため、”Model Name”には2つ目のネットワークの名前を設定します。ここでは”Sound”と設定しています。
   なお、指定可能な文字は半角英数文字で、最大8文字です。
   ・上記の設定後、”Translate”ボタンを押して変換します。

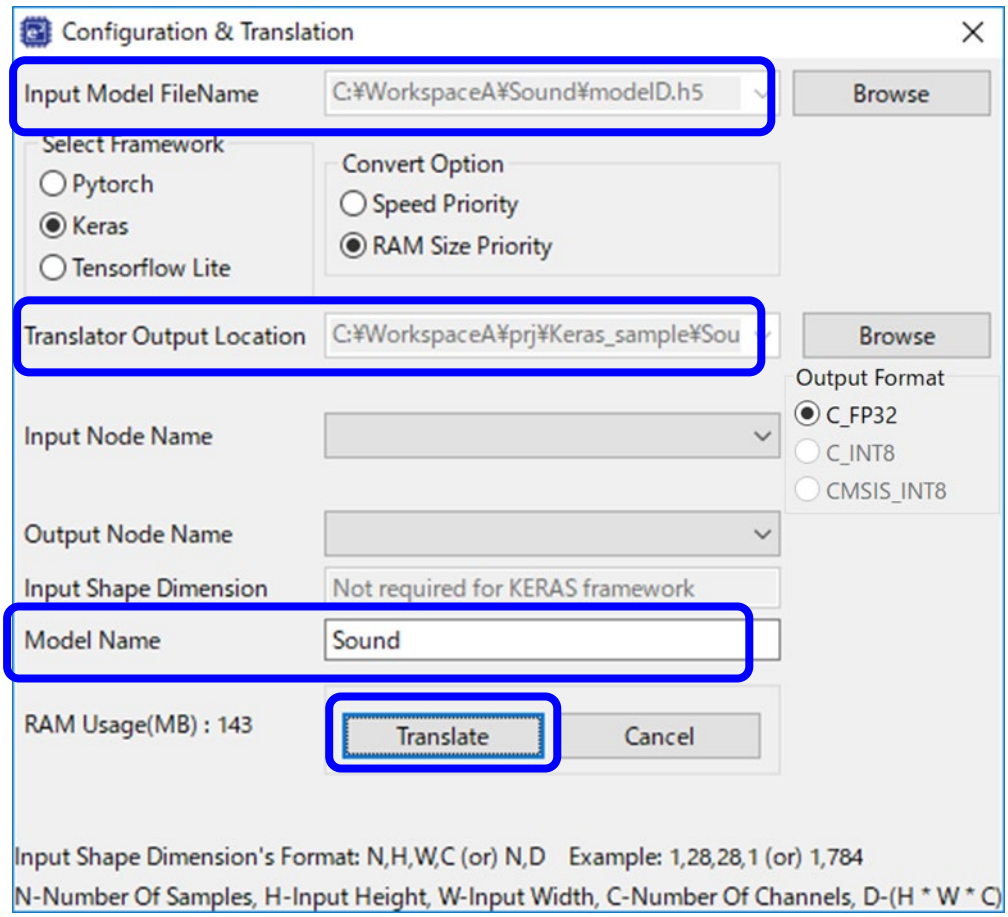

図3.14 2つ目のネットワークの変換(異音検出用)

3. 変換後のソースファイルを確認します。
   以下の通り”Model Name”で指定した名称が追加された形で、それぞれのソースファイル/ヘッダファイルが生成されます。ただし、typedef.hおよびimput_image_0.hについては同じ内容となるため、”Model Name”で指定した名称は追加されません。
   推論関数を使用するは、それぞれ以下の関数を呼び出します。
   ・異常信号検出用の推論関数 : ”dnn_compute_Signal.c”内のdnn_compute_Signal()関数
   ・異音検出用の推論関数 : ”dnn_compute_Sound.c”内のdnn_compute_Sound()関数

表3-3 変換後のソースファイルとヘッダファイル

	1つ目のネットワーク (異常信号検出用)	2つ目のネットワーク (異音検出用)
変換結果	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. “network_Signal.c”、および”network_Sound.c”をマージします。
   これらの2つのファイルはそれぞれのニューラルネットワークの各レイヤーに相当する関数や変数を定義しています。2つのネットワークが共通に使用しているレイヤーがある場合、関数や変数が2重に定義されている形となっているため、ビルドエラーとなります。
   このため、2つのファイルで重複している関数や変数の定義がある場合は、片方の定義をコメントアウトしてください。

5. プログラムをビルドします。
   以上の手順で複数のニューラルネットワークを1つのプログラムで動作させることできます。

3.3. 学習済みモデルのサンプルとメイン関数のサンプルコード

e-AIトランスレータのインストールフォルダに学習済みモデルのサンプルとメイン関数のサンプルを同梱しています。必要に応じてご使用ください。

[格納場所]
C:¥Renesas¥e2_studio¥eclipse¥plugins¥com.renesas.eaitranslator_2.3.0¥Translator
|
|--- main.c：メイン関数のサンプルコード(推論の呼び出し処理)
|--- input：入力文字のサンプル 0～9 ヘッダファイル形式
|
--- model
|
|--- Keras：Kerasを使用したAIの学習済みモデル
|--- tflite：TensorFlow Liteを使用したAIの学習済みモデル
---- PyTorch：PyTorchを使用したAIの学習済みモデル

[Kerasの学習済みモデル]
サンプルとして提供しているモデルは、MNIST(手書き文字認識処理)です。
・0～9の手書き文字をニューラルネットワークへ入力し、どの数値が入力されたかを推論します。
・入力データは、サイズ：28×28ピクセル、グレースケールのデータです。
・推論の結果として、入力された文字がどの数字であったかを確率で出力します。
学習済みモデルをe-AIトランスレータで変換し、上記のmain.cや入力文字のヘッダファイルと組み合わせることでMCU/MPU上での推論動作の確認が可能です。

