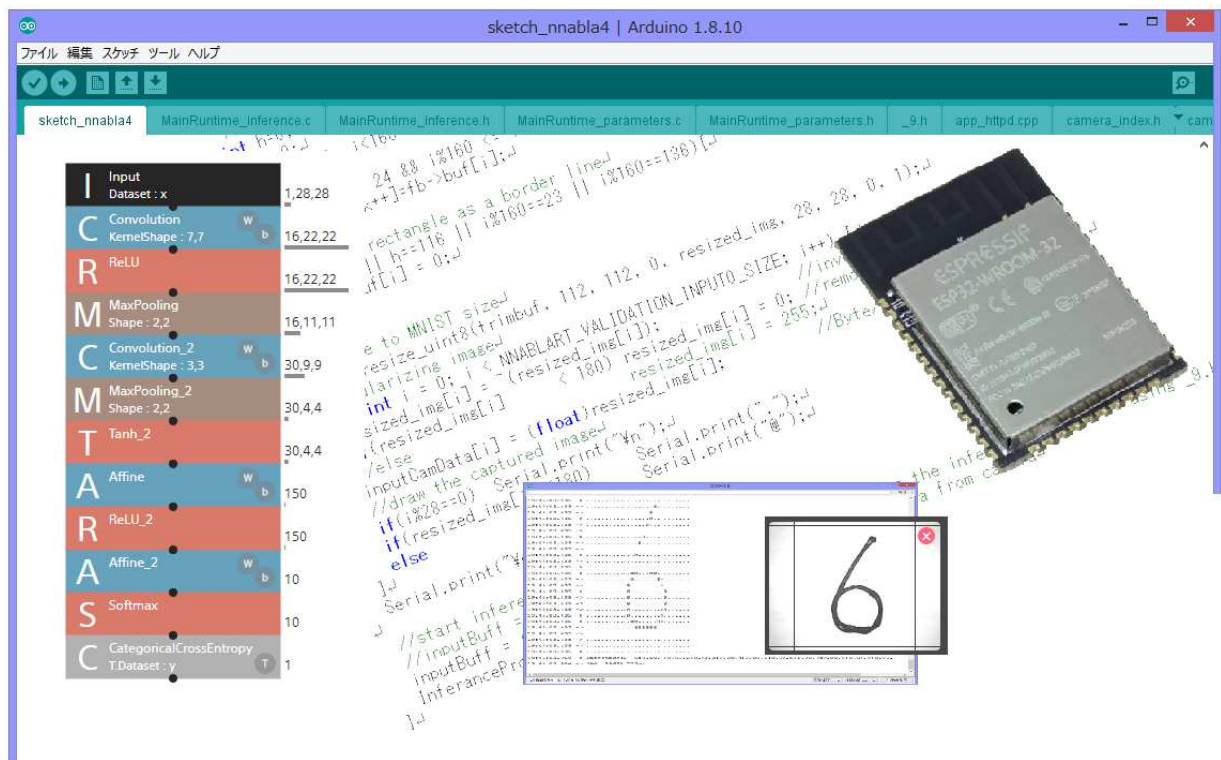


SAMPLE

Implementing Deep Learning with Arduino IDE

How to Implement Deep Learning Inference on ESP32 with Arduino IDE using SONY Neural Network Console.



This textbook and the attached software are licensed to the course participants.

Unauthorized duplication or distribution is prohibited.

To use with more than one person, you need to subscribe the course for the number of people.

info@it-style.jp

*Sorry for your inconvenience, some of the captured images are different from the images on Windows English version.

Firstly

Now you see some projects that require AI technology such as deep learning and/or machine learning under the condition of the decline of the microcomputer device cost. Somehow the dominant language for AI is complicated Python. A lot of information is based on Python. And enormous CPU power is required to build a large neural network, and it is impossible to do this with only a small microcomputer. Of course, some embedded microcomputer systems have large ROM / RAM and run an OS such as Linux, Android, Windows, etc. to use Python. But these large systems are the same as PCs. Since our target is Arduino microcomputers, so we don't put focus on the details regarding Python in this course.

Most of the people who take this course have touched the microcomputer, and those who can do something like blinking a LED and can read the manuals. So, I can provide a guideline to those who are new to deep learning and wondering how to start designing with simple microcomputers. The goal is to understand the process to design an application which equips AI functions in it that fits within a small one-chip microcomputer capability. To train the neural network itself should be done on a PC beforehand, then you can use the inference engine part for a microcomputer. With this technique, various things can be applied even with low power microcomputers. In other words, I will tell you the best shortcut to have microcomputers obtain AI functions, although there is a huge difference between large computers and microcomputers in its capabilities.

By the way, we will target Espressif's ESP32 as a microcomputer, which is rapidly spreading in the IoT field. As an Arduino microcomputer without using the standard development environment for ESP32 named ESP-IDF, we will use Arduino IDE. If you use ESP-IDF, you can use all the detailed libraries, but it requires a great deal of effort to implement the software. On the other hand, in the Arduino environment, the libraries are well categorized and organized up to the awesome level that does not cause any problems in the normal development. So, it is easy to use even for beginners of microcomputers. In addition, many textbooks have been published, and to develop programs with Arduino has become a practical standard. To use Arduino IDE must be the best choice for microcomputer engineers to learn the technology necessary for installing AI functions. You can experience the process without any troubles.

This course is a self-learning course with a textbook that summarizes the three courses of software classes offered by IT-STYLE. The course is about 6 months in the classroom. Since this is not a face-to-face classroom, it is configured to operate as it is as practical as possible, so it can be completed in about 20 hours at minimum.

■ License terms

- This manual is licensed only to the purchaser.
- Originally, this course is a classroom lecture. If you want to share knowledge with multiple people, please purchase the course with multiple people.
- Copyright belongs to IT Style Japan.
- Registered trademarks, such as products and brand names of each company, are not printed to the extent that there is no misunderstanding.

