

Изменено	10/03/2023 11:53:33	Создано	10/03/2023 11:53:33	<a href="https://www.electronshik.ru/news/show/13356?from=terraelectronica">https://www.electronshik.ru/news/show/13356? from=terraelectronica</a>
----------	------------------------	---------	------------------------	--

---

## Знакомство с пакетом расширения X-CUBE-AI для реализации искусственного интеллекта. Часть 2

---

### Генерация, сборка и программирование

#### Генерация IDE-проекта

Ниже приведена стандартная для среды STM32CubeMX последовательность действий при создании нового IDE-проекта (рис. 34):

Перейдите на вкладку «Project Manager».

Задайте местоположение проекта и его имя.

Выберите требуемый набор инструментальных средств и среду разработки (EWARM для IAR™, TrueSTUDIO® для Atollic IDE или другой).

Увеличьте минимальный размер кучи/стека, предлагаемый по умолчанию, чтобы обезопасить себя от возможного переполнения при первом запуске программы (тестовому приложению «AI Validation» требуется куча размером не менее 2 кбайт).

Project Settings

Project Name

my\_ai\_project

Project Location

C:\ai\_lab\project\

Browse

Application Structure

Basic

☐ Do not generate the main()

Toolchain Folder Location

C:\ai\_lab\project\my\_ai\_project\

Toolchain / IDE

EWARM V8

☐ Generate Under Root

Linker Settings

Minimum Heap Size

0x2000

Minimum Stack Size

0x4000

Mcu and Firmware Package

Mcu Reference

STM32F746ZGTx

Firmware Package Name and Version

STM32Cube FW\_F7 V1.14.0

☒ Use latest available version

☒ Use Default Firmware Location

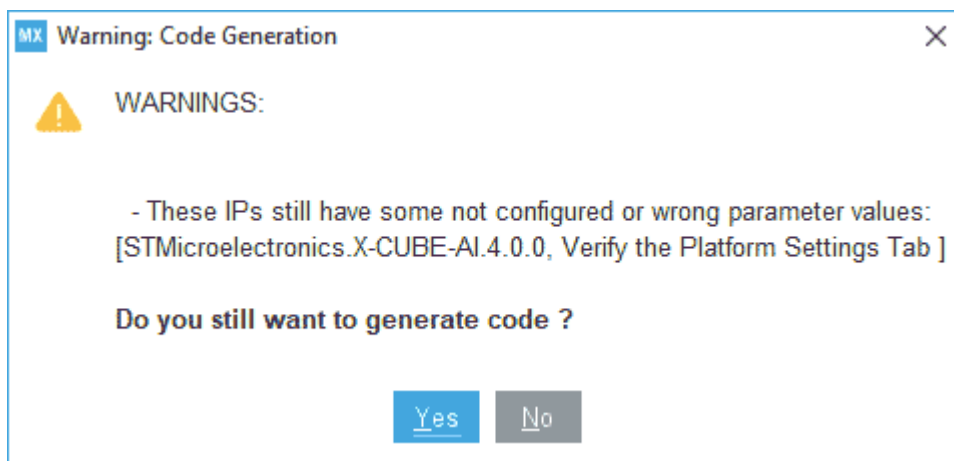
C:/tools/stm32cube/STM32Cube\_FW\_F7\_V1.14.0

Browse

Рис. 34. Настройки для генератора файлов IDE-проекта

Нажмите на кнопку «GENERATE CODE», чтобы сгенерировать код, соответствующий текущему проекту (включая файлы IDE-проекта).

Во время генерации файлов проекта может появиться окно с сообщением, показанное на рис. 35. Это говорит о том, что пользователь выбрал генерацию дополнительного приложения из состава пакета X-CUBE-AI, но не выполнил необходимые настройки целевой платформы (не сконфигурировал используемый модуль USART). Более подробно об этом сказано в разделе «Добавление компонента X-CUBE-AI».



На этом этапе окно STM32Cube MX можно закрыть. Вы в любой момент можете снова открыть файл <project\_name>.iос, чтобы включить и настроить новые периферийные модули, добавить программные компоненты или запустить процесс валидации на целевой плате.

## Сборка и программирование

После успешной генерации проекта сборка прошивки и загрузка ее в целевой микроконтроллер производится как обычно:

Запустите IDE и откройте сгенерированный файл проекта.

Выполните сборку и загрузите прошивку в микроконтроллер. Если вы собирали тестовое приложение, то никаких изменений в исходные файлы вносить не требуется. В противном случае в проект необходимо добавить код пользовательского приложения, который будет обращаться к сгенерированному API логического вывода.

## Внутреннее устройство X-CUBE-AI

### Оптимизатор алгоритма и занимаемой памяти

Оптимизатор генератора Си-кода ищет компромисс между размером используемой памяти (ОЗУ и ПЗУ) и временем, необходимым для формирования логического вывода (также учитывается и энергопотребление). При этом оптимизатор работает без входных данных, то есть для применения алгоритмов сжатия и оптимизации не требуются ни набор данных, использованный для обучения, ни тестовый набор данных. Соответственно, отсутствует этап переобучения/уточнения значений весов/смещений, который позволил бы сохранить точность исходной модели.

Сжатие значений весов/смещений при степени сжатия: «нет», x4, x8 (рис. 36);

имеет смысл только для плотных (полносвязных) слоев сети;

применяется алгоритм совместного использования весов (кластеризация методом K-средних);

при отсутствии сжатия («none») гарантируется точность исходной DL-модели. Остаточная погрешность ( $\sim 10^{-8}$ ) связана с переходом от 64-битных значений с плавающей точкой, используемых в модели, к 32-битным, используемым в языке Си. В случае больших нейронных сетей эта ошибка вполне может достигать значения  $10^{-6}$ .

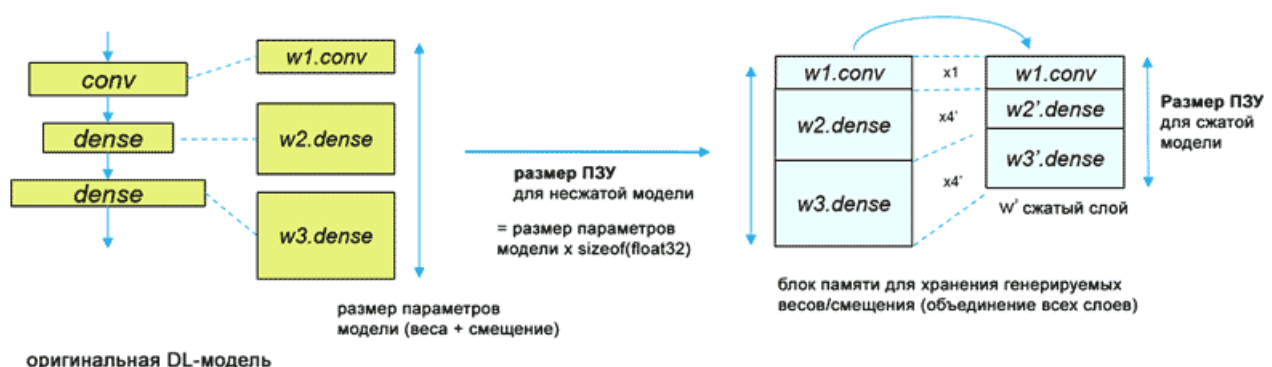


Рис. 36. Сжатие весов/смещений

Преимущество такого подхода заключается в большой скорости процесса сжатия, однако конечный результат не полностью идентичен исходному (сжатие производится с потерями), что может повлиять на общую точность. Поэтому был разработан процесс валидации сгенерированной модели на языке Си, позволяющий оценить величину ошибки (подробнее об этом рассказано в разделе «Механизм валидации (вычисление относительной ошибки L2)»):

Объединение операций. Объединяются два слоя для оптимизации размещения данных и соответствующего вычислительного ядра. Одни слои, такие как «Dropout», «Reshape», во время преобразования или оптимизации удаляются, другие, например, нелинейности и субдискретизации после сверточного слоя, объединяются с предыдущим слоем. В результате итоговая сеть зачастую имеет меньше слоев, чем исходная (рис. 37).