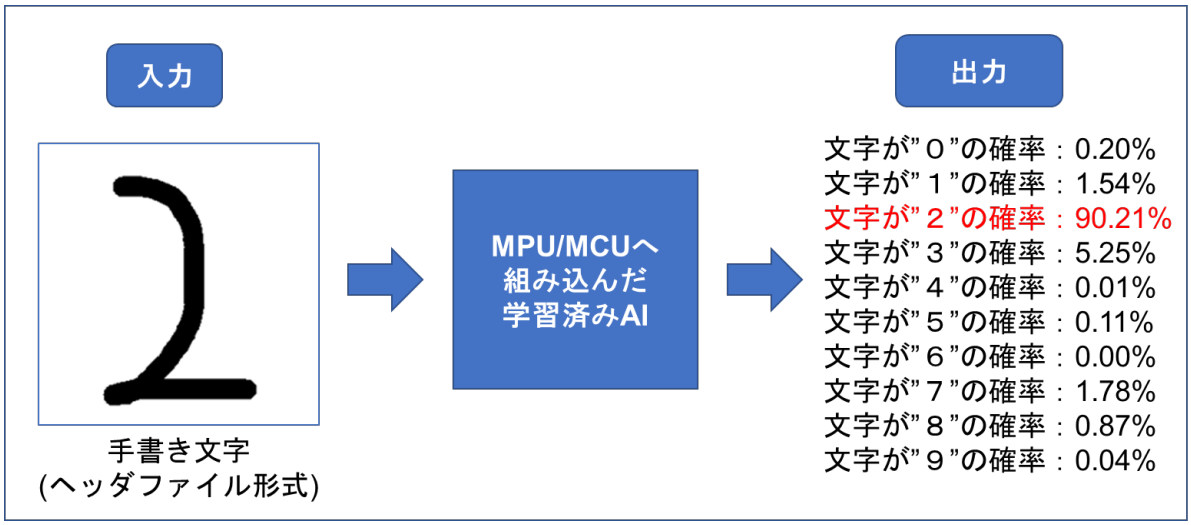

図3.15 Keras用サンプルの動作イメージ

[TensorFlowLite用の学習済みモデル]
サンプルとしていくつかのモデルを同梱しています。
上記のmain.cや入力文字のヘッダファイルと組み合わせて使用することはできませんが、学習済みモデルをe-AIトランスレータで変換する動作の確認が可能です。

[PyTorch用サンプル]
サンプルとして提供しているモデルは、花の画像分類です。
・チューリップ、ヒマワリ等の画像をニューラルネットワークへ入力し、どの花の画像が入力されたかを推論します。
・入力データは、チャンネル数：3(RGB)、サイズ：100×100ピクセルのデータです。
(e-AIトランスレータの”Input Shape Dimention”は”3,100,100”に設定します。)
上記のmain.cや入力文字のヘッダファイルと組み合わせて使用することはできませんが、学習済みモデルをe-AIトランスレータで変換する動作の確認が可能です。

3.4. ROM使用量/RAM使用量の確認

推論処理のROM使用量/RAM使用量の概算値は”checker_log_output.txt”で確認できます。
以下の要領で確認します。

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　　：推論処理のROM使用量
②(上段)：各レイヤーで使用されるRAMサイズの合計
　　(e-AIトランスレータ V1.4.0以前での推論処理のRAM使用量)
②(下段)：実際に推論処理で使用されるRAMのサイズ
③　　：推論処理に含まれる積和演算の回数
　　回数が多いほど推論処理が遅くなる傾向があります。

3.5. PyTorchのモデルファイル作成と保存

PyTorchで学習済みモデルを作成する場合は以下の3点に注意して作成してください。
以下の3点を守っていない場合、変換がエラーとなります。

1.Pythonファイルの構成
モデル定義用のPythonファイルと学習用のPythonファイルを別ファイルとして作成してください。

2.モデルファイルの保存方法
モデルファイルを保存する際、”state_dict”を指定しないでください。

3.e-AIトランスレータでの変換
モデル定義用Pythonファイル(*.py)と保存したモデルファイル(*.pth)を同じフォルダに格納します。
e-AIトランスレータの”Input Model File Name”ではモデルファイル(*.pth)を指定します。