■ Target of this course

- Person who wants to implement AI function by using microcomputer ESP32 in Arduino IDE.
- Person can use the Arduino IDE to make LED blinking operations such as UNO or other microcomputers.
- Person has C language knowledge to some extent.

■ Required environment

- Windows 8.1 or later
- Wi-Fi environment 2.4GHz , own SSID and password required in some examples.
- Arduino development environment (Arduino IDE)
- Anaconda or Python in some examples.
- ESP32 microcomputer board (M5Camera is most suitable. See below)

■ the term

NNC ... SONY Neural Network Console

NNabla ... Sony made deep learning framework “Neural Network Libraries
nnabla (lowercase) in C code

Arduino IDE ... Arduino integrated development environment

ESP-IDF ... ESP32 C ++ library provided by Espressif, source code

Sketch ... Projects and programs created with Arduino IDE

ESP32 ... Indicated as a representative name of microcomputer or mounted module

M5Camera ... PSRAM Camera Module with ESP32 WROVER (OV2640)

■ Working folder

In my(writer's) environment, the folders under C: ¥users¥oichiro are used as working folders. Please replace the ¥username part as appropriate. Yoichiro is my username.

*¥ means \ (backslash in Windows English version) as the delimiter of the file/folfer path.

TOC

Firstly	1
~Prior knowledge~	4
About AI.....	4
About ESP32	6
~Experience Deep Learning~	8
Sign in NNC	8
Watch NNC videos	10
The first Image Recognition with NNC	12
~Preparation~	14
Install Arduino IDE and ESP32 library	14
Install Python (Anaconda)	17
Install NNabla	19
~actual practice~	21
Get the result out of NNC.....	21
Prepare NNabla CRuntime	25
Fist sketch	28
Merge with NNC code	31
MNISTdata set	33
Execute inference with Sketch.....	34
~application~	38
MNIST on NNC.....	38
Build Camera sample.....	41
Bind cam sketch and neural network	43
Preprocessor.....	46
Lastly.....	47
~bonus~	48
Integration of CRuntimeLibrary.....	48
MNIST dataset	50
Q&A	53

~Prior knowledge~

About AI

Many books have been published and many paid seminars have been held. I think that it takes a lot of effort to get to know the details. When you complete this course, you will see what is required, what is not required and also it will be possible to see what the microcomputer software/hardware engineers should do in designing of systems and what part AI specialists should do.

When you are testing AI code only on your PC, you tend to enter into academic territory too deeply and spend days worrying in agony. In the devices equipped so-called microcomputers, high performance and complex AI functions on high-end computers are not required. Therefore, it is possible to say that AI is installed in the systems for the time being as long as it is equipped with an inference engine that consists of a neural network. And this simple implementation would be the first step for the future complicated deep learning system.

Lately, I have heard that some engineers are worried if they can say that their systems are based on AI technology, when just IF statements are in the code. Probably, their code can't be called AI systems. So, I opened a microcomputer deep learning course with Arduino IDE.

Honestly, I'm not an AI expert. I've been involved in microcomputer software for over 30 years, so I think my explanations is biased a little toward the embedded microcomputer software engineer's point of view. It may include some omission and/or trick to make explanation easy. But it is no problem when you just grab the whole flow of the process.

By the way, AI is generally called machine learning or deep learning, but as the simplest understanding, deep learning is the most complicated and cutting-edge research field, and its basis is called machine learning. Machine learning has existed for decades, and we have recognized that many researchers have been experimenting with it. I think that the increase in computer power made it possible to carry out a large amount of vector calculations all at once, and the deep learning boom of today has occurred.

A typical example is Google's TensorFlow, which is probably the most famous one. And some others are provided by Amazon, Microsoft, IBM etc. In many cases, APIs are provided as Python libraries. However, the APIs are not unified, and the names and functions are different. As for TensorFlow and wrapper Keras, there is a lot of information on the net, and also many books published for starters in earnest to learn it. So, if you are involved in company operation or have a solid budget and support, you should start with TensorFlow.

In this course, we will use Neural Network Console (NNC) provided by SONY this time.

The reason is immediately apparent when you use it, but anyway, the user interface is easy for

beginners and the integrated environment makes it possible to experience deep learning without understanding difficult technical terms.

The biggest point is that this SONY NNC makes the inference engine publicly available under the Apache license. Moreover, it supports not only Python but also C++ and C language.

C++ is based on the use of the file system, but the runtime library of C language can handle C code almost like ordinary ROM-based software, so it feels very easy to handle for microcomputer software engineers.

Support is also extensive with online manuals and video, and many samples are on the web.

If you reached a certain level and got tired of this NNC, you can challenge advanced things with other tools. So, such advanced level is not within the scope of this course though.

On the contrary, you will be charged if you exceed the trial time. However, the fee is set very cheap if you don't use GPU. So, you don't have to worry about it. Besides, Windows version is offered for free to use permanently.

After reading the above explanation including the author's impression, please do research on the information of various people on the Internet. There is a fair amount of information you can get.

About ESP32

Next, I will write a little about ESP32 used as a target microcomputer in this course. If you are familiar with ESP32, skip this chapter.

The word IoT has been around for several years. The world is changing as everything started to be connected to the Internet, but 20 years ago, the same idea already existed as Ubiquitous or Home Automation.

However, at the time the price of devices, especially wireless chips and microcomputers, was high, and even if they were implemented, it seemed like the boom has passed because of the too small user's merit. The, the price of these types of microcomputers has fallen steadily these days. And it has fallen to a certain level at which they can be mounted in household products. ESP32 is a standing out microcomputer module that can be obtained individually and at very low cost.

The microcomputer itself has a model number called ESP32-WROOM-32, but each board maker sells it as a module that equips a USB port, power supply regulator, etc. and it has become easy to use for experiments.



ESP32 Developer Board



ESP32-DevKitC

It is difficult to put a lot of commentary on paper, so please do an Internet research. There is a lot of information about ESP32.