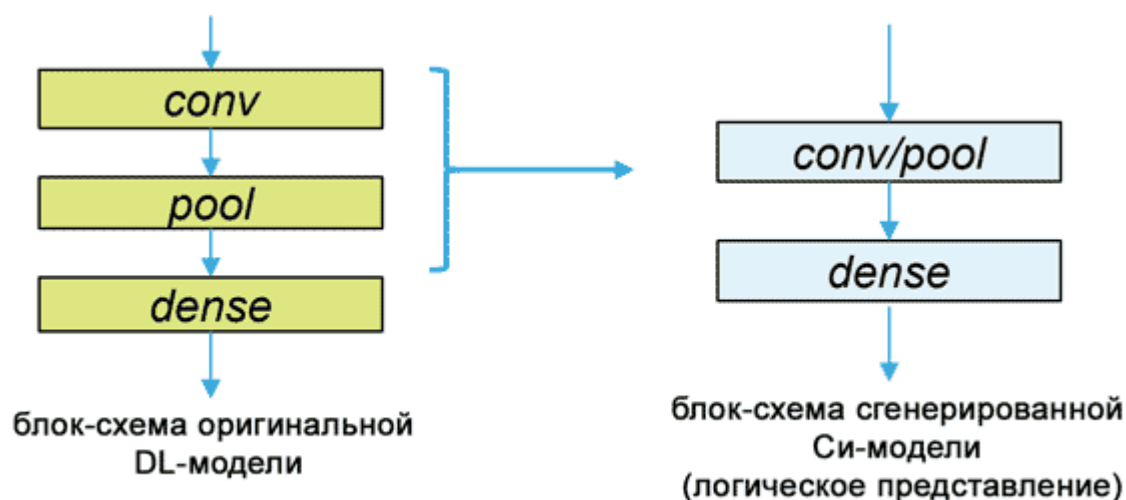


Рис. 37. Объединение операций

Оптимальное использование памяти активации/рабочей памяти. В ходе данного процесса выделяется специальный блок памяти для хранения значений временного скрытого слоя (выходы операторов активации). Его можно рассматривать как промежуточный буфер, используемый функцией логического вывода. Этот буфер многократно используется разными слоями, поэтому его размер определяется максимальным объемом памяти, который требуется двум последовательно выполняющимся слоям (рис. 38).

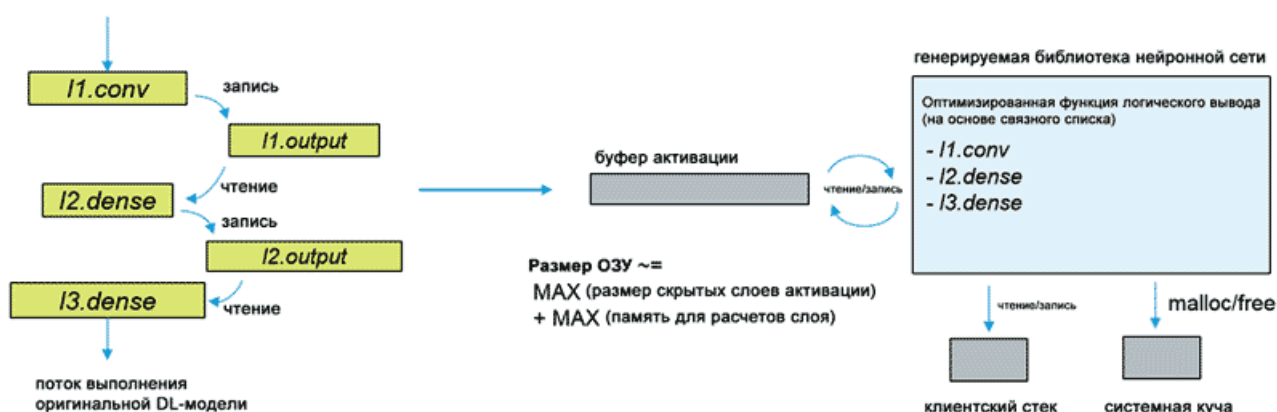


Рис. 38. Оптимальный размер рабочего буфера (буфер активации)

## Механизм валидации (вычисление относительной ошибки L2)

В пакете реализован простой и быстрый механизм валидации, позволяющий количественно сравнить точность сгенерированной модели на языке Си и загруженной DL-модели (рисунок 39). На вход обеих моделей подаются идентичные входные тензоры - фиксированный набор случайных данных либо пользовательский набор данных, [10]. Затем для всех логических выводов вычисляется относительная ошибка L2. Величина ошибки L2 выходного слоя менее 0.01 свидетельствует о корректности сгенерированной Си-модели. Для более точной оценки сгенерированной Си-модели во время валидации выводятся и другие метрики [10]. Пакет расширения X-CUBE-AI содержит механизм выполнения правил логического вывода для всех поддерживаемых фреймворков глубокого обучения. Обратите внимание: даже если Си-модель не пройдет валидацию, ее все равно можно использовать, несмотря на то, что ее точность будет хуже, чем точность оригинальной модели на языке Python™. Разумеется, необходимы дополнительные проверки, например, с использованием специальных наборов данных при наличии контроля результатов работы нейронной сети.

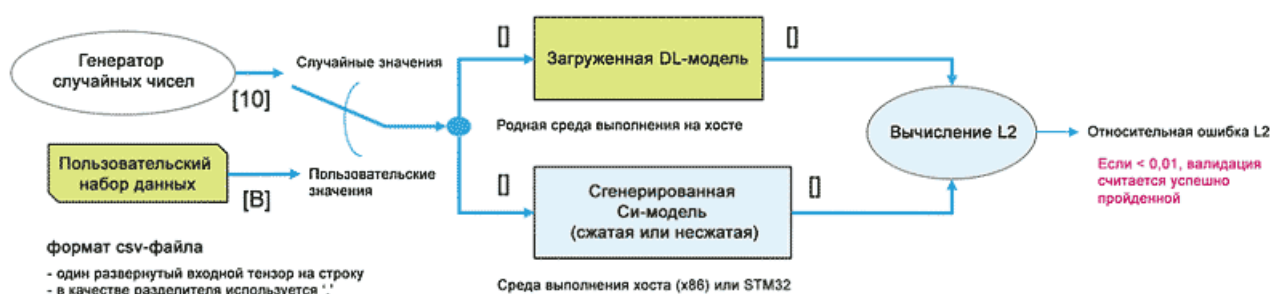


Рис. 39. Общая схема процесса валидации

Относительная ошибка L2 вычисляется по формуле:

где:

$F_j$  - плоские массивы  $j$ -го выхода слоя модели на языке Си;

$f_i$  - плоские массивы  $i$ -го выхода соответствующего исходного слоя.

Валидация может выполняться двумя способами:

**Валидация на ПК.** В этом режиме сравниваются результаты работы DL-модели с результатами полученной на ее основе Си-модели для платформы X. Все вычисления выполняются на хосте (ПК). Данные, которые выводятся в процессе валидации, подробно описаны в разделе «Валидация сгенерированной Си-модели».

**Валидация на целевой плате.** В этом режиме DL-модель сравнивается с Си-моделью, выполняющейся на целевой плате (рисунок 40). Для этого способа необходимо специальное тестовое приложение, которое включает в себя сгенерированную библиотеку нейронной сети и обработчик последовательного порта для обмена данными с хостом. Результаты работы этой программы и ее использование описаны в разделе «Приложение AI Validation».

Отличительные особенности валидации на целевой плате:

осуществляется автоматическое обнаружение подключенных плат с MKSTM32;

сигнатура встраиваемой сгенерированной Си-модели сличается с сигнатурой проверенной DL-модели;

ошибка L2 вычисляется только для последнего слоя сети (из-за низкой скорости обмена по последовательному интерфейсу);  
в отчете выводится дополнительная информация, в частности, время формирования логического вывода для каждого слоя (подробнее об этом сказано в разделе «Приложение AI Validation»).

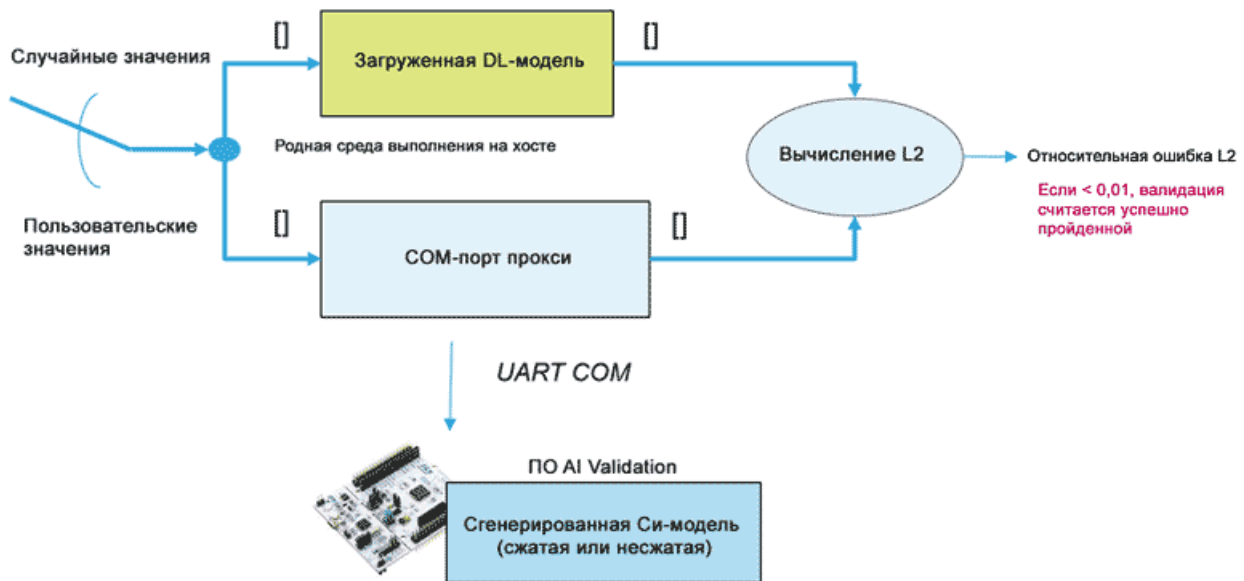


Рис. 40. Валидация на целевой плате

## Генерируемая библиотека нейронной сети для STM32

Для каждой импортируемой DL-модели генерируются только специальные (зависящие от DL-модели) файлы на языке Си. В качестве префикса имен этих файлов используется имя сети, заданное пользователем (подробнее об этом сказано в разделе «Загрузка файла предварительно обученной DL-модели»). Все эти файлы используют внутренний закрытый API, реализованный библиотекой `network_runtime.a`:

файлы `<name>.c` и `<name>.h` с топологией нейронной сети;  
файлы `<name>_data.c` и `<name>_data.h` для весов/смещений.