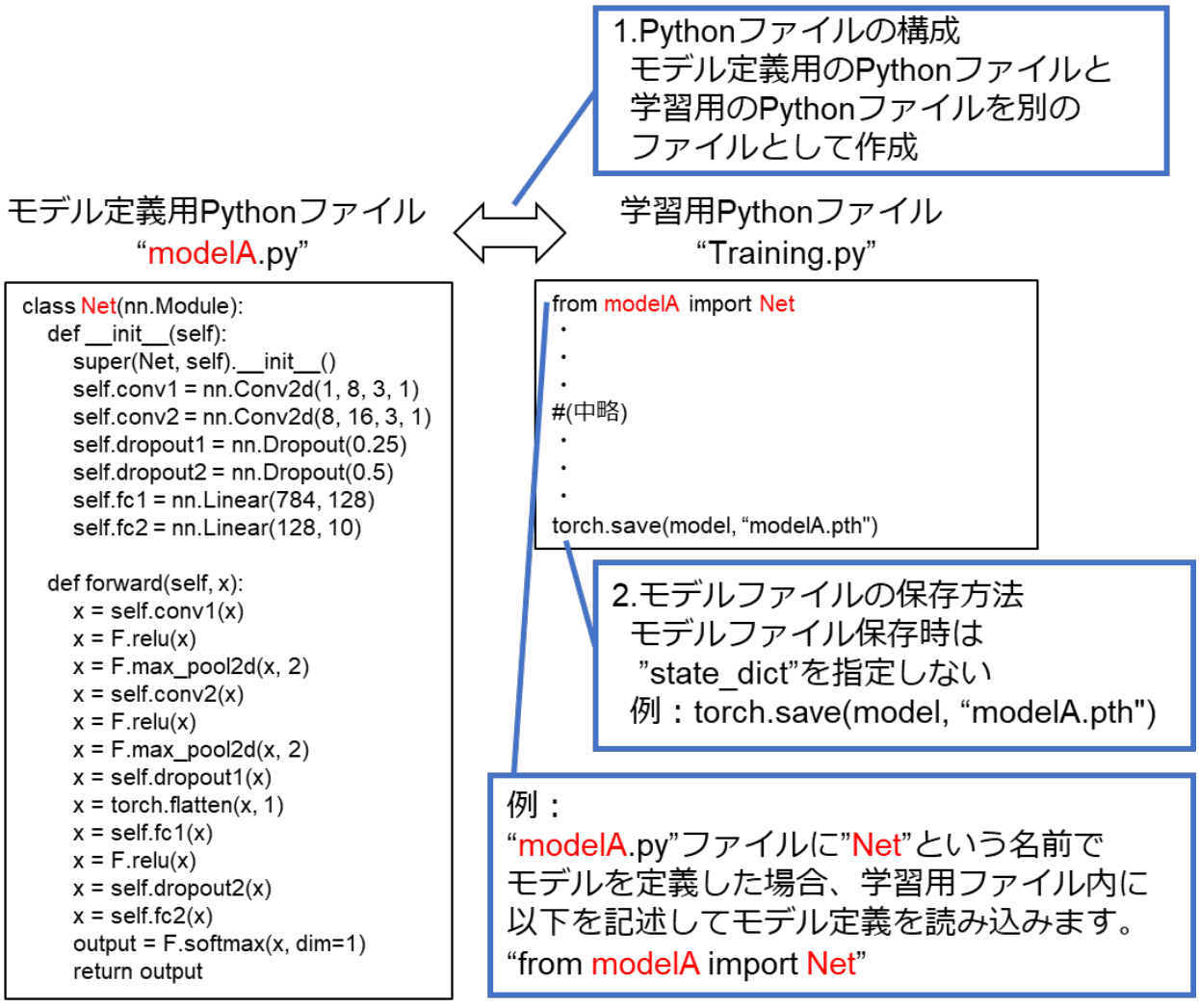

図3.16 PyTorchのモデルファイル作成と保存

3.6. TensorFlow Liteのモデルファイル作成と保存

KerasもしくはTensorFlow(tf.keras)の学習済みモデルはTensorFlow Liteを使用して8bit量子化できます。以下の図に示すように8bit量子化することで、重みデータやレイヤー間の中間データを32bit float形式から8bit signed int形式に圧縮することができます。ROMサイズやRAMサイズを削減することができるため、AI処理の組込みに大変有効な技術です。

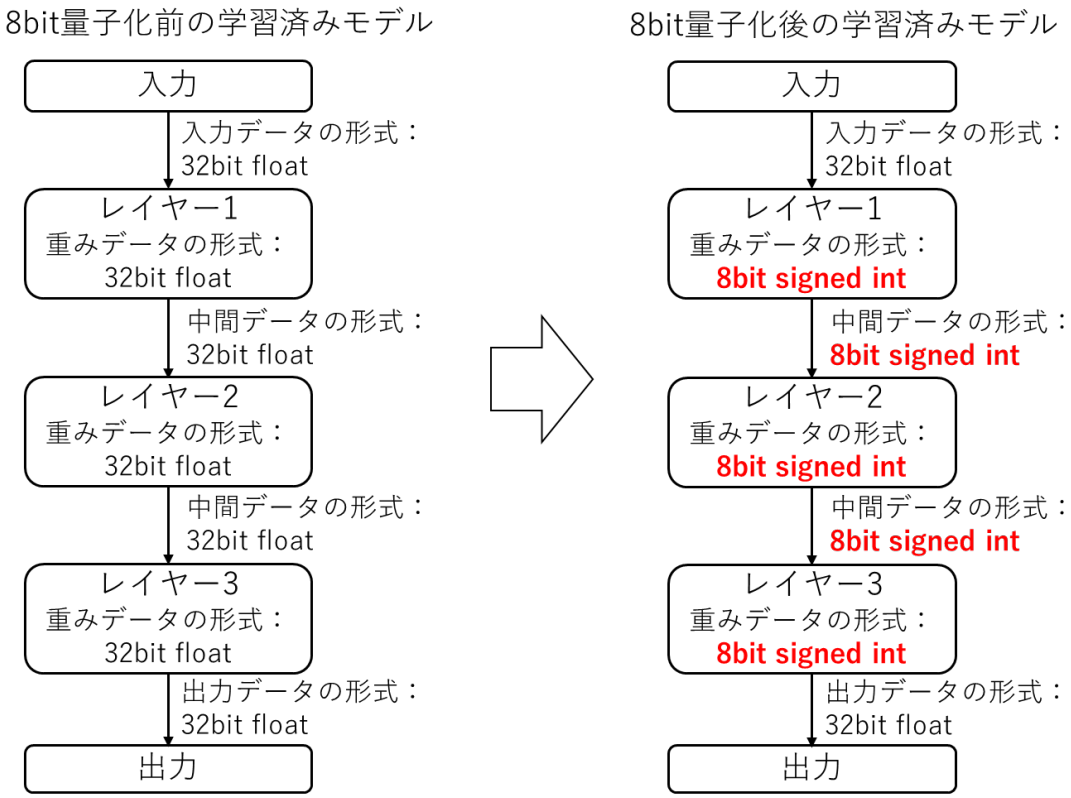

図3.17　TensorFlow Liteの8bit量子化によるモデルの圧縮例

・e-AIトランスレータが対応しているTensorFlow Liteモデル
e-AIトランスレータは以下の形式で8bit量子化されたTensorFlow Liteモデルに対応しています。
学習時の入出力データの形式と8bit量子化時の入出力データの形式が合うように量子化を実施してください。

表3-4　e-AIトランスレータが対応しているTensorFlow Liteモデル

量子化方式	入力データの形式	出力データの形式	重みデータの形式	中間データの形式 (Activationの形式)
重みデータと中間データを量子化[注]	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

注：重みデータのみを量子化したTensorFlow Liteモデルには対応していません。

・TensorFlow Liteモデルの作成方法
TensorFlow Liteモデルは以下の手順で作成します。
1.KerasもしくはTensorFlow(tf.keras)で学習済みモデル(拡張子が*.h5のファイル)を作成
2.TFLiteConverterを使用して学習済みモデルを8bit量子化
3.量子化後のモデルを拡張子が*.tfliteのファイルとして保存

次ページに8bit量子化するためのスクリプト例を示します。8bit量子化する際の参考としてください。
[スクリプト例の説明]
・学習時のデータセット：MNIST
・学習済みモデルファイル名：cnn_model.h5
・出力ファイル1：8bit_cnn_model_fp32io.tflite
(重みデータと中間データを量子化、入出力形式：32bit float)
・出力ファイル2：8bit_cnn_model_uint8io.tflite
(重みデータと中間データを量子化、入出力形式：8bit unsigned int)
※スクリプト中の”tf.uint8”を”tf.int8”に変更することで8bit signed int入出力モデルも出力できます。