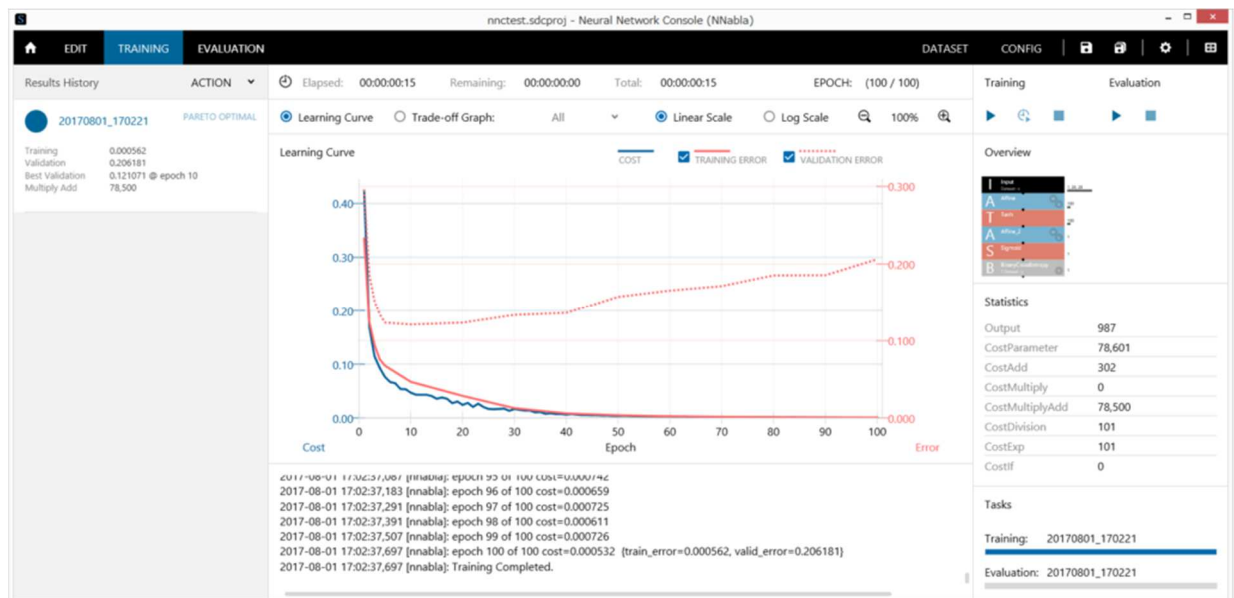
* ESP32-CAM, a low-priced model, can be used in this course, but it is not recommended for anyone who is not familiar with handling ESP32.

~Experience Deep Learning~

Sign in NNC

For the time being, as a experience deep learning, Let's get a grasp of the development flow first. There are criticisms that the definition of deep learning is a little different, but in this course, we will call the expression about artificial intelligence such as machine learning and deep learning as deep learning for now.

Now, as I mentioned in the introduction to AI, this course uses the Sony Neural Network Console as a deep learning platform. Compared with the famous Google TensorFlow etc.,

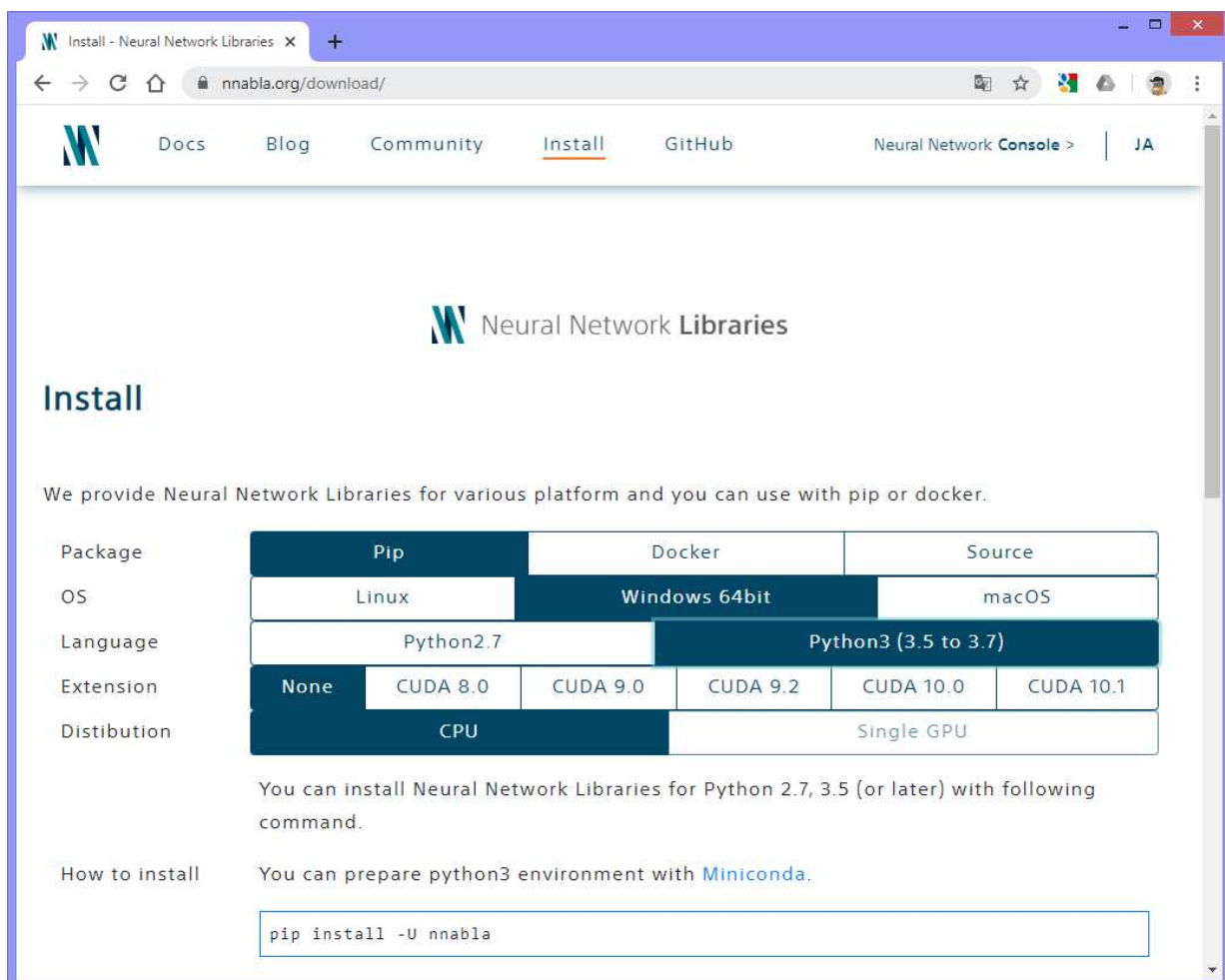


Install NNabla

NNabla is an NNC library module. It works with Python, so install it with the pip command.