Примечание. Для всех наборов инструментальных средств и семейств МК STM32 создаются одни и те же специальные файлы.

Библиотека, или, другими словами, вычислительное ядро нейронной сети `network_runtime.a`, предоставляется в виде статической библиотеки:

все неиспользуемые идентификаторы и методы удаляются на этапе линковки;  
не используется интерпретация сетевого графа (в отличие от SDKARM-NN), поскольку данный подход не оптимален для устройств с ограниченным объемом памяти.

## Интеграция встраиваемого ПО

На рис. 41 показано, каким образом генерируемый пакет поддержки нейронной сети интегрируется в программу для МК (включая зависимости времени выполнения).

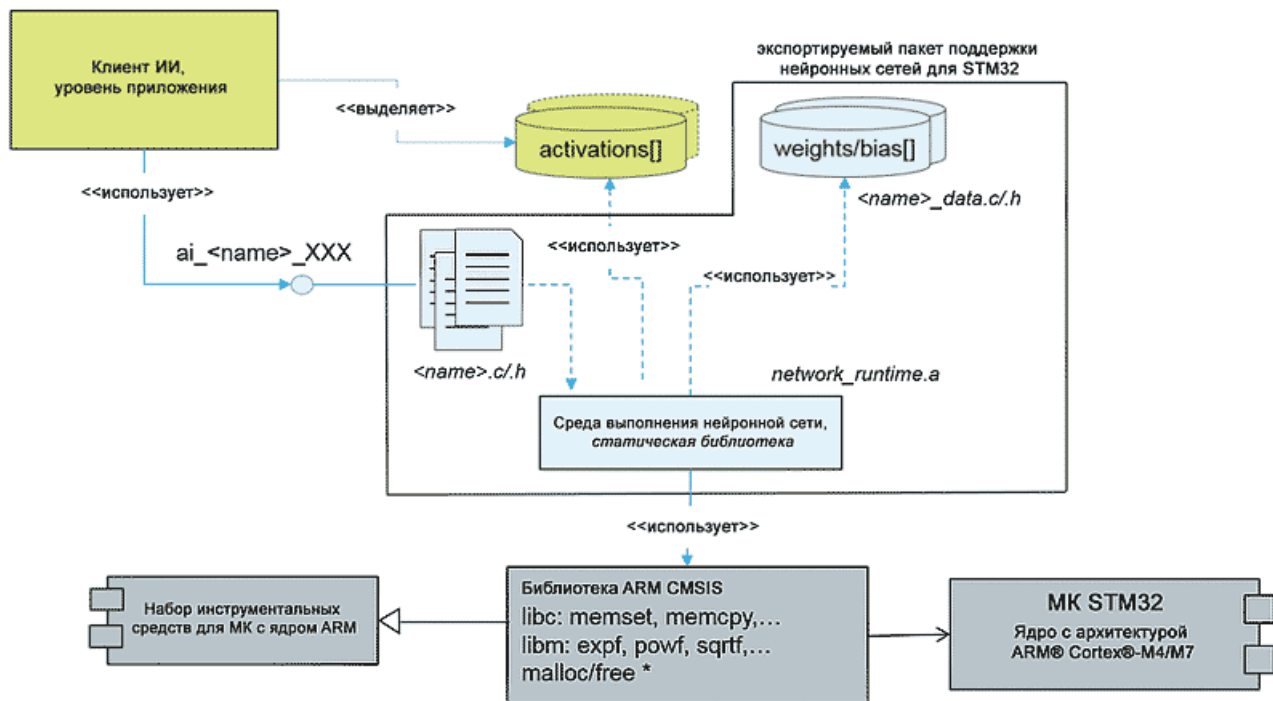


Рис. 41. Интеграция со встроенным ПО

Для пользовательского приложения экспортированная библиотека нейронной сети может рассматриваться как «черный ящик» или самодостаточный программный объект. Исходный код содержится только в специальных файлах: топологии сети (файлы `<name>.c` и `<name>.h`) и значений весов/смещений (файлы `<name>_data.c` и `<name>_data.h`). Все эти файлы базируются на общей библиотеке времени выполнения `network_runtime.a`. Зависимости от среды выполнения минимальны:

стандартные функции работы с памятью библиотеки `libc` (`memcpy`, `memset`). Как правило, они предоставляются пакетом инструментальных средств для МК;

Библиотека CMSIS для поддержки оптимизированных операций Cortex®-M (FPU- и DSP-команды). Она входит в состав пакета STM32 HAL;

Функции `malloc/free`, которые в настоящий момент необходимы для поддержки слоев рекуррентного типа (слои «GRU» и «LSTM»). В будущих версиях планируется использовать статическое выделение памяти. Влияние на производительность нивелируется числом рекуррентных ячеек и соответствующего времени обработки;

также требуется математическая библиотека (с поддержкой DSP/FPU), реализующая функции `expf`, `powf`, `tanhf` и `sqrtf`;

минимальный стек (реальное значение можно измерить с использованием тестового приложения «AI system performance» (о чем подробно рассказано в разделе «Приложение «AI system performance»)).

Примечание. Все внешние зависимости должны разрешаться на этапе линковки с конечным приложением пользователя (при формировании образа прошивки). Буферы памяти активации могут быть размещены динамически в куче или как глобальный массив (секции `.bss` и `.data`). Ознакомьтесь с описанием функции `ai_<name>_init()`, чтобы понять, как передавать веса/смещения в функции библиотеки нейронной сети.



При создании IDE-проекта библиотека нейронной сети экспортируется в отдельную подпапку <project\_name>/Middlewares/ST/AI/. Специальные файлы нейронных сетей сохраняются в стандартных для STM32CubeMX подпапках Inc и Src. Также в проект добавляются специфические файлы из библиотек CMSIS-DSP.

Конкретное имя файла `network_runtime.a` зависит от версии пакета X-CUBE-AI и используемой IDE, например, `NetworkRuntime410_CM4_GCC.a` или `NetworkRuntime410_CM4_IAR.a`.

### API логического вывода для нескольких сетей

В файлах `app_x-cube-ai.c` и `app_x-cube-ai.h` содержится реализация общего API логического вывода с поддержкой нескольких сетей, который может использоваться клиентским приложением. Этот API практически идентичен оригинальному клиентскому API логического вывода для встраиваемых приложений [9], отличается только функция `create()`. Для создания экземпляра сети в эту функцию необходимо передать Си-строку, содержащую имя сети. Данный интерфейс, в основном, используется дополнительными тестовыми приложениями, обеспечивая единообразный механизм взаимодействия с разными встроенными нейронными сетями.

### Возможность одновременного входа и потокобезопасность

В библиотеке не предусмотрено никаких механизмов синхронизации для защиты точек входа от одновременного обращения. Если API используется в многопоточном окружении, защита экземпляра сети должна обеспечиваться самим пользовательским приложением.

Для экономии ОЗУ один и тот же блок памяти (`SizeSHARED`) может использоваться несколькими сетями. В этом случае пользователь должен гарантировать, что текущий процесс формирования логического вывода ни при каких условиях не будет прерван обращением к другой сети.

Примечание. Если предполагается возможность такого прерывания (из-за ограничений реального времени или задержек), то каждая сеть должна использовать индивидуальный буфер активации.

### Соглашения по размещению кода и данных

В текущей архитектуре памяти STM32 (семейства STM32L4/STM32F4/STM32F3 и STM32F7/STM32H7) расположение кода и данных никак не влияет на быстродействие. Модуль Flash ART и подсистема кэширования ядра ARM® (архитектура Cortex®-M7) эффективно нивелируют влияние задержек при обращении к памяти. Код нейронной сети (секция `.text`) и константы (секция `.rodata`) могут располагаться во внутренней Flash-памяти. Данные (секции `.data` и `.bss`) должны располагаться во внутреннем ОЗУ. При работе библиотек используется стек клиентского приложения, который должен располагаться в быстродействующей памяти без циклов ожидания.

Примечание. К долговременности хранения данных в буфере активации не предъявляется никаких требований. Его действительно можно рассматривать как буфер для хранения временных значений. Соответственно, между формированием двух логических выводов



буфер можно использовать, к примеру, для предварительной обработки данных. А при переключении устройства в режим глубокого сна соответствующую микросхему памяти можно вообще отключить.

## Отладка

Библиотеку следует рассматривать как оптимизированный черный ящик в двоичном формате (исходные коды не предоставляются). Нет никакой возможности узнать значение внутренних данных или внутреннее состояние во время работы. Корректность переноса нейронной сети между семействами МК гарантируется генератором пакета X-CUBE-AI. Некоторую косвенную информацию можно получить, используя функцию `ai_<name>_get_error()`.