なお、TensorFlow Liteによる8bit量子化の詳細については、以下のWebページも合わせてご確認ください。
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)
```

図3.18　TensorFlow Liteで8bit量子化するためのスクリプト例

3.7. CMSIS_INT8の使用方法

e-AIトランスレータの”output format”で”CMSIS_INT8”を選択した場合、生成されたソースコードとCMSIS NN、CMSIS DSPライブラリを組み合わせることで推論速度を高速化することができます。
e-AIトランスレータで”CMSIS_INT8”を選択してソースファイルを生成後、以下の手順を実施してプロジェクトにCMSIS NN、CMSIS DSPライブラリを追加してください。

・CMSIS_INT8ライブラリの対応デバイス
RAファミリ、RXファミリ

・RAファミリでの使用方法

1. 統合環境e² studioでプロジェクトを作成後、プロジェクトエクスプローラーにある”configuration.xml”をダブルクリックします。その後、”FSP Configuration”ビューの”Stacks”タブを開きます。

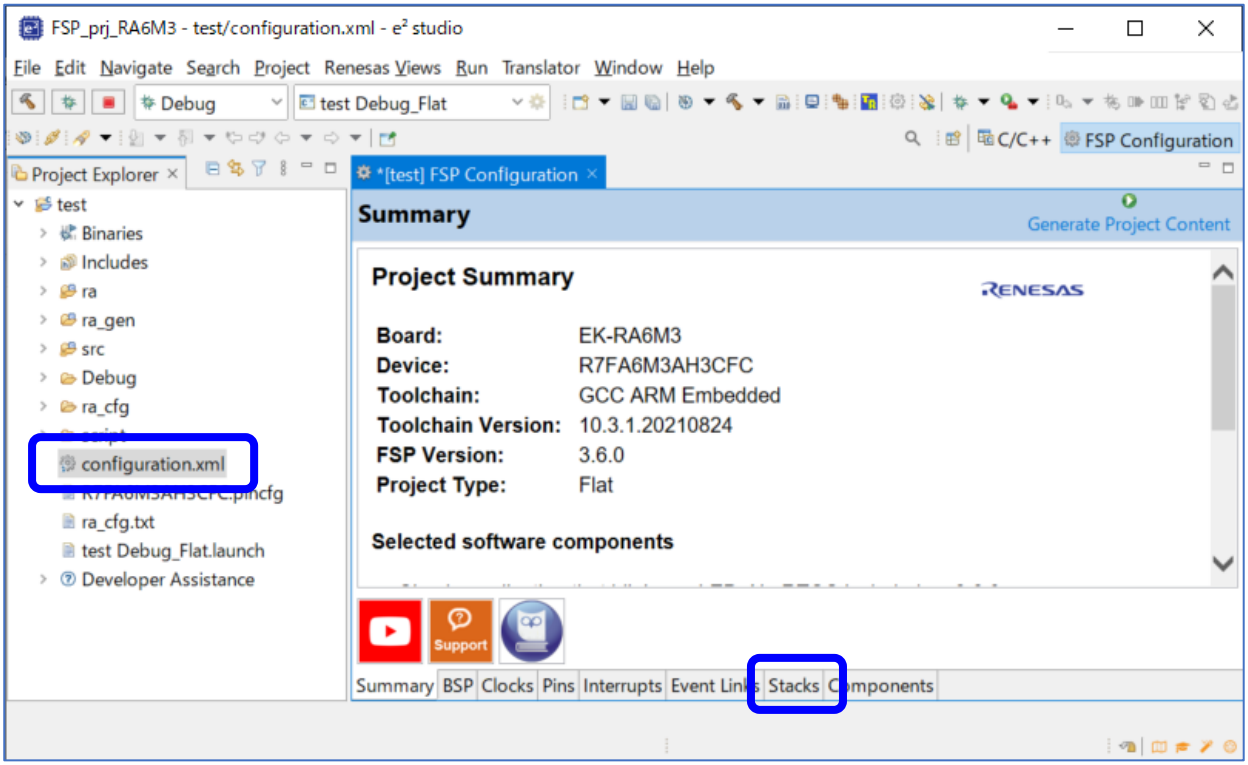

図3.19 ”FSP Configuration”ビューの”Stacks”タブ

2.”Stacks”タブの上側にある”New Stack”をクリックし、”Artificial Intelligence”にある”Arm CMSIS5 NN Library Source”を選択します。

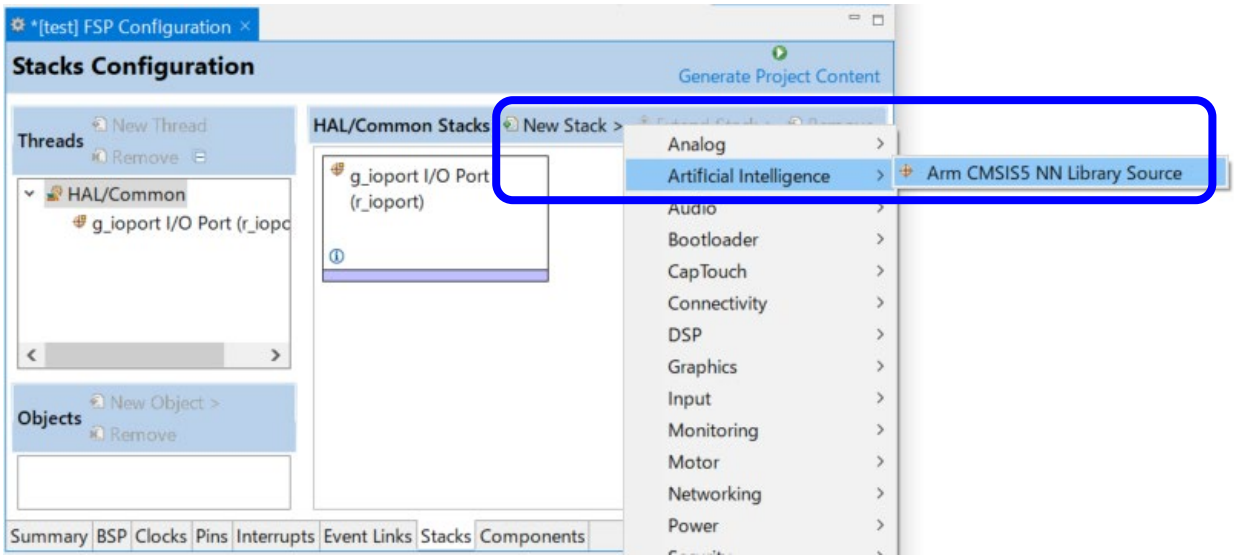

図3.20　CMSISライブラリの追加

3. ”FSP Configuration”ビューに”Arm CMSIS NN Library Source”と”Arm CMSIS DSP Library Source”が追加されたことを確認して”Generate Project Content”ボタンを押します。

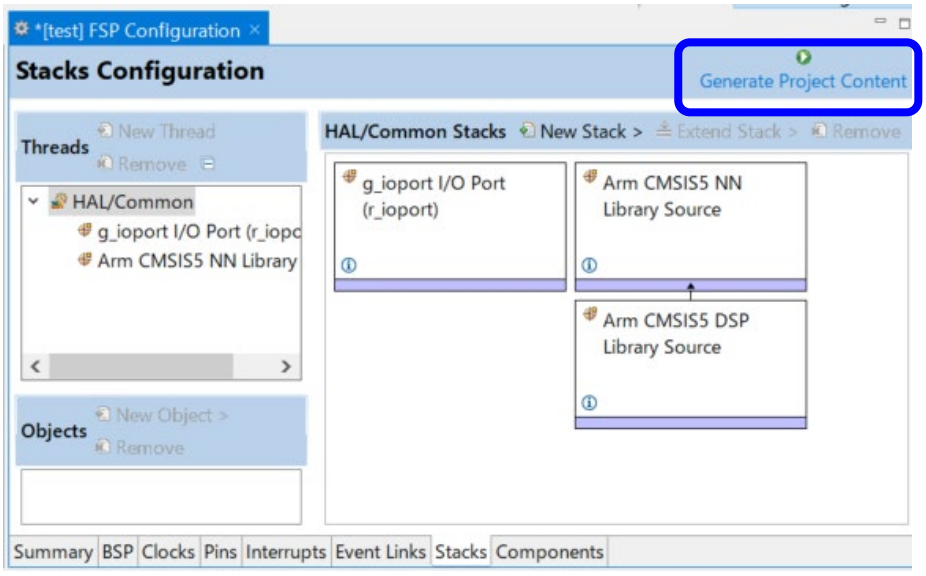

図3.21　CMSISライブラリの生成

・RXファミリでの使用方法

別途CMSIS for RXライブラリの入手が必要です。

使用方法については、CMSIS for RXライブラリの添付ドキュメントでご確認ください。

・”Convert option”、"Output format"オプションに関する補足
”Convert option”と"Output format"はどちらも実行速度の高速化や ROM/RAM 使用量の削減に関するオプションです。各オプション使用時の動作については以下の表を参照してください。

表 3-5 ”Convert option”、"Output format"オプションに関する補足

オプション名		設定の内容と動作
Convert option		重みデータの配置場所を切り替えるオプション
	RAM Size Priority(推奨注)	重みデータを ROM に配置することで、変換後のプログラムの RAM 使用量を優先します。
	Speed Priority	重みデータを RAM に配置することで、変換後のプログラムの実行速度を優先します。
Output Format		推論プログラムを切り替えるためのオプション
	C_INT8	全ての MPU/MCU で使用可能な推論プログラムを生成します。
	CMSIS_INT8	CMSIS ライブラリを活用することで推論速度を高速化した推論プログラムを生成します。RA、RX ファミリのみ対応しています。