<https://nnabla.org/download/>

Open the site and follow the instruction.



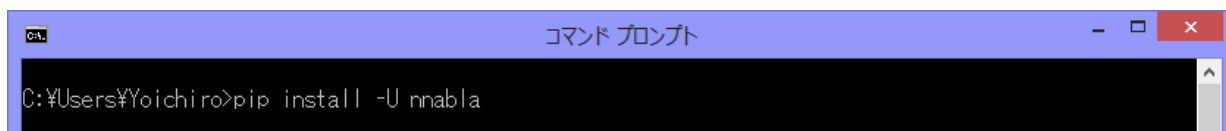
The screenshot shows the NNabla website's installation page. The page title is "Install - Neural Network Libraries" and the URL is "nnabla.org/download/". The navigation menu includes "Docs", "Blog", "Community", "Install", and "GitHub". The main heading is "Install". Below the heading, there is a table with the following structure:

Package	Pip	Docker			Source
OS	Linux	Windows 64bit			macOS
Language	Python2.7		Python3 (3.5 to 3.7)		
Extension	None	CUDA 8.0	CUDA 9.0	CUDA 9.2	CUDA 10.0 CUDA 10.1
Distribution	CPU			Single GPU	

Below the table, there is a text block: "You can install Neural Network Libraries for Python 2.7, 3.5 (or later) with following command." and a "How to install" section with the command: "You can prepare python3 environment with [Miniconda](#)." and a code box containing: "pip install -U nnabla".

Select Package : pip、 OS : Windows64、 Language : Python3.Distribution : CPU

Then type like the below.



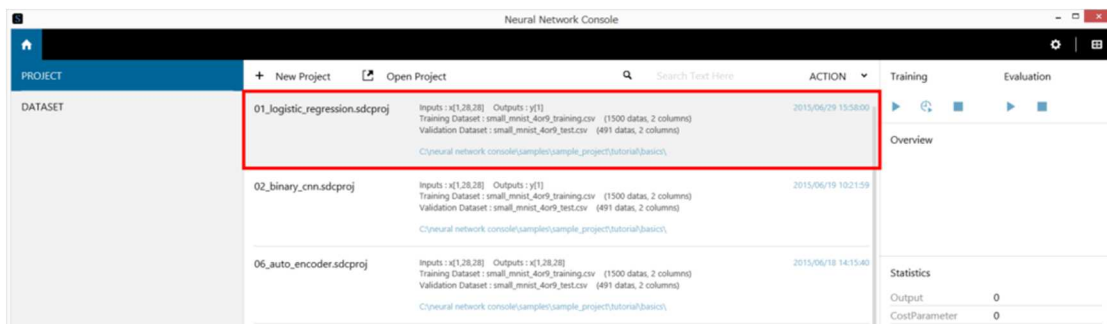
The screenshot shows a Windows command prompt window titled "コマンド プロンプト". The command entered is "C:\Users\Yoichiro>pip install -U nnabla".

After the installation is

~actual practice~

Get the result out of NNC

We will use the trained neural network of the 1-layer Logistic Regression that can judge the numbers 4 and 9 that we experienced at the beginning of



When the TRAINING and EVALUATION execution is completed, the parameters of this learning result and the configuration of the neural network are acquired

MainRuntime_example.c is an inference sample program that uses this neural network configuration and parameters. If you can read

```
int nnablart_mainruntime_inference(void* context) {
    nnablart_mainruntime_local_context_t* c = (nnablart_mainruntime_local_context_t*)context;
    exec_affine(&(c->f0));
    exec_sigmoid(&(c->f1));
    return NN_ERROR_CODE_NOERROR;
}
```

Prepare NNabla CRuntime

As the first goal, we will use Arduino IDE to make an application instead of the development environment of ESP-IDF which has high level difficulty.

To do this, you need NNabla C Runtime Library for Arduino IDE.

First, open the option specification file for the gcc compiler used by Arduino.