## Встраиваемый клиентский API логического вывода

Чтобы пользователь мог применить сгенерированный код нейронной сети, одновременно с библиотекой генерируется и простой клиентский API (функции с именами вида `ai_<name>_XX()` на рисунке 41). Объявления этих функций находятся в файле `<project_name>/Src/<name>.h`. Имена всех функций и макроопределений генерируются на основе имени сети, заданного пользователем. Подробно этот API описан в материале [9].

## Приложение «AI system performance»

Приложение «AI system performance» представляет собой независимое bare-metal-приложение, исполняемое на целевой плате, которое позволяет определить основные параметры сгенерированной нейронной сети, критичные для ее интеграции с пользовательским приложением. Точностные характеристики сети в этом приложении не проверяются. В отчете, который формируется в результате работы приложения, содержится следующая информация:

скорость формирования логического вывода («duration» – длительность в мс, «CPUCycles» – тактов ЦПУ, «CPUWorkload» – загрузка ЦПУ в процентах);  
потребление памяти в байтах («stack» – стек, «heap» – куча).

Для запуска приложения следует выполнить следующие действия:

Откройте и сконфигурируйте на хосте терминальную программу, подключенную к COM-порту (обычно реализованному в виде виртуального порта на шине USB).

Настройте COM-порт на ПК. Его настройки должны быть идентичны настройкам модуля USART STM32 (более подробно об этом сказано в разделе «Настройки аппаратной и программной платформ»):

115200 бод;

8 бит;

1 стоп-бит;

без контроля четности.

Выполните сброс платы для запуска приложения.

Во время работы приложения введите в консоли «r» или «R», чтобы приостановить выполнение основного цикла. В приложении реализована примитивная консоль, поддерживающая команды, показанные на рис. 42.

```

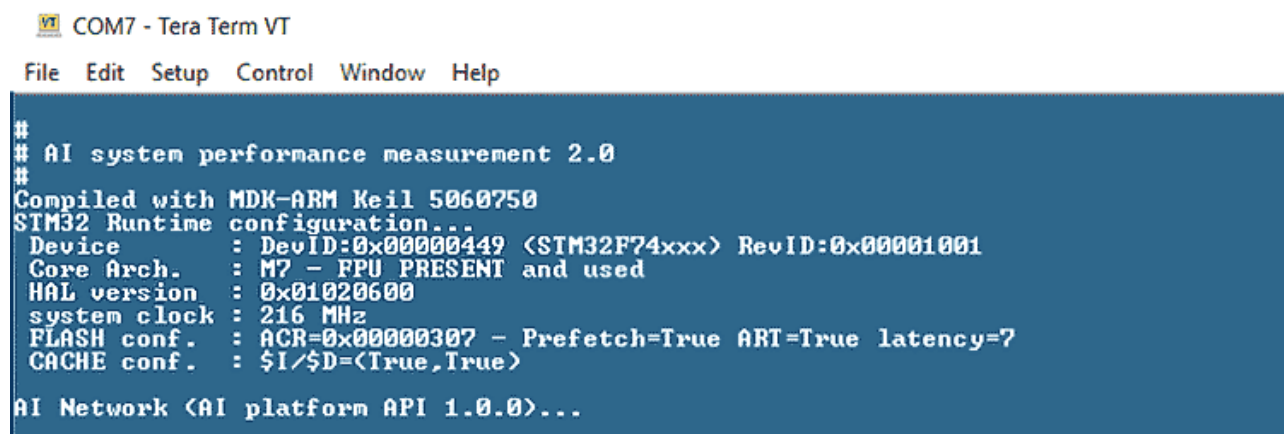
Possible key for the interactive console:
[q,Q]    quit the application
[r,R]    re-start (NN de-init and re-init)
[p,P]    pause
[h,H,?]  this information
xx       continue immediately

```

Рис. 42. Команды приложения «AI system performance»

### Системная информация времени выполнения

На рисунках 43 и 44 показана начальная часть журнала, в которой содержится информация о рабочем окружении для случая использования сред разработки Keil® и Atollic соответственно: идентификатор устройства, тактовая частота, использованный набор инструментальных средств и прочее.



```

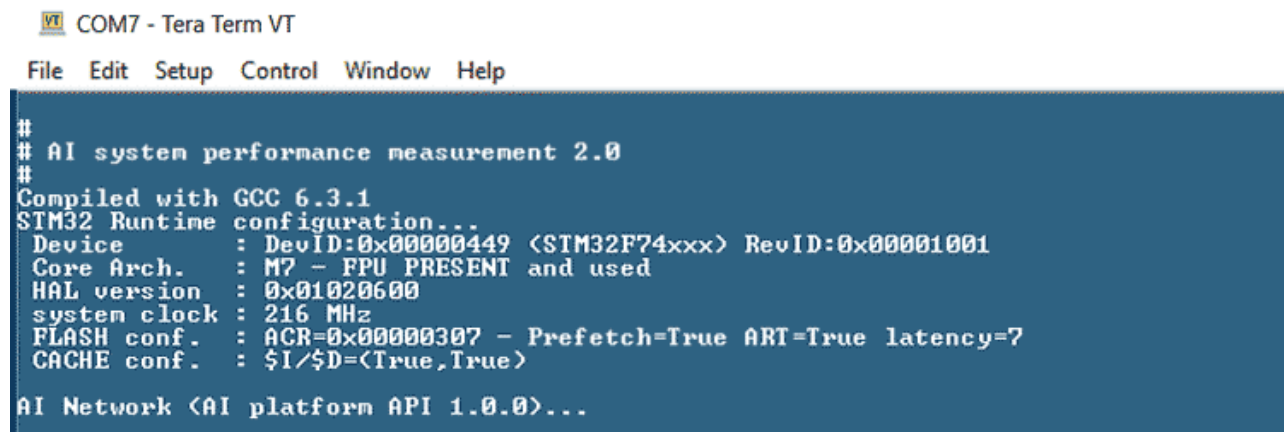
COM7 - Tera Term VT
File Edit Setup Control Window Help

#
# AI system performance measurement 2.0
#
Compiled with MDK-ARM Keil 5060750
STM32 Runtime configuration...
Device       : DevID:0x00000449 <STM32F74xxx> RevID:0x00001001
Core Arch.   : M7 - FPU PRESENT and used
HAL version  : 0x01020600
system clock : 216 MHz
FLASH conf.  : ACR=0x00000307 - Prefetch=True ART=True latency=7
CACHE conf.  : $I/$D=<True,True>

AI Network <AI platform API 1.0.0>...

```

Рис. 43. Системная информация Keil® IDE



```

COM7 - Tera Term VT
File Edit Setup Control Window Help

#
# AI system performance measurement 2.0
#
Compiled with GCC 6.3.1
STM32 Runtime configuration...
Device       : DevID:0x00000449 <STM32F74xxx> RevID:0x00001001
Core Arch.   : M7 - FPU PRESENT and used
HAL version  : 0x01020600
system clock : 216 MHz
FLASH conf.  : ACR=0x00000307 - Prefetch=True ART=True latency=7
CACHE conf.  : $I/$D=<True,True>

AI Network <AI platform API 1.0.0>...

```

Рис. 44. Системная информация Atollic IDE

Примечание. Чтобы увидеть эту информацию, введите в консоли «r» или «R» во время выполнения основного цикла.

### Информация о встраиваемой Си-модели нейронной сети

Во второй части журнала, показанной на рисунке 45, содержатся основные статические характеристики сгенерированной нейронной сети (сетей). В частности, выводится

информация о необходимом размере ОЗУ/Flash (поля «activation»/«weights», соответственно) и о логической сложности сети (поле «complexity»). Также выводится информация о параметрах входных и выходных тензоров. Клиентское приложение тоже может получить эту информацию с помощью функции `APLai_<name>_get_info()`.

```
Found network "net1"
Creating the network "net1"..
Network configuration...
Model name       : net1
Model signature   : dc582ba86ee8a11fd06b698b258a4310
Model datetime    : Sun Dec  9 08:56:25 2018
Compile datetime  : Dec  9 2018 09:01:15
Runtime revision  : (3.3.0)
Tool revision     : (rev-) (3.3.0)
Network info...
signature         : 0x0
nodes            : 7
complexity        : 874970 MACC
activation        : 45572 bytes
weights          : 794136 bytes
inputs/outputs    : 1/1
IN tensor format  : HWC layout:90,3,1 (s:270 f:AI_BUFFER_FORMAT_FLOAT)
OUT tensor format : HWC layout:1,1,6 (s:6 f:AI_BUFFER_FORMAT_FLOAT)
Initializing the network

Found network "net2"
Creating the network "net2"..
Network configuration...
Model name       : net2
Model signature   : b773f449281f9d970d5b982fb57db61f
Model datetime    : Sun Dec  9 08:57:12 2018
Compile datetime  : Dec  9 2018 09:01:15
Runtime revision  : (3.3.0)
Tool revision     : (rev-) (3.3.0)
Network info...
signature         : 0x0
nodes            : 21
complexity        : 4550024 MACC
activation        : 64004 bytes
weights          : 159536 bytes
inputs/outputs    : 1/1
IN tensor format  : HWC layout:49,10,1 (s:490 f:AI_BUFFER_FORMAT_FLOAT)
OUT tensor format : HWC layout:1,1,12 (s:12 f:AI_BUFFER_FORMAT_FLOAT)
Initializing the network
```

Рис. 45. Информация о Си-модели нейронной сети

Примечание. Чтобы увидеть эту информацию в журнале, введите в консоли «г» или «R» во время выполнения основного цикла.