注：一般的に重みデータのサイズは大きいため、”RAM size priority”の使用を推奨します。

4. 注意事項

・未対応のニューラルネットワークは、正しく変換できません。
対応済みのニューラルネットワークは、「1.2 変換可能なニューラルネットワークの種類」および「1.3 サポートAPI一覧」でご確認ください。

・ニューラルネットワークの入力層にReshape層などの前処理がある場合、変換した推論関数から前処理は取り除かれます。このため前処理後の形状で推論関数へデータを入力してください。
また、”Output format”に”CMSIS_INT8”を指定し、かつ推論関数の入力が3D shapeとなっている場合は、前処理が取り除かれず変換エラーとなる場合があります。このような場合は前処理を取り除いてから保存したモデルを使用してください。

・推論関数の入力層が4D shapeとなる場合、[N, C, H, W]の形式でデータを入力してください。
また、推論関数の出力データが4D shapeとなる場合、 [N, C, H, W]の形式となります。
なお、PyTorchでの学習時は[N, C, H, W]の形式、KerasおよびTensorFlow(tf.keras)での学習時は[N, H, W, C]の形式でデータを入力するのが標準の仕様となっています。
※N：Number of Samples、H：Input Height、W：Input Width、C：Number of Channels

・入力データの前処理後、ニューラルネットワーク内で最初にデータを処理するレイヤーとして使用可能なものは畳み込み層、もしくは全結合層です。
活性化関数、クリップ関数、正規化層、プーリング層は使用できません。使用した場合、eAI-311エラーが発生します。

・ニューラルネットワークの学習に使用するPCとe-AIトランスレータを使用するPCが異なる場合、両方のPCにインストールするディープラーニングフレームワークのバージョンが同じとなるようにしてください。
また、KerasやTensorFlow(tf.keras)でLambdaレイヤー(keras.layers.Lambdaもしくは tf.keras.layers.Lambda)を使用する場合は、両方のPCにインストールするPythonのバージョンも同じとなるようにしてください。バージョンの不整合を検出した場合、eAI-310エラーが発生します。

・Reshape処理等データの構造を変更できるのは、4D shapeから2D shape、2D shapeから4D shapeのみです。
4D shapeから4D shape、2D shapeから2D shapeのReshape処理はサポートしていません。また、3D shapeを含むReshape処理についてもサポートしていません。このようなReshape処理は使用しないでください。

・PyTorchで学習済みモデルを作成する場合は以下の3点に注意して作成してください。
以下の3点を守っていない場合、変換がエラーとなります。
1.Pythonファイルの構成
モデル定義用のPythonファイルと学習用のPythonファイルを別ファイルとして作成します。
2.モデルファイルの保存方法
モデルファイルを保存する際、”state_dict”を指定しないでください。
3.e-AIトランスレータでの変換
モデル定義用Pythonファイル(*.py)と保存したモデルファイル(*.pth)を同じフォルダに格納します。
e-AIトランスレータの”Input Model File Name”ではモデルファイル(*.pth)を指定します。
詳細は「3.5. PyTorchのモデルファイル作成と保存」をご確認ください。

・ルネサス製コンパイラのCC-RX、CC-RLを使用する場合、以下のようなコンパイラ設定の変更が必要となる場合があります。