First sketch

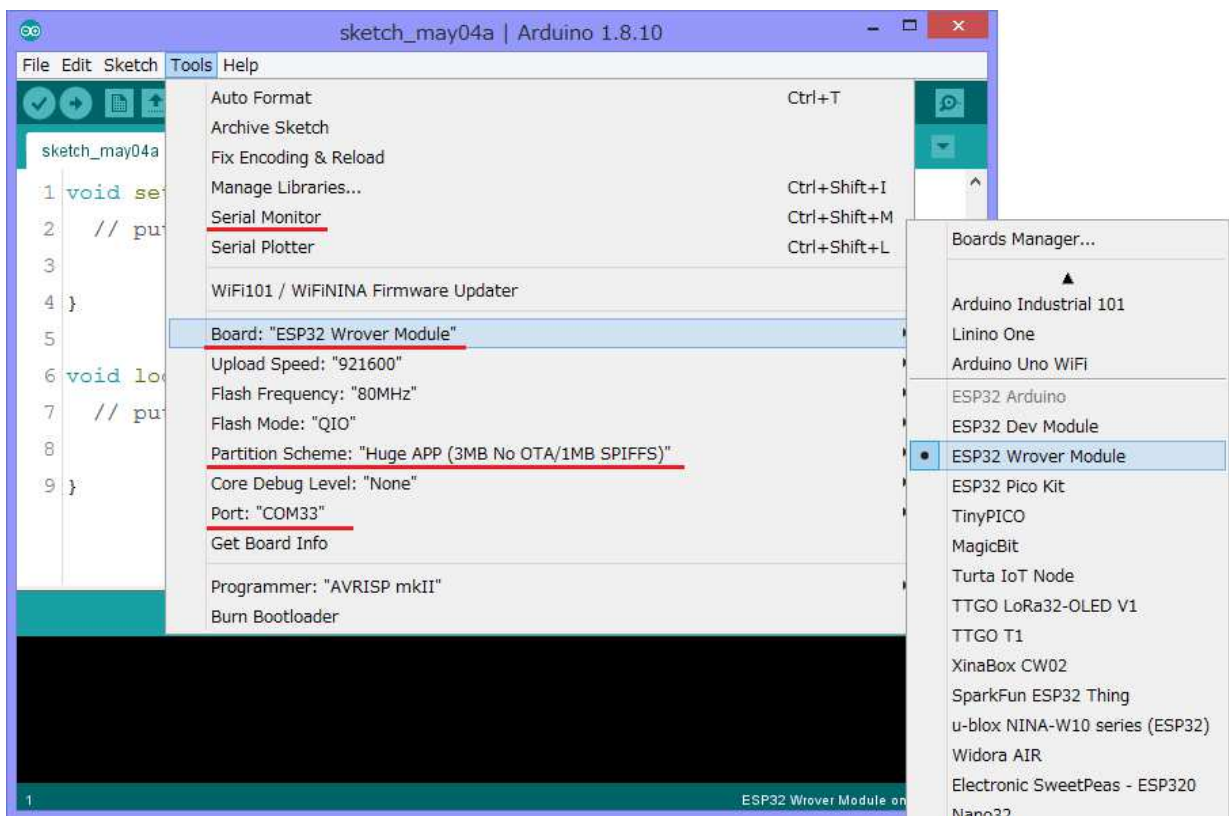
Now that you're ready to create the Arduino sketch (program), let's create the program.

'Hello world', I think that it is the most common first program for beginners. In the embedded system, Blinking LED is the replacement. Unfortunately, M5Camera does not have an LED, so Make a Hello World program using serial communication instead. And then link NNabla to it.

Create a new project

Start Arduino IDE and click [File]-[New File] to open a new file.

As mentioned in the Arduino installation, set the ESP32 environment from the [Tools]



Now simply add 2 lines so that "Hello NNabla." is displayed on the serial monitor.

Merge with NNC code

Let's extend sketch_nnabla1.ino.

Copy the C source files extracted in the "Get the result out of NNC" chapter to the sketch_nnabla1 folder.

Copy the functions.h network.h to

```
1 // Copyright (c) 2017 Sony Corporation. All Rights Reserved.↵
  // ↵
  // Licensed under the Apache License, Version 2.0 (the "License");↵
  // you may not use this file except in compliance with the License.↵
- // You may obtain a copy of the License at↵
  // ↵
  // http://www.apache.org/licenses/LICENSE-2.0↵
  // ↵
  // Unless required by applicable law or agreed to in writing, software↵
10 // distributed under the License is distributed on an "AS IS" BASIS,↵
  // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.↵
  // See the License for the specific language governing permissions and↵
  // limitations under the License.↵
  ↵
- // *WARNING*↵
  // THIS FILE IS AUTO-GENERATED BY CODE GENERATOR.↵
  // PLEASE DO NOT EDIT THIS FILE BY HAND!↵
  ↵
  #include "MainRuntime_inference.h"↵
20 ↵
  #include "network.h"↵
  #include "functions.h"↵
  ↵
  #include <string.h>↵
- ↵
```

MNISTdata set

We confirmed that the output file from NNC can be converted into C source files in the Arduino environment, and can be built by merging it with the user application (currently just simply displays 'Hello NNabla').

Using the actually trained neural network and parameters, we will create a program that makes an inference with the ESP32 microcomputer.

In order to step forward stably and to get the overall understanding, we initially set the input data fixed. Specifically, we use an array in C from the handwritten numeral data set called MNIST.

In this course, some converted files as C sources from grayscale 28 * 28 pixel images of 4 and 9 characters in the MNIST dataset are prepared in advance.

Although there are only a few sample files prepared, the conversion method is described at the end of the book as a reference for creating your own or your future applications.