### Производительность встраиваемой Си-модели нейронной сети

Как показано на рисунках 46 и 47, в последней части журнала (основной цикл) содержится информация об измеренной производительности всей системы. Для измерения количества тактов ЦПУ, необходимых для формирования логического вывода (поле «CPU cycles»), на вход нейронной сети подаются случайные данные. На основе этого значения вычисляются значения полей «CPU Workload» и «cycles/MACC». На время измерения все прерывания МК запрещаются:

duration показывает время формирования одного логического вывода в мс;

CPU cycles показывает число тактов ЦПУ, затраченных на формирование одного логического вывода;

CPU Workload отражает степень загрузки ЦПУ в течение 1 с;

cycles/MACC отражает число тактов ЦПУ, необходимое для выполнения одной операции MACC.

```
Running PerfTest on "net1" with random inputs (16 iterations)...
.....
Results for "net1", 16 inferences @216MHz/216MHz (complexity: 874970 MACC)
duration      : 28.047 ms (average)
CPU cycles    : 6058169 -198595/+29532 (average,-/+>)
CPU Workload  : 2%
cycles/MACC   : 6 (average for all layers)
used stack    : 352 bytes
used heap     : 0:0 0:0 (req:allocated,req:released) cfg=0

Running PerfTest on "net2" with random inputs (16 iterations)...
.....
Results for "net2", 16 inferences @216MHz/216MHz (complexity: 4550024 MACC)
duration      : 150.323 ms (average)
CPU cycles    : 32469972 -42110/+14510 (average,-/+>)
CPU Workload  : 15%
cycles/MACC   : 7 (average for all layers)
used stack    : 352 bytes
used heap     : 0:0 0:0 (req:allocated,req:released) cfg=0
```

Рис. 46. Производительность Си-модели

```
Running PerfTest on "network" with random inputs (16 iterations)...
.....
Results for "network", 16 inferences @80MHz/80MHz (complexity: 3729752 MACC)
duration      : 569.124 ms (average)
CPU cycles    : 45529988 -134210/+211019 (average,-/+>)
CPU Workload  : 56%
cycles/MACC   : 12 (average for all layers)
used stack    : 284 bytes
used heap     : 16:29568 16:29568 (req:allocated,req:released) cfg=3
```

Рис. 47. Производительность Си-модели с контролем кучи и стека

Примечание. В строке «usedheap» выводится число вызовов функции malloc() и общий объем выделенной памяти, а также число вызовов функции free() и общий объем освобожденной памяти, произведенных при формировании всех логических выводов. Между двумя выводами или тестовыми итерациями счетчик не сбрасывается, что позволяет обнаружить возможную утечку памяти. Для системы, журнал которой приведен на рисунке 47, минимальный размер кучи составляет около 2 кбайт (29568 байт на итерацию).

Обратите внимание: на данный момент монитор кучи доступен только для сред разработки на основе GCC.

## Приложение «AI validation»

Приложение «AI validation» представляет собой независимое приложение, исполняемое на целевой плате, которое обеспечивает валидацию нейронной сети на целевой платформе, как было описано в разделе «Механизм валидации (вычисление относительной ошибки L2)». Данное приложение использует последовательный интерфейс на базе модуля USART для экспорта API логического вывода на хост.

Само приложение «AI validation» создается и загружается в целевой МК как обычный IDE-проект либо автоматически компилируется и загружается из самой среды STM32CubeMX (рисунок 48).

Если это приложение не было создано автоматически, необходимо выполнить следующие действия:

Если МК на плате содержит программу - сбросьте или перезапустите его.

Откройте (если уже закрыли) файл с расширением ioc, который был создан одновременно с генерацией встроенного ПО.

Перейдите на панель конфигурирования X-CUBE-AI и откройте вкладку <network\_name>, соответствующую той сети, корректность которой нужно проверить.

Нажмите кнопку «Validation on target», чтобы начать процесс валидации. Перед тем как нажать кнопку «ОК», пользователь может указать используемый COM-порт, как показано на рисунке 48. В противном случае будут поочередно опрошены все доступные COM-порты (хост будет использовать первую найденную плату).

Validation on target

Use communication port: ☒ Automatic ☐ Manual Baudrate: 115,200

Automatic compilation and download

☒ Enabled Communication port on the target: ST-Link TX pin: PD8 RX pin: PD9

Toolchain / IDE: EWARM V8

MCU Debug interface: SWD

OK Cancel

Рис. 48. Выбор COM-порта хоста для валидации на плате

Как и в случае валидации на ПК, итоговый результат выводится в поле «Validation status». Более подробная информация выводится в окне журнала «Output» STM32CubeMX.

### Системная информация времени выполнения

В начале журнала, как показано на рис. 49, содержится основная информация о системе: идентификатор устройства, тактовая частота МК, конфигурация подсистемы памяти, перечень внедренных нейронных сетей. Для проверяемой сети дополнительно выводится описание формата входных и выходных тензоров, а также версии используемых инструментальных средств ИИ.

```
MCUs Selection  Output
ON-DEVICE STM32 execution ("net1", default, 115200)..

<Stm32com id=0x1bfd54ec0f0 - CONNECTED(COM7/115200) devid=0x449/STM32F74xxx msg=1.0>
0x449/STM32F74xxx @72MHz/72MHz (FPU is present) lat=2 Core:I$/D$ ART: PRFTen ARTen
found network(s): ['net1', 'net2']
description      : 'net1' (90, 3, 1)-[7]->(1, 1, 6) macc=874970 rom=775.52KiB ram=44.50KiB
tools versions  : rt=(3, 3, 0) tool=(3, 3, 0)/(1, 1, 0) api=(1, 0, 0) "Sat Dec 8 23:55:41 2018"

Running with inputs=(10, 90, 3, 1)..
..... 1/10
..... 2/10
```

Рис. 49. Системная информация времени выполнения

## Производительность встраиваемой Си-модели нейронной сети

Во второй части журнала, как показано на рисунке 50, содержится информация об измеренной производительности всей системы (duration – среднее время формирования одного логического вывода в мс). На основе этого значения вычисляется параметр «cycles/MACC». На время измерения все прерывания МК запрещаются:

duration показывает время формирования одного логического вывода в мс;

CPU cycles показывает число тактов ЦПУ, затраченных на формирование одного логического вывода;

cycles/MACC показывает число тактов ЦПУ, необходимое для выполнения одной операции MACC.

```
..... 9/10
..... 10/10
RUN Stats      : batches=10 dur=24.266s tfx=21.957s 0.491KiB/s (wb=10.547KiB,rb=240B)

Results for 10 inference(s) @72/72MHz (macc=874970)
duration      : 78.846 ms (average)
CPU cycles    : 5676924 (average)
cycles/MACC   : 6.49 (average for all layers)
```

Рис. 50. Производительность Си-модели

## Производительность отдельных слоев нейронной сети

В следующей части журнала, как показано на рис. 51, содержится дополнительная информация о сгенерированной Си-модели: имя и тип реализованного слоя (Clayer/id/desc), выходной формат (oshape) и среднее время формирования логического вывода (мс).



```
Inspector report (layer by layer)
signature      : EF7C5473
n_nodes        : 7
num_inferences : 10
```

Clayer	id	desc	oshape	ms
0	0	10011/ (Merged Conv2d / Pool)	(10, 44, 1, 128)	24.277
1	4	10005/ (Dense)	(10, 1, 1, 128)	53.165
2	4	10009/ (Nonlinearity)	(10, 1, 1, 128)	0.010
3	5	10005/ (Dense)	(10, 1, 1, 128)	1.303
4	5	10009/ (Nonlinearity)	(10, 1, 1, 128)	0.010
5	6	10005/ (Dense)	(10, 1, 1, 6)	0.066
6	6	10014/ (Softmax)	(10, 1, 1, 6)	0.015
				78.846 (total)

Рис. 51. Производительность по слоям – валидация на целевой плате

### Результат валидации на целевой плате

Последняя часть журнала, показанная на рис. 52, содержит конечный результат процесса валидации. Содержимое этой части журнала аналогично содержимому отчета при валидации на ПК, за исключением того, что ошибка L2 выводится только для последнего слоя (подробнее об этом сказано в разделе «Механизм валидации (вычисление относительной ошибки L2)»).

```
MACC / frame: 874970
ROM size:      775.52 KBytes
RAM size:      44.50 KBytes (Minimum: 44.50 KBytes)

Matching criteria: L2 error < 0.01 on the output tensor

Ref layer 6 matched with C layer 6, error: 0.0010825497

Validation: OK
```

Рис. 52. Последняя часть журнала валидации на целевой плате

## Ошибки, возникающие при подключении

По умолчанию используются следующие настройки модуля USART:

- 8 бит;
- 1 стоп-бит;
- без контроля четности;
- 115200 бод.