[標準ライブラリ設定の追加]
コンパイラのデフォルト設定で標準ライブラリ”math.h”が有効になっていないため、ビルドエラーとなることがあります。このような場合はコンパイラの標準ライブラリ設定で”math.h”を有効にしてください。
[C99オプションへの設定変更]
コンパイラのデフォルト設定がC89やC90となっているため、ビルドエラーとなることがあります。このような場合はC99へ設定を変更してください。
[ヒープメモリの確保]
ニューラルネットワーク内に以下の処理が含まれる場合、e-AIトランスレータはmallocを含むCソースファイルを生成するため、ヒープメモリを確保する必要があります。
Max pooling(padding付きの場合のみ)、Sigmoid、Softsign、Softplus、TanH

・TensorFlow Liteの8bit量子化モデルを使用する場合、以下の注意事項があります。
- Batch normalization レイヤーは Convolution - Batch Normalization - ReLU の順序で Bacth Normalizationを使用してください。その他の順序で使用した場合、推論誤差が発生することがあるため、組み込み後に推論精度を十分に確認してください。
- 入力データ用の変数をニューラルネットワーク内の演算で使用するため、入力データ用の変数はconst定義しないでください。

・"Input Model File Name"の"Browse"ボタンを押して入力モデルファイルを指定した後で再度"Browse"ボタンを押すと、指定したモデルファイルが存在するフォルダとは異なるフォルダが開く場合があります。(開くフォルダが異なっているだけで、実際には正しい入力モデルファイルが指定されています。)

・”output format”に”CMSIS_INT8”もしくは"C_INT8"を指定し、かつニューラルネットワーク内にPadding処理付きのPoolingレイヤーを含む場合、PC上の推論結果とマイコン上の推論結果に若干の誤差が生じることがあります。
誤差が生じた際は組み込み後に推論精度を十分に確認してください。

・”output format”に”CMSIS_INT8”を指定し、かつ"convert option"に"RAM Size Priority"を指定する場合、ビルド時に次のワーニングが発生します。
“Initialization discards 'const' qualifier from pointer target type.”
これはconst定義が無い構造体(CMSIS-NNライブラリ内)に対して、const定義された構造体(e-AIトランスレータが出力)を代入しているために発生するワーニングですが、実際の動作としては問題ありません。

・PyTorchのSoftmax関数は対応済みですが、LogSoftmax関数には未対応です。
PyTorchのモデルを使用する場合、このような未対応関数を含むニューラルネットワークを変換してもエラーが発生しない場合があります。
このため、入力したモデルと出力ファイル”dnn_compute.c”を比較して問題なく変換できているかを確認してください。

5. エラーメッセージ

本章ではe-AIトランスレータが出力するエラーメッセージと対処方法について説明します。
e-AIトランスレータでエラーが発生すると、次のようなダイアログを表示します。
ダイアログ内に表示されているフォルダにあるログファイル内で、”eAI-xxx”の形式でエラー番号が確認できます。エラー発生時は、このエラー番号から対処方法をご確認ください。

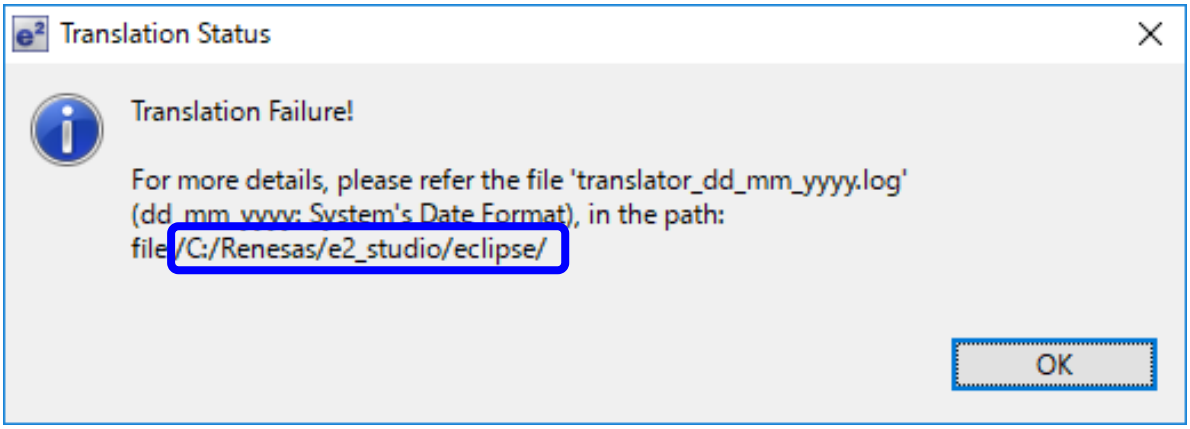

図5.1　エラーダイアログ内のログファイル出力フォルダ位置