_4.h



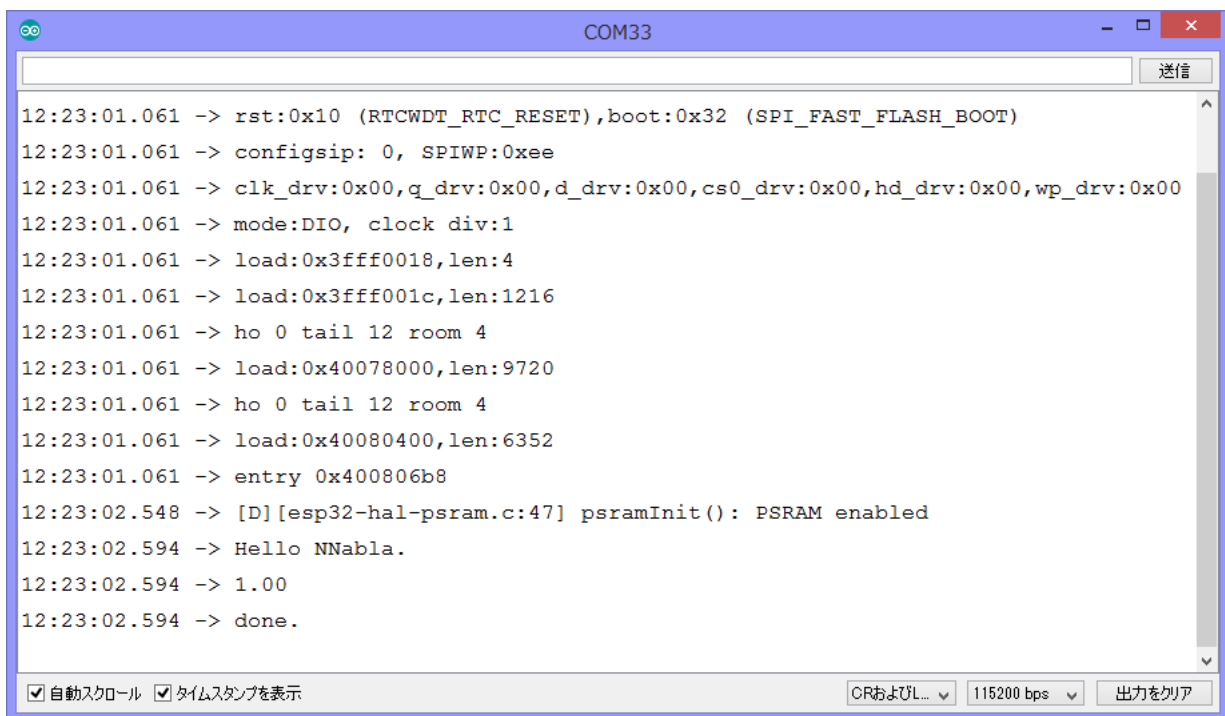
_9.h

The files which converted into array data in C from 0 to 9 are

Execute inference with Sketch

Copy sketch_nnabla1 folder as it is and rename it to sketch_nnabla2. Please rename the sketch_nnabla1.ino inside the folder to sketch_nnabla2.ino as well.

Then, copy the files _4.h and _9.h from



```
COM33
12:23:01.061 -> rst:0x10 (RTCWDT_RTC_RESET),boot:0x32 (SPI_FAST_FLASH_BOOT)
12:23:01.061 -> configsip: 0, SPIWP:0xee
12:23:01.061 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
12:23:01.061 -> mode:DIO, clock div:1
12:23:01.061 -> load:0x3fff0018,len:4
12:23:01.061 -> load:0x3fff001c,len:1216
12:23:01.061 -> ho 0 tail 12 room 4
12:23:01.061 -> load:0x40078000,len:9720
12:23:01.061 -> ho 0 tail 12 room 4
12:23:01.061 -> load:0x40080400,len:6352
12:23:01.061 -> entry 0x400806b8
12:23:02.548 -> [D][esp32-hal-psram.c:47] psramInit(): PSRAM enabled
12:23:02.594 -> Hello NNabla.
12:23:02.594 -> 1.00
12:23:02.594 -> done.
```

自動スクロール タイムスタンプを表示 CRおよびL... 115200 bps 出力をクリア

～application～

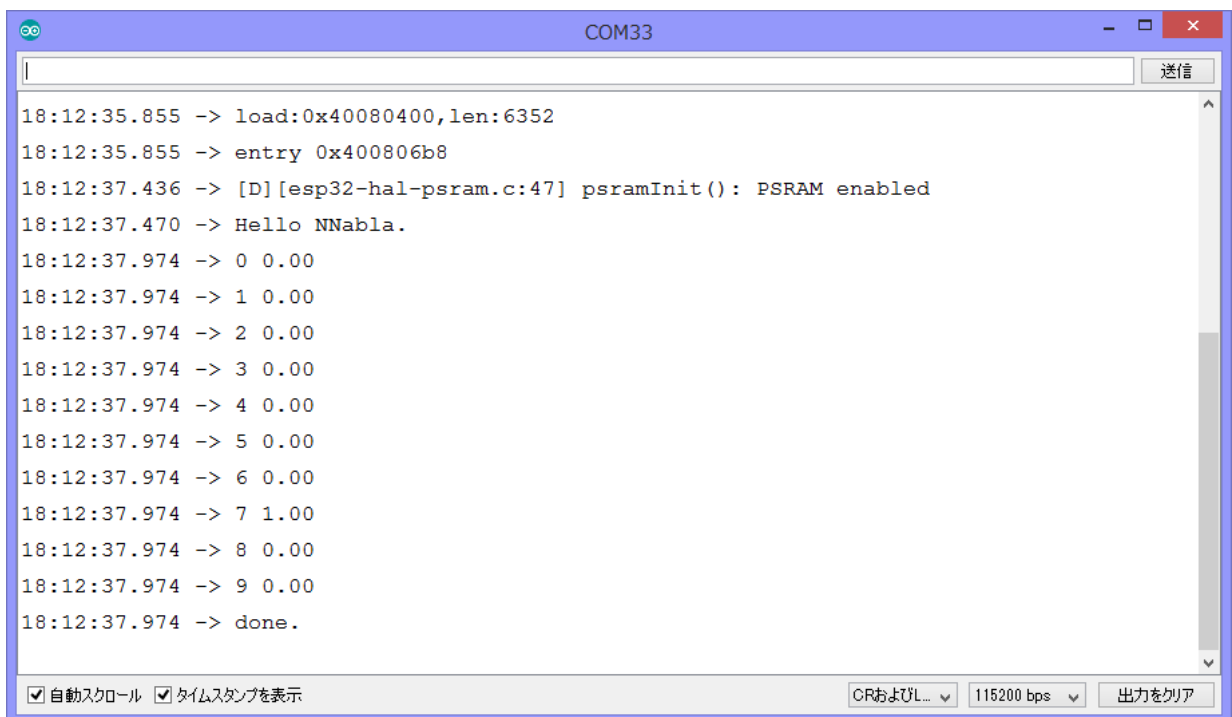
In the course so far, we handled the ESP32 microcomputer on the Arduino IDE, output the neural network on NNC and got C source files, and linked them with the NNabla C Runtime library to make an inference engine that can execute inference from unknown input image.

Here, we will extend it further and create an application. The function is that the system can continuously make an inference from 0-9 handwritten numbers captured by a video camera.

MNIST on NNC

First, using NNC and MNIST data set, we will create a neural network for judging handwritten digits from 0 to 9.