Если скорость обмена будет изменена пользователем, необходимо ту же скорость задать и для остальных частей системы (для получения подробной информации обратитесь к разделу «Настройки аппаратной и программной платформ»), как показано на рисунке 48.

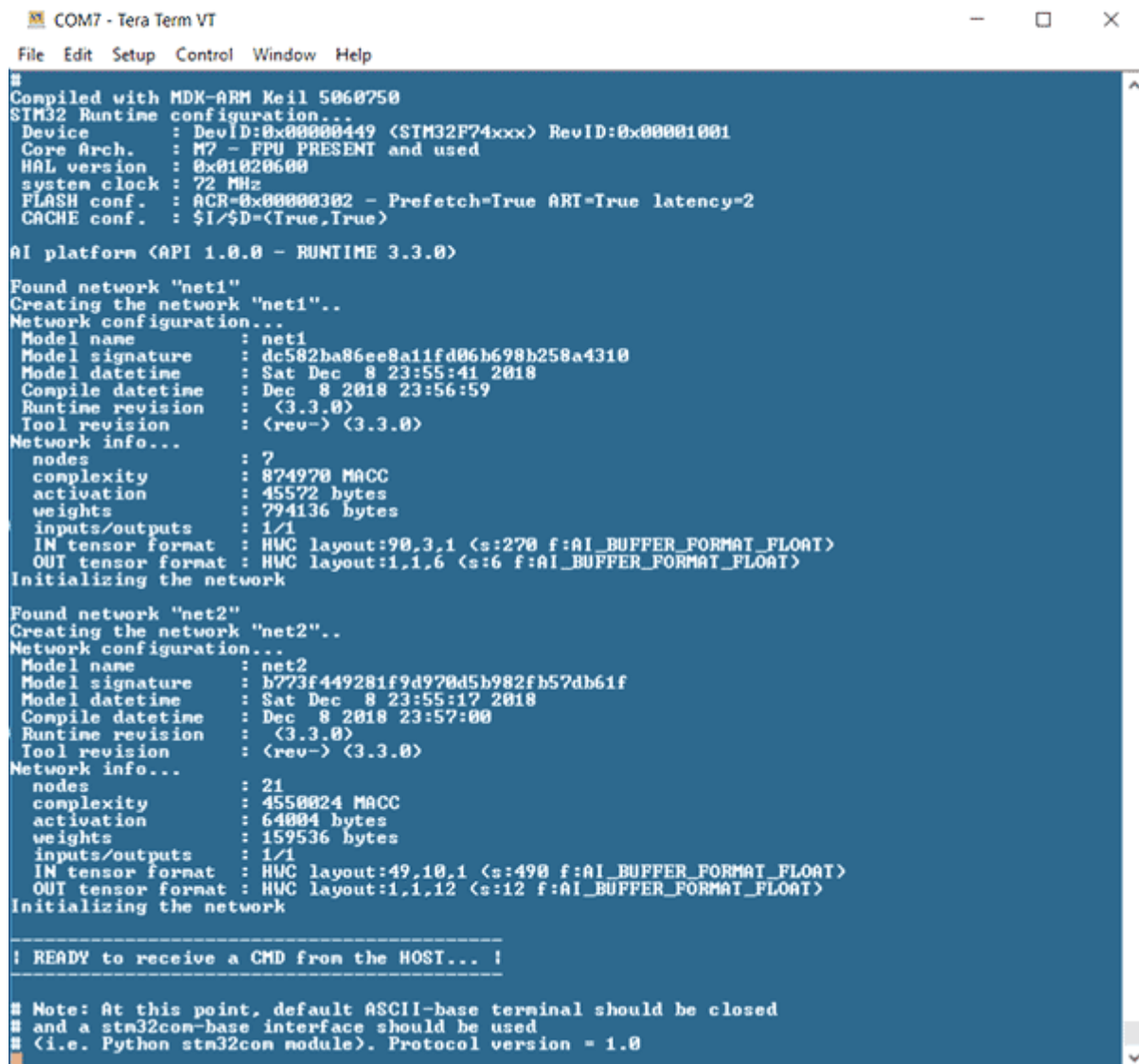
Ошибка: no connected board, invalid firmware or board restart needed



Данная ошибка говорит о том, что плата не подключена к СОМ-порту, либо ПК не может ее определить, или же на ней выполняется вовсе не приложение «AI validation».

Эта ошибка также может появиться в том случае, если прошивка по какой-то причине находится в некорректном состоянии. В этом случае следует перезапустить плату.

Чтобы убедиться в том, что прошивка корректно загружена в целевой МК, откройте на ПК окно терминала последовательного порта и подайте питание на плату. Вы должны увидеть журнал загрузки (рисунок 53). Не забудьте закрыть соединение перед повторным запуском процесса валидации.



```
COM7 - Tera Term VT
File Edit Setup Control Window Help

#
Compiled with MDK-ARM Keil 5060750
STM32 Runtime configuration...
Device       : DevID:0x00000449 <STM32F74xxx> RevID:0x00001001
Core Arch.   : M7 - FPU PRESENT and used
HAL version  : 0x01020600
System clock : 72 MHz
FLASH conf.  : ACR=0x00000302 - Prefetch=True ART=True latency=2
CACHE conf.  : $I/$D=<True,True>

AI platform <API 1.0.0 - RUNTIME 3.3.0>

Found network "net1"
Creating the network "net1"..
Network configuration...
Model name       : net1
Model signature   : dc582ba86ee8a11fd06b698b258a4310
Model datetime    : Sat Dec 8 23:55:41 2018
Compile datetime  : Dec 8 2018 23:56:59
Runtime revision  : <3.3.0>
Tool revision     : <rev-> <3.3.0>
Network info...
nodes            : 7
complexity       : 874970 MACC
activation       : 45572 bytes
weights          : 794136 bytes
inputs/outputs   : 1/1
IN tensor format : HWC layout:90,3,1 <s:270 f:AI_BUFFER_FORMAT_FLOAT>
OUT tensor format: HWC layout:1,1,6 <s:6 f:AI_BUFFER_FORMAT_FLOAT>
Initializing the network

Found network "net2"
Creating the network "net2"..
Network configuration...
Model name       : net2
Model signature   : b773f449281f9d970d5b982fb57db61f
Model datetime    : Sat Dec 8 23:55:17 2018
Compile datetime  : Dec 8 2018 23:57:00
Runtime revision  : <3.3.0>
Tool revision     : <rev-> <3.3.0>
Network info...
nodes            : 21
complexity       : 4550024 MACC
activation       : 64004 bytes
weights          : 159536 bytes
inputs/outputs   : 1/1
IN tensor format : HWC layout:49,10,1 <s:490 f:AI_BUFFER_FORMAT_FLOAT>
OUT tensor format: HWC layout:1,1,12 <s:12 f:AI_BUFFER_FORMAT_FLOAT>
Initializing the network

=====
! READY to receive a CMD from the HOST... !

# Note: At this point, default ASCII-base terminal should be closed
# and a stm32com-base interface should be used
# (i.e. Python stm32com module). Protocol version = 1.0
```

Рис. 53. Корректная прошивка: журнал начальной загрузки

### Ошибка: «network\_name» is not a valid network

Эта ошибка говорит о том, что ожидаемая Си-модель, идентифицируемая по заданному имени, отсутствует в прошивке подключенной платы. Дополнительные сведения смотрите в окне журнала «Output» STM32CubeMX

### Ошибка: the embedded STM32 model does not match the Cmodel

Данная ошибка говорит о том, что сигнатура сгенерированной Си-модели не соответствует ожидаемой. Параметры, используемые для контроля сигнатуры:

размер ОЗУ/ПЗУ;

значение МАСС;

число узлов;

версии инструментальных средств.

Дополнительные сведения смотрите в окне журнала «Output» STM32CubeMX

## Приложение «AI template»

При выборе данной опции, несмотря на ее название, генерируемый IDE-проект на самом деле не содержит законченного приложения, которое бы демонстрировало использование сгенерированной Си-модели нейронной сети. В качестве такого приложения можно использовать файлы aiSystemPerformance.h и aiSystemPerformance.c. В данном же случае создаются только специальные файлы, а созданный проект можно использовать как отправную точку при разработке нового bare-metal приложения с двумя точками входа (функции initprocess) [9].

## Поддерживаемые фреймворки глубокого обучения и типы слоев нейронных сетей

Ядро X-CUBE-AI в настоящее время поддерживает следующие фреймворки глубокого обучения:

Keras: [//keras.io/](https://keras.io/);

Lasagne: [//lasagne.readthedocs.io/en/latest/](https://lasagne.readthedocs.io/en/latest/);

Caffe: [//caffe.berkeleyvision.org/](https://caffe.berkeleyvision.org/);

ConvNetJs: [//cs.stanford.edu/people/karpathy/convnetjs/](https://cs.stanford.edu/people/karpathy/convnetjs/);

TensorFlow™ Lite: [www.tensorflow.org/lite/](https://www.tensorflow.org/lite/).

Для каждого фреймворка поддерживается только ограниченный набор слоев и их параметров в зависимости от возможностей API нейронной сети на языке Си и от реализации парсера для конкретного фреймворка [8].