```
[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..
```

図5.2　ログファイル内のエラー番号表示位置

表 5-1 エラー番号と対処方法(1/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
セットアップ関連	eAI-101	Python not installed. Python 3.7.7がインストールされていない場合のエラーです。 「2.3. Python3.7.7のインストール」をご確認の上、インストールしてください。
	eAI-102	64bits Python version is not installed. インストールされているPythonが64bit版ではなく、32bit版の場合に発生するエラーです。64bit版をインストールしてください。また、お使いのPCが64bit版Windowsであることもご確認ください。
	eAI-103	”tensorflow” package is not installed. tensorflowパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-104	”progressbar” package is not installed. progressbarパッケージがインストールされてない場合のエラーです。 「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-105	”prettytable” package is not installed. prettytableパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-106	”pytorch” package is not installed. pytorchパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。

表 5-2 エラー番号と対処方法(2/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
セットアップ関連	eAI-107	“h5py” package is not installed. h5pyパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-108	“pycrypto” package is not installed. pycryptodomeパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-110	“configparser” package is not installed. configparserパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-111	“psutil” package is not installed. psutilパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。
	eAI-112	"Keras" package is not installed. Kerasパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。 ただし、TensorFlow-backend版のKerasを使用している場合はTensorFlowのインストールが正しく行われているかをご確認ください。
	eAI-113	"onnx" package is not installed. onnxパッケージがインストールされていない場合のエラーです。「2.4. Python Packageのインストール」をご確認の上、インストールしてください。

表 5-3 エラー番号と対処方法(3/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
設定値 関連	eAI-208	Invalid Input Shape Dimensions “Input Shape Dimensions”に使用できない文字がある場合のエラーです。入力可能な文字は数値とカンマ(,)のみです。なお、数値およびカンマは半角文字で入力してください。
	eAI-209	Input Shape Dimensions Not Found “Input Shape Dimensions”が設定されていない場合のエラーです。“Input Shape Dimensions”に、ニューラルネットワークの入力層のサイズを設定してください。
	eAI-210	Incorrect Input Shape Dimensions “Input Shape Dimensions”に指定した値が正しくない場合のエラーです。"C,H,W "、もしくは"D"の順番で入力してください。 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. “Model Name”エリアに関するエラーです。モデル名が以下2点を守れているかご確認ください。 ・半角英数文字のみで指定する ・8文字以内で指定する
	eAI-215	Input model filename is missing. “Input Model File Name”に指定されたファイルが見つからない場合のエラーです。正しいファイルを指定しているかご確認ください。
	eAI-217	Unsupported library provided with Incorrect input_model_filename. “Input Model File Name”と”Output Format”の設定に矛盾がある場合のエラーです。“Input Model File Name”に正しいモデルファイルを指定しているか、”Output Format”の設定が正しいかをご確認ください。

表 5-4 エラー番号と対処方法(4/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
フレームワーク関連	eAI-301	Framework Dependent File Not Found “Input Model File Name”で指定したファイルに関するエラーです。指定したファイルが正しいかどうか、ご確認ください。 「3.1. 基本的な使用手順」の手順5をご参照の上、フォルダ内に必要なファイルが全てそろっているかご確認ください。 PyTorchをご使用の場合は、「3.5. PyTorchのモデルファイル作成と保存」もご確認ください。
	eAI-302	pytorch Network Configuration Mismatch PyTorchの学習済みモデルとして指定されたファイルの拡張子が”*.pth”と”*.py”ではない場合のエラーです。 指定したファイルの拡張子が”*.pth”と”*.py”になっているかご確認ください。
	eAI-303	Memory Error in Tensorflow PCのメモリ不足により発生するエラーです。エラーとなった場合、PC上のアプリケーションを全て終了してから再度e-AIトランスレータを使用してください。
	eAI-304	Error in importing frozen graph 学習済みモデルから変換に必要なパラメータを正しく取得できなかった場合のエラーです。お手数ですが、本エラーが発生した場合はルネサスまでお問い合わせください。
	eAI-305	Unsupported network 未サポートのニューラルネットワークを使用している場合のエラーです。 「1.2. 変換可能なニューラルネットワークの種類」および「1.3. サポートAPI一覧/未サポートAPI一覧」をご参照の上、未サポートのニューラルネットワークやAPIを使用していないかご確認ください。

表 5-5 エラー番号と対処方法(5/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
フレームワーク関連	eAI-306	Unsupported Algorithm/Model 未サポートのモデルを指定した場合のエラーです。 以下の例のように学習済みモデル内に未サポートのアルゴリズムが含まれていないかご確認ください。 - ニューラルネットワークではないアルゴリズム - 出力レイヤーが複数あるニューラルネットワーク
	eAI-307	Tensorflow Version mismatch サポートしていないTensorFlowバージョンで学習済みモデルが作成されている場合のエラーです。 学習済みモデルがTensorFlow 2.4.0で作成されているかご確認ください。
	eAI-308	Keras model file versions mismatch サポートしていないKerasバージョンで学習済みモデルが作成されている場合のエラーです。以下のバージョンで作成されているかご確認ください。 ・単体版Keras：2.4.3 ・TensorFlow-backend版Keras：2.4.0-tf (TensorFlow 2.4.0に同梱)
	eAI-309	Multiple Bias Error 学習済みモデル内の演算で、バイアスを2重に加算している場合のエラーです。学習用のスクリプト内でバイアスを2重に加算していないかご確認ください。 例) “tf.keras.layers.conv2d”APIの出力にバイアスを加算している (“tf.keras.layers.conv2d”はAPI内でバイアスの加算処理があるため、APIの出力値へバイアスを加算する必要はありません。)
	eAI-310	Unable to load keras/tf.keras model file Lambdaレイヤーを使用しているKerasモデルの読み込みに失敗した場合のエラーです。 モデルを学習したPCとe-AIトランスレータを使用しているPCでKerasやPythonのバージョンを同じにしてください。 詳細は「4.注意事項」を参照してください。