The figure shows the result when `_7.h` is specified. The program can correctly infer that the probability of 7 is 1.00.



```
18:12:35.855 -> load:0x40080400,len:6352
18:12:35.855 -> entry 0x400806b8
18:12:37.436 -> [D][esp32-hal-psram.c:47] psramInit(): PSRAM enabled
18:12:37.470 -> Hello NNabla.
18:12:37.974 -> 0 0.00
18:12:37.974 -> 1 0.00
18:12:37.974 -> 2 0.00
18:12:37.974 -> 3 0.00
18:12:37.974 -> 4 0.00
18:12:37.974 -> 5 0.00
18:12:37.974 -> 6 0.00
18:12:37.974 -> 7 1.00
18:12:37.974 -> 8 0.00
18:12:37.974 -> 9 0.00
18:12:37.974 -> done.
```

Build Camera sample

Next, we will handle the input from the camera.

From Arduino IDE, select [File]-[Sketch example]-[ESP32]-[Camera]-[CameraWebServer].

In this course, the implementation from

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
- //#define CAMERA_MODEL_M5STACK_PSRAM
#define CAMERA_MODEL_M5STACK_WIDE //#####nnabla
//#define CAMERA_MODEL_AI_THINKER
↓
#include "camera_pins.h"
20 ↓
const char* ssid = "atern-"; //#####nnabla
const char* password = "1224ea"; //#####nnabla
↓
```

If the build finished successfully, the IP address to access this camera will be displayed on the serial monitor. Please type the URL on your browser to connect.

```
19:30:52.916 -> WiFi connected
19:30:52.916 -> Starting web server on port: '80'
19:30:52.916 -> Starting stream server on port: '81'
19:30:52.916 -> Camera Ready! Use 'http://192.168.0.8' to connect
```

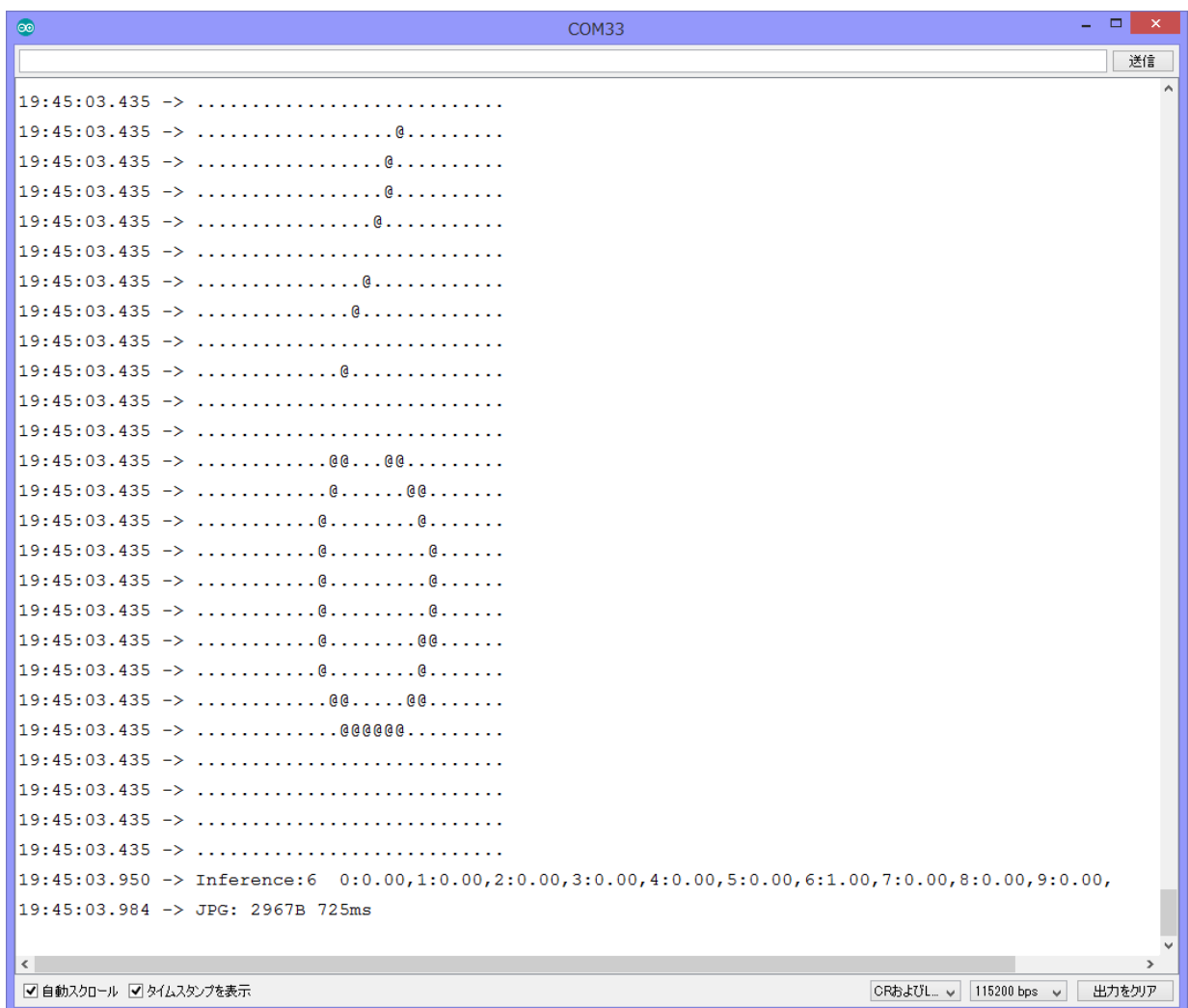
Bind cam sketch and neural network

The C source file from NNC, a sample sketch of the camera and several files are combined into one application program. Please refer preprocessing in the next chapter.

sketch_nnabla4 folder contains the sketch.

The build should succeed with the above modification.

Shoot handwritten characters with the camera and check if the inference can be done correctly.



The serial monitor displays the binary image of the shooting character and the inference result of each number. Since Softmax is used at the end of the neural network, the output result is shown as the total of the probabilities of 10 numbers is 1.00.

Preprocessor

The image data from the camera is stored in the frame buffer.