## Обработка ошибок

Ядро X-CUBE-AI способно обнаруживать разные ошибки и сообщать о них пользователю [7].

## Часто задаваемые вопросы

Как использовать файлы журнала при отладке?

В процессе выполнения валидации на целевой плате все сообщения, пересылаемые между хостом и платой, включая данные, сохраняются в отдельном файле журнала: C:\Users\\.stm32cubemx\ai\_stm32\_msg.log

В случае проблем при валидации или генерации файлов дополнительную отладочную информацию можно найти в файле C:\Users\\.stm32cubemx\STM32CubeMX.log

Какие существуют ограничения на использование нескольких сетей?

Единственным ограничением, которое может воспрепятствовать использованию нескольких сетей, является доступный размер ОЗУ и Flash-памяти. Каждая нейронная сеть имеет свой собственный набор функций `ai_<name>_XXX()` (подробнее об этом читайте в разделе «Встраиваемый клиентский API логического вывода»).

Если буфер активации используется одновременно несколькими сетями, необходимо соблюдать осторожность и исключить возможность обращения к одной сети при работе другой (обратитесь к разделу «Возможность одновременного входа и потокобезопасность» для получения более подробной информации).

Ошибка: Unable to compile file “arm\_dot\_prod\_f32.c”

Компиляция файла `arm_dot_prod_f32.c` может пройти неудачно в случае повторного создания файлов IDE-проекта:

```
compiling arm_dot_prod_f32.c...
```

```
../Drivers/CMSIS/Include/arm_math.h(314): error: #35:
```

```
#error directive: "Define according the used Cortex core ARM_MATH_CM7, ARM_MATH_CM4, ARM_MATH_CM3, ARM_MATH_CM0PLUS or ARM_MATH_CM0"
```

```
#error "Define according the used Cortex core ARM_MATH_CM7, ARM_MATH_CM4, ARM_MATH_CM3, ARM_MATH_CM0PLUS or ARM_MATH_CM0"
```

```
../Drivers/CMSIS/DSP_Lib/Source/BasicMathFunctions/arm_dot_prod_f32.c: 0 warnings, 1 error
```

В зависимости от целевого устройства STM32, необходимо в настройках проекта переопределить следующие макросы:

```
ARM_MATH_CM7, __FPU_PRESENT=1
```

Ошибка: Used heap or stack: disabled or not yet supported

Данное сообщение в журнале говорит о том, что монитор стека (и, соответственно, кучи) отключен или вовсе не реализован в используемом наборе инструментальных средств.

Таблица 4. Поддержка мониторинга кучи и стека

Набор инструментальных средств	Монитор стека	Монитор кучи	Примечание
GCC	Поддерживается	Поддерживается	SW4STM32
IAR™ 8.x и 7.x	Поддерживается	Не реализован	-
MDK-ARM	Не реализован	Не реализован	-

Пример активации мониторинга кучи и стека:

```
/* @file: aiSystemPerformance.c */
```

```
...
```

```

#if defined(__GNUC__)
#define _APP_STACK_MONITOR_ 1
#define _APP_HEAP_MONITOR_ 1
#elif defined (__ICCARM__)
#define _APP_STACK_MONITOR_ 1
#define _APP_HEAP_MONITOR_ 0
#else
#define _APP_STACK_MONITOR_ 0
#define _APP_HEAP_MONITOR_ 0
#endif
...

```

Почему значение “usedheap” всегда равно нулю?

Только для проектов на базе GCC:

used heap : 0:0 0:0 (req:allocated,req:released) cfg=0

Такой результат вовсе не обязательно свидетельствует о проблеме. Большая часть кода библиотеки network\_runtime.a использует предварительно выделенные буферы (буферы активации). Для определенных слоев (рекуррентного типа) текущая реализация библиотеки требует динамического выделения дополнительных рабочих буферов с использованием функции malloc().

Монитор кучи использует внутренние механизмы набора инструментальных средств, которые позволяют создавать обертки системных вызовов malloc() и free(). Чтобы разрешить формирование таких оберток, в скрипте сборки необходимо указать для линкера ключи -Wl,-wrap=malloc -Wl,-wrap=free, как показано на рисунке 54.

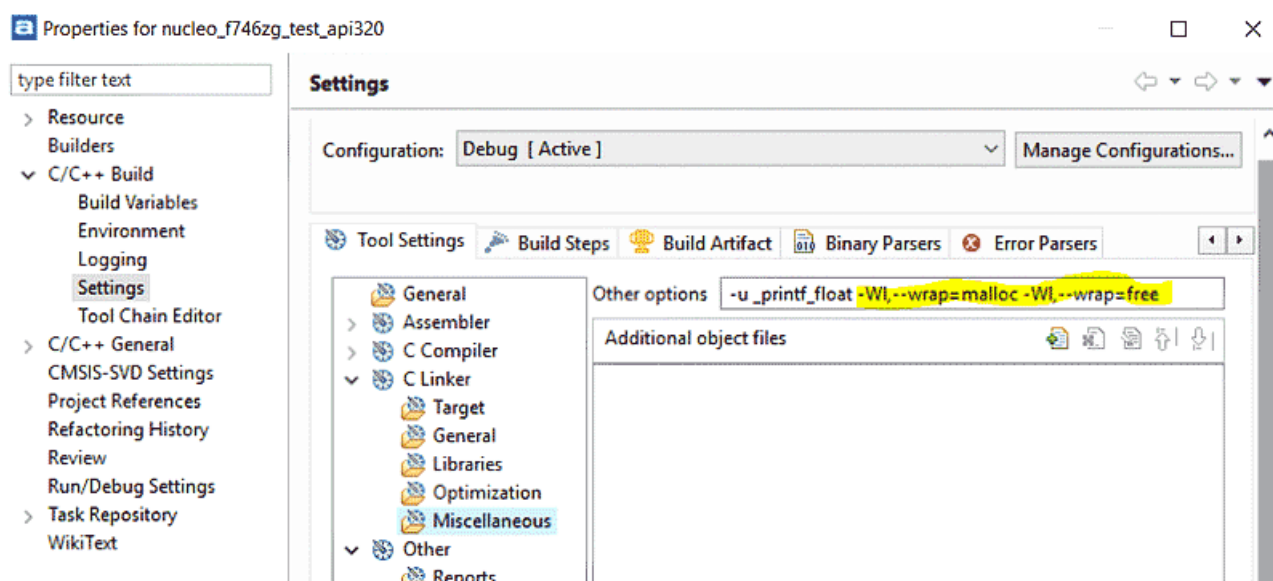


Рис. 54. Опции линкера для включения мониторинга кучи

В проекте на базе GCC при выводе форматированных значений с плавающей запятой печатается пустое место

Для вывода форматированных значений с плавающей запятой необходимо добавить следующую опцию линкера:

-u \_printf\_float

Нужно ли включать или конфигурировать периферийный модуль TIMER?

В этом нет необходимости. Механизм измерения числа тактов ЦПУ, затрачиваемых на формирование логического вывода, использует специальный модуль отладки DWT (модуль отслеживания данных и трассировки), входящий в состав ядра ARM® Cortex®-М и потому имеющийся на всех поддерживаемых устройствах STM32. Этот модуль использует собственный счетчик, работающий от тактового сигнала ЦПУ (системный тактовый сигнал HCLK).

Как в сгенерированном проекте обновить только экспортируемую библиотеку нейронной сети?

При внесении изменений в экспортируемые и генерируемые файлы, следует их обрамлять специальными комментариями(`/* USERCODEBEGIN*/` и `/*...END*/`), которые среда STM32CubeMX обрабатывает особым образом. Для загрузки новой модели нейронной сети и для обновления IDE-проекта следует повторно открыть ранее сохраненный файл проекта `<project_name>.ioc`.

Можно ли экспортировать библиотеку нейронной сети в проект, создаваемый без использования среды STM32CubeMX?

Поскольку все компоненты экспортируемой библиотеки нейронной сети размещаются в отдельной папке, имеющей четко определенную структуру (подробней об этом сказано в разделе «Генерируемая библиотека нейронной сети для STM32»), эту подпапку можно полностью скопировать в дерево исходников любого конечного проекта:

создайте новый фиктивный проект STM32CubeMX для требуемого МК STM32;

Сгенерируйте IDE-проект для желаемого набора инструментальных средств или IDE. Этот шаг требуется для включения в проект правильной библиотеки `network_runtime.a`, которая зависит как от набора инструментальных средств, так и от конкретной архитектуры ARM® Cortex®-М целевого устройства;

скопируйте получившуюся папку в дерево исходников нового проекта;

Добавьте файлы `network.c`, `network_data.c` и библиотеку `network_runtime.a` в сборку проекта и обновите опции компилятора C/C++ и линкера согласно инструкции из раздела «Генерируемая библиотека нейронной сети для STM32»;

Вернитесь к шагу 1 для обновления и проверки модифицированной модели нейронной сети.

Где интерфейс командной строки?

В состав пакета расширения X-CUBE-AI входит многофункциональная утилита командной строки [7].

## Ресурсы и документация

## Использование документации пакета X-CUBE-AI

Чтобы добраться до документации, входящей в состав пакета расширения X-CUBE-AI, выполните следующие действия:

Наведите курсор на один из режимов X-CUBE-AI и нажмите на «details and documentation» (рисунок 55).