	eAI-311	Input restricted layer ニューラルネットワーク内で最初にデータを処理するレイヤーが活性化関数、クリップ関数、正規化層、プーリング層となっている場合のエラーです。 詳細は「4.注意事項」を参照してください。

表 5-6 エラー番号と対処方法(6/6)

エラーカテゴリ	エラー番号	エラーメッセージと対処方法
ファイル アクセス関連	eAI-401	File Open Error “Input Model File Name”に指定したフォルダ内のファイルが開けなかった場合のエラーです。フォルダがアクセス禁止になっていないかどうか、他のアプリケーションでファイルを使用中で無いかどうかご確認ください。
	eAI-402	File Creation Error ”Translator Output Location”に指定したフォルダへファイルを生成できなかった場合のエラーです。フォルダが書き込み禁止設定になっていないかどうかご確認ください。
	eAI-403	File Overwrite Error ”Translator Output Location”に指定したフォルダ内のファイルを上書きできなかった場合のエラーです。フォルダが書き込み禁止設定になっていないかどうか、他のアプリケーションでファイルを使用中で無いかどうかご確認ください。
	eAI-404	File Not Found “Input Model File Name”に指定したフォルダ内にファイルが無かった場合のエラーです。指定したフォルダが正しいかどうかご確認ください。
その他	eAI-501	Uncaught Exception 上記のいずれにも分類できないエラーです。 PyTorchモデルの指定時は、「3.5. PyTorchのモデルファイル作成と保存」でモデルの保存方法が正しいかどうかをご確認ください。 その他の原因と考えられる場合は、ルネサスまでお問い合わせください。

改訂記録

表 6-1　改訂履歴(1/3)

Rev.	発行日	改訂記録	
		ページ	ポイント
1.00	2017.6.30	ー	初版発行
1.01	2017.9.8	P5	誤記訂正
		P8	Pythonインストール時のPATH設定説明を変更
		P16	誤記訂正
		P19	誤記訂正
2.00	2017.11.6	全般	e-AIトランスレータのバージョンを変更(V1.0.0→V1.0.1) バージョンアップに伴うGUIイメージの変更やインストール先フォルダ名の変更を反映
		P5	インストールが必要なソフトウェアに「Microsoft Visual Studio 2015 Visual C++ 再頒布可能パッケージ」を追加
		P7, P8	「1.3. サポートAPI一覧/未サポートAPI一覧」を追加
		P9	「1.4. e-AIトランスレータV1.0.0からV1.0.1の変更点」を追加
		P12, P13	Pythonインストール後の確認方法を追加
		P15, P16, P21	Ubuntu環境にインストールするPythonのバージョンを修正 (修正前：2.7→修正後：2.7.6)
		P19	"Input Model Location"のフォルダ名/パス名に使用可能な文字の種類を追記
		P20, P21	Ubuntu環境からWindows環境へCaffeの学習済みモデルを変換する場合の手順を変更
		P23	"Translator Output Location"のフォルダ名/パス名に使用可能な文字の種類を追記
		P24	"Input Shape Dimensions"に入力可能な文字の種類を追記 "Input Shape Dimensions"の入力形式を拡張した旨を追記
		P27	注意事項を1点追記
		P28～P31	「5. エラーメッセージ」を追加
2.01	2018.5.15	P5, P14	インストールするTensorFlowのバージョンに関する記述を追加
		P20	Caffe使用時の".prototxt"ファイルに関する記述を追加
3.00	2018.7.24	全般	e-AIトランスレータのバージョンを変更(V1.0.1→V1.0.2) バージョンアップに伴うGUIイメージの変更やインストール先フォルダ名の変更を反映
		P5, P9, P14	TensorFlow バージョン1.8.0の対応に伴い修正
		P6, P7, P8	LRN (Local Response Normalization)の対応に伴い修正
		P9	e-AIトランスレータ V1.0.2での変更点を追加
		P18	"RAM Usage(MB)"の説明を追加
		P21, P22	Ubuntu環境からWindows環境へCaffeの学習済みモデルを変換するPythonスクリプトファイルの仕様変更に伴い修正

表 6-2　改訂履歴(2/3)

Rev.	発行日	改訂記録	
		ページ	ポイント
4.00	2020.2.3	全般	e-AIトランスレータのバージョンを変更(V1.0.2→V1.4.0) バージョンアップに伴うGUIイメージの変更やインストール先フォルダ名の変更を反映
		P5, P7, P8, P21	Kerasフレームワークのサポートに関する内容を追加
		P6, P8	Batch Normalization層のサポートに関する内容を追加
		P14	インストールが必要なソフトウェアを追加、またソフトウェアのバージョン情報を更新。
		P9	e-AIトランスレータ V1.4.0での変更点を追加
		P15-P16	Ubuntu14.04のサポート終了に伴い、Caffeのセットアップに関する説明を修正
		P31-P34	複数のニューラルネットワークを変換するための手順を追加
		P36-P37	注意事項を追加、削除
		P38-P44	エラー番号とメッセージを追加
4.01	2020.3.10	P5, P14	誤記訂正
6.00	2020.10.27	全般	e-AIトランスレータのバージョンを変更(V1.4.0→V1.6.0) バージョンアップに伴いインストール先フォルダ名の変更を反映
		P9	分岐構造のあるネットワークのサポートに関する記述を追加
		P9, P36	RAM使用量削減機能に関する記述を追加
		P5, P9	e^2 studio 64bit版サポートに関する記述を追加
		P6-P9	追加サポートしたレイヤーに関する記述を追加
		P37, P38	注意事項を追加、修正
		P44	エラーメッセージを追加、修正
7.00	2021.2.26	全般	e-AIトランスレータのバージョンを変更(V1.6.0→V2.0.0) バージョンアップに伴いインストール先フォルダ名の変更を反映
		P5-P9	PyTorchに関する記述を追加。 Keras、TensorFlow(tf.keras)のサポートバージョンを更新。 Caffe、TensorFlow(tf.nnやtf.layers)に関する記載を削除
		P10-P15	インストール方法を更新
		P16-P33	使用方法を更新
		P34-P35	注意事項を追加、削除
		P37-P42	エラーメッセージを追加、削除
7.01	2021.3.26	P14	onnx、h5py、progressbar33、prettytable、configparser、psutilをインストールする際のバージョンを追記
8.00	2021.9.21	全般	e-AIトランスレータのバージョンを変更(V2.0.0→V2.1.0) バージョンアップに伴いインストール先フォルダ名の変更を反映
		P5-P9	TensorFlow Liteの8bit量子化モデルに関する記述を追加。 Keras、TensorFlow(tf.keras)のサポートバージョンを更新。
		P14, P15	インストール方法を更新
		P17-P37	使用方法を更新。 「3.6 TensorFlow Liteのモデルファイル作成と保存」を追加
		P41, P44	エラーメッセージを修正、追加
8.01	2021.10.8	P15	TensorFlow Liteのバージョンを修正

表 6-3　改訂履歴(3/3)

Rev.	発行日	改訂記録	
		ページ	ポイント
9.00	2022.3.18	全般	e-AIトランスレータのバージョンを変更(V2.1.0→V2.2.0) バージョンアップに伴いインストール先フォルダ名の変更を反映
		P5	e^2 studioのサポートバージョンを変更
		P7	サポートAPIを追加(torch.nn.LocalResponseNorm)
		P9, P19, P24, P41, P42, P43	「CMSIS_INT8」ソースファイル出力形式に関する説明を追加
		P9, P39	対応済みTensorFlowLite形式のモデルに関する記述を追加
		P44, P45	注意事項を更新/追加
10.00	2022.9.22	全般	e-AIトランスレータのバージョンを変更(V2.2.0→V2.3.0) バージョンアップに伴いインストール先フォルダ名の変更を反映
		P5	e^2 studio、FSPのサポートバージョンを変更
		P9, P19, P24, P43	機能改善に伴い「CMSIS_INT8」と「C_INT8」に関する記述を見直し
		P45, P46	注意事項を更新/追加
		P52	エラーメッセージの説明を見直し
10.01	2022.12.9	P44	注意事項の誤記を修正

e-AIトランスレータ V2.3.0
ユーザーズマニュアル

発行年月日　2017年6月30日　Rev.1.00
2022年12月9日　Rev.10.01

発行　ルネサス エレクトロニクス株式会社
〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

e-AIトランスレータ V2.3.0

ルネサス エレクトロニクス株式会社

R20UT4135JJ1001