```
camera_fb_t * fb;
```

Since FORMAT is specified as PIXFORMAT_GRAYSCALE, it is a binary array of 8-bit brightness. Now we need to think about how to fit this raw data into the MNIST image data

This is so called pre-process that occurred because we used the MNIST dataset this time, but if all the training data were collected by ourselves rather than using pre-made dataset like MNIST dataset, in the first place, a lot of problems would occur in the processing of the original data. In a real development project, creating a training data set under the uniform standard can be a daunting task.

Lastly

This is the end of the course.

I've explained the deep learning course with the Arduino environment. I think you have almost grasped the big picture.

Through the course, you saw something important when you dealt with AI implementation along the general flow. It is likely that application engineers who try to incorporate AI technology will have to put the most effort into creating the deep learning training dataset. It is also important to design the preprocessing part properly for dealing with the raw data in the inference part.

AI neural network itself will be able to be built with good platforms such as NNC, which will be provided by leading-edge AI engineers in the future.

As for AI, there is no answer that this is the correct answer. In simple terms, research is being done to find the solution where the inference result is the best and to find know-how regarding neural networks.

Meanwhile, it is very difficult for an individual engineer to cover deep expert skills in many fields. Therefore, the designs and performance of the application systems will depend on how many experiences the engineer has.

I hope that this course will help you a little.

筆： むらよ ITスタイル代表 村上陽一郎
by Yoichiro Murakami IT-Style.Japan

~bonus~

Integration of CRuntimeLibrary

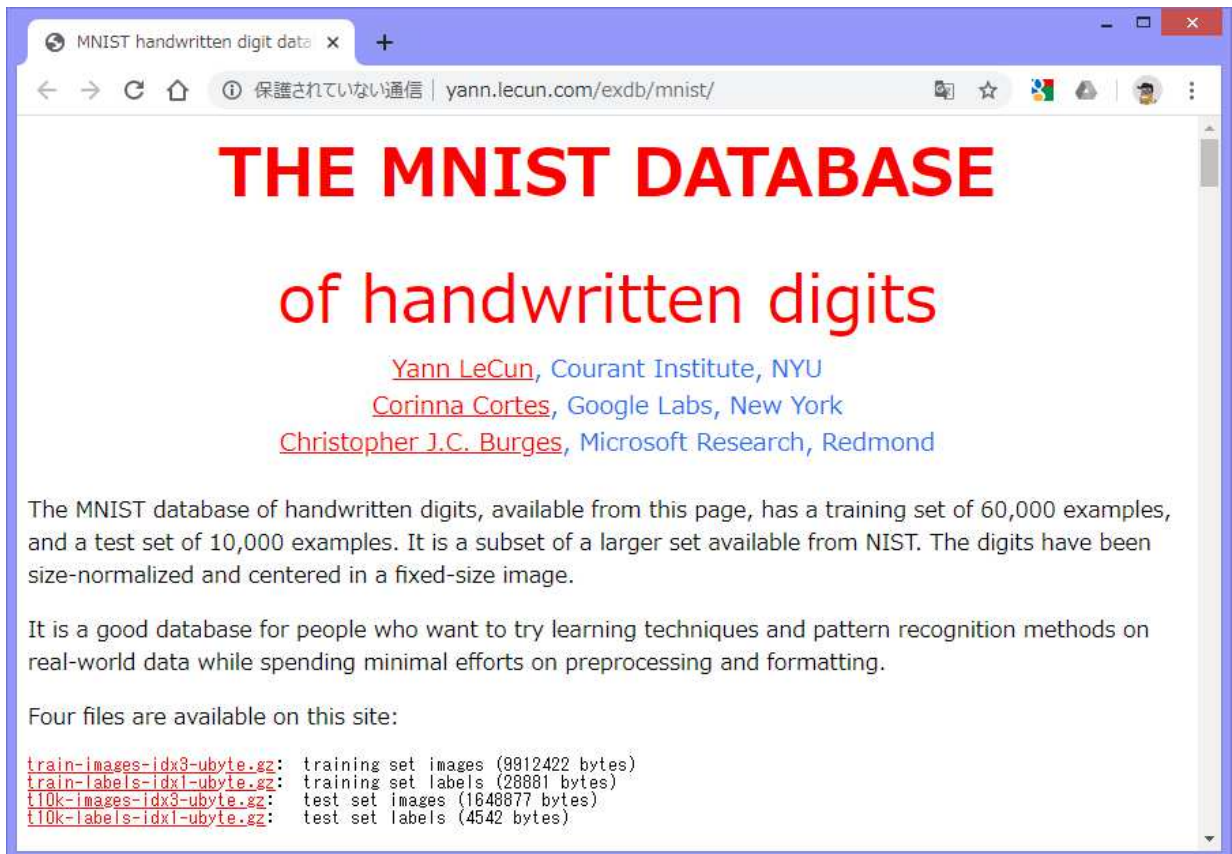
NNabla has a C language library. The source code is released on GitHub,

MNIST dataset

MNIST dataset is used in many AI experimental sample implementations.

<http://yann.lecun.com/exdb/mnist/>

You can download the dataset here, but all of the datasets are binary data. So, conversion is necessary.



Processing takes a time.

If it successfully finished, you see generate two folders, [training] and [validation]. And inside of each folder, there are 0 to 9 folders and 60,000 png image files respectively.

neural_network_console_171 > samples > sample_dataset > MNIST > training > 4

4の検索

neural_network_console_171						
libs	2.png	9.png	20.png	26.png	53.png	58.png
samples						
sample_dataset	60.png	61.png	64.png	89.png	92.png	115.png
CIFAR10						
CIFAR100	127.png	131.png	139.png	142.png	150.png	163.png
FashionMNIST						
iris_flower_dataset	164.png	166.png	194.png	217.png	222.png	237.png
MNIST						
training	257.png	271.png	272.png	275.png	289.png	292.png
0						
1	294.png	297.png	314.png	329.png	336.png	338.png
2						
3						
4						
5						

Total 53pages