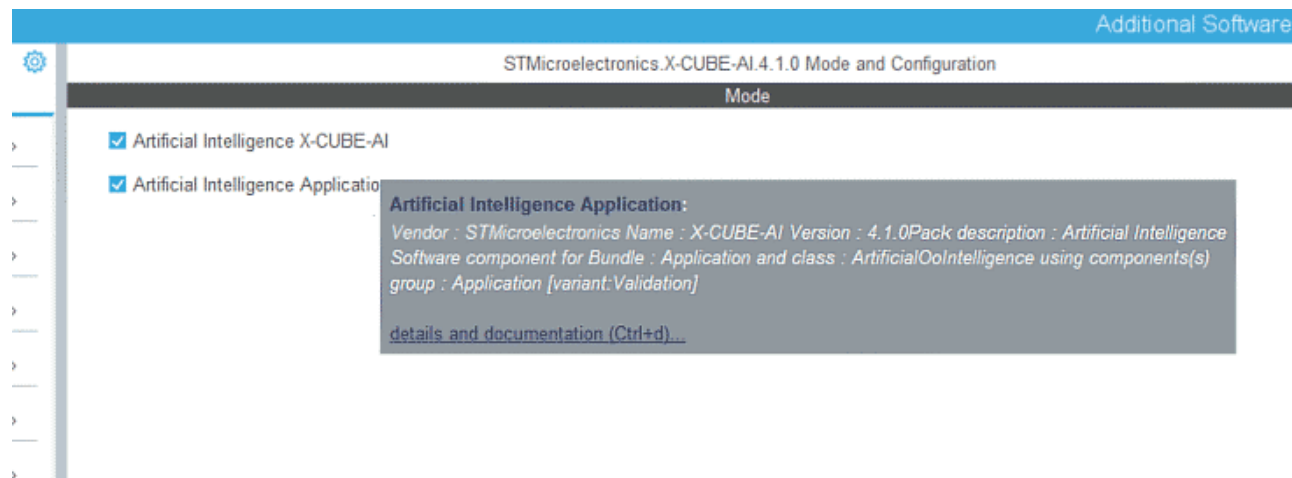


Рис.55. Доступ к внутренней документации пакета (шаг 1)

Нажмите на «SW Pack documentation» (рисунок 56).

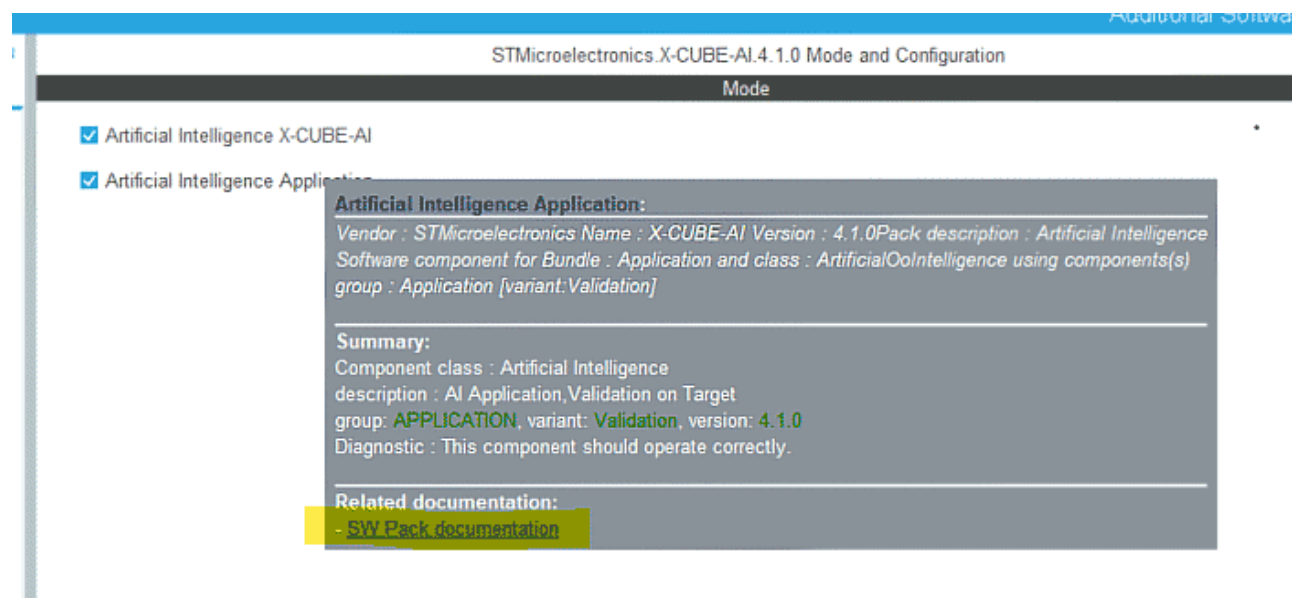


Рис. 56. Доступ к внутренней документации пакета (шаг 2)

В открывшемся окне браузера вы увидите ссылки на доступные файлы документации пакета X-CUBE-AI (рисунок 57).

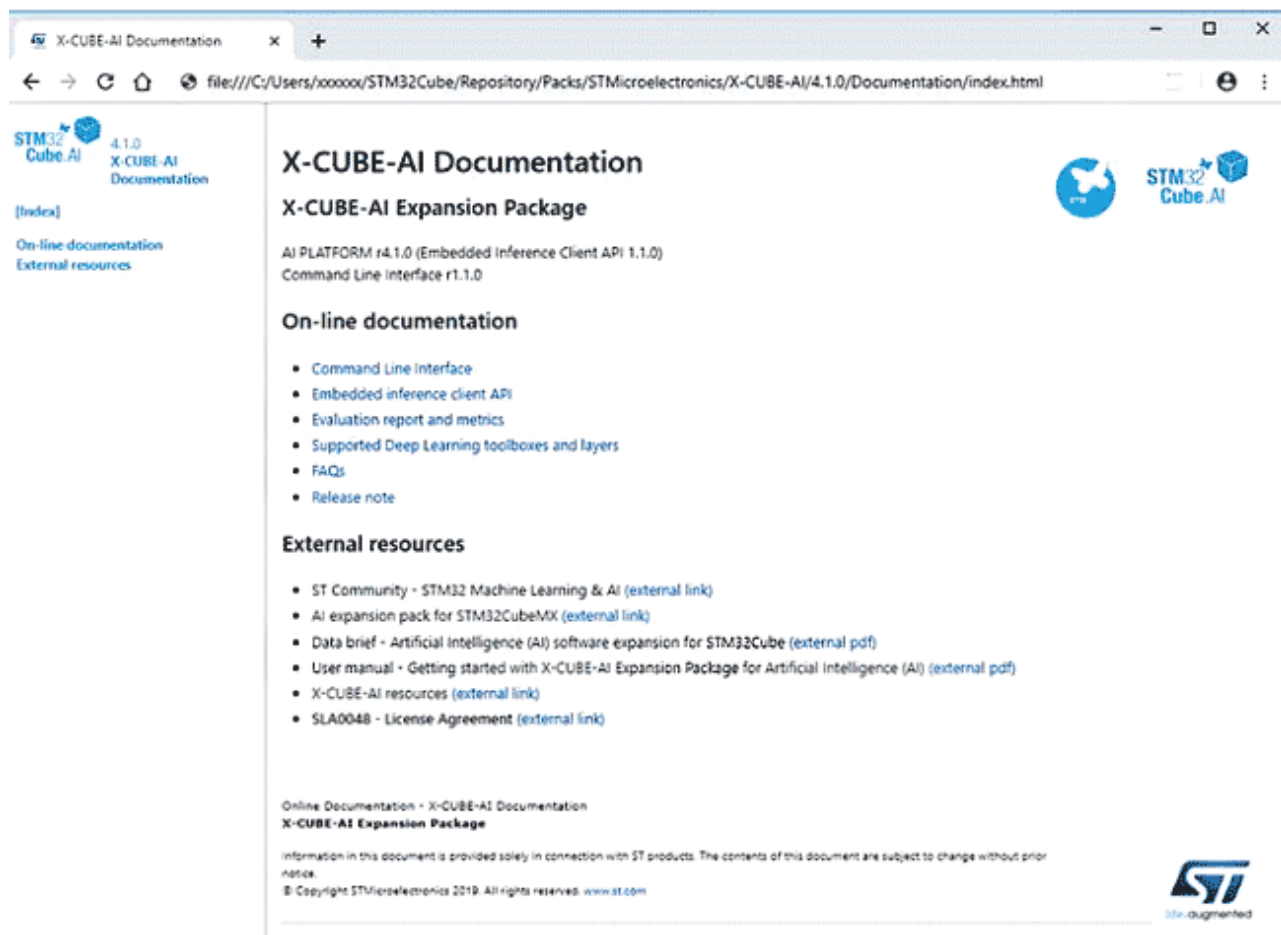


Рис. 57. Стартовая страница документации пакета

Дополнительные материалы:

Источник:

Автор: ST Microelectronics Переводчик: Андрей Евстифеев (г. Королев)

Производители:

Разделы: , ,

Опубликовано: 24.12.2019