Robin Bird™ Machine Learning Library

INTERNAL USE ONLY © 2019 NXP Semiconductor, Inc. All rights reserved.

Version 1.0, September 25, 2019

Table of Contents

Chapter 1. INTRODUCTION

Robin Bird™ is a small *Machine Learning* library, fully experimental, written in *C++* for learning purpose only. This library provides simple implementations of *Machine Learning* algorithms for solving real world problems on embedded systems. These algorithms are released as simple command-line programs and classes which can be used and integrated into low to medium-scale *Machine Learning* solutions.

1.1. LICENSE

Robin Bird™ is licensed under MIT License *Copyright 2019 NXP Semiconductors*.

1.2. Artificial Intelligence and Machine Learning

Artificial Intelligence (*AI*) brings to mind science fiction stories about robots and talking machines. Some might even relate *Machine Learning* (*ML*) to advanced robots bent on destroying the world. Today this technology is more about building intelligent systems with decision making abilities. Think about *ML* as a sub-field of *AI* that can analyze large amounts of data and automate analytical models. This branch of AI was born from pattern recognition and the theory that computer can learn, and having the ability to independently draw knowledge from experience.

The Deep Blue Computer could play *chess* in a high level championship but that's all it could do. Presently, large data centers and huge storage capacities make things that were believed for years to be distant concepts possible. Writing a program to play tic-tac-toe would require a huge number of specific conditional branches to handle even these simple moves and rules. The problem grows significantly with more complicated games such as chess that has an average of *35* possible moves each turn.

In *chess* these choices become ever more complex as the game progresses and rules written in a structured programming language begin to break down. The program and its decisions for every possibility become impracticable to manage. The rules do not need to be written by a human. With the right instructions, the computer can learn to create its own decision trees.

Machine Learning combined with hardware-accelerated embedded systems can reach significant improvements in image and video recognition. The reason is a certain level of performance embedded systems have reached over the last few years. This explains how hardware-accelerated embedded systems have been gaining in several types of industrial processes. Autonomous vehicles and voice control assistants — once subjects of science fiction — have attained significant research and development success and even consumer products in today's marketplace.

This field boils down on spending time to write many lines of code that eventually solve a problem by applying some type of intelligent algorithm. For instance, some smart houses have lighting systems which automatically turn on and off based on whether anyone is present in the room. This idea does not amuse, but thinking about this consider that this system is actually making independent decisions.

1.3. Robin Bird Algorithm

The much-needed algorithms required for real-time image and video recognition are being developed and will get it all quickly. Not just that, several types of algorithms for different issues are being created every day and embedded systems are fully prepared to completely embrace this technology replacing the work of an human being. *ML* is the study of algorithms that learn from examples and experiences instead of relying on hard-coded rules. *Robin Bird™* solves real world problems using only math and object-oriented programming, including an easily extensible implementation of object-oriented classes for enthusiast developers.

For instance, think about the following situation where it is required to write code to recognize two types of fruit or cars or animals. This is difficult without *ML*. *Robin Bird™* can make predictions based on a huge amount of data just by recognizing aspects and relations of patterns. Problems like image, speech and character recognition belongs to a category that is called *Classification* problems, which a certain given input, the machine should be able to select a category where it belongs and labeled. *Robin Bird™* include algorithms for recognition problems, it can be integrated with *ML* solutions from low to medium scale.

Chapter 2. GETTING STARTED

For helping the *Robin Bird™* contributing to the library follow the instructions in the Building RB on GNU/Linux Host PC section, otherwise for only running the library sample applications on any *i.MX board* use the instructions in the Building RB on eIQ™ section.

2.1. Building RB on GNU/Linux Host PC

The next steps were tested under the *GNU/Linux Ubuntu* distribution *18.04.03 LTS*.

2.1.1. Prerequisites

Follow a list of the required packages/libraries for running *Robin Bird™* on a regular *GNU/Linux Host PC*:

gtkmm	*C++ interfaces for GTK+ and GNOME*
libpng	*C functions for handling PNG images*

This list of dependencies may increase over time.

2.1.2. Installing Dependencies

1. On *x86* platform, run the following command-lines:

   Host GNU/Linux PC Terminal

   ```
   $ apt-get install libgtkmm-3.0-dev libpng++-dev ①
   ```

   ① using `apt-get` package manager to install dependencies.

   You need root privileges (*sudo*) to use `apt-get` package manager.

2.1.3. RB Source Code

1. Retrieve *Robin Bird™* source code using the following command-lines:

   Host GNU/Linux PC Terminal

   ```
   $ git clone <link> ①
   $ cd robin-bird/ ②
   ```

   ① cloning the repo.

   ② entering the project directory.

2. Checkout the *robin-bird-v1.0* branch as follows:

   Host GNU/Linux PC Terminal

   ```
   $ git checkout robin-bird-v1.0 ①
   ```

   ① checking out the right branch project.

3. Create a build project directory to compile the library as follows:

   Host GNU/Linux PC Terminal

 © 2019 NXP Semiconductor, Inc. All rights reserved

```
$ mkdir build ①
$ cd build ②
```

① creating a build directory.

② entering the build directory.

4. Use the following flags for including *fuzzing*, *samples* and the *lib* to the *Unix Makefile* generated by *cmake* tool:

ROBIN_BIRD	*enables Robin Bird™ library*
SAMPLES	*enables Robin Bird™ samples*
INSTURMENT_FOR_FUZZING	*enables fuzzing tests*
ROBIN_BIN_INSTALL_DIR	*defines where the samples are installed*

5. For a *regular* installation, use the following example:

   Host GNU/Linux PC Terminal

```
$ cmake .. -DROBIN_BIRD=ON \
           -DSAMPLES=ON \
           -DINSTURMENT_FOR_FUZZING=ON \
           -DROBIN_BIN_INSTALL_DIR=/bin ①
$ make ②
```

① configuring the project using cmake and the choosen flags.

② compiling the project.

6. Install the *binaries* and dynamic link the library:

   Host GNU/Linux PC Terminal

```
$ make install ①
$ ldconfig ②
```

① installing the project.

② configuring the library.

You need root privileges (sudo) to install and configure the library.

7. Use *pkg-config* helper tool to check if *Robin Bird™* is correctly installed:

   Host GNU/Linux PC Terminal

```
$ pkg-config robin --libs --cflags
```

The *pkg-config* retrieves the following output:

```
-L/usr/local/lib -lrobin
```

8. If all goes well, go the Robin Bird™ Applications section.

2.2. Building RB on eIQ™

Follow the next instructions to build *Robin Bird™* using *Yocto Project*:

1. Generate an *eIQ™ Machine Learning Software* image for any target board.
2. Then, download this recipe and put inside your *build* folder project.
3. Add the following lines into the *local.conf* file:

   Host GNU/Linux PC Terminal

   ```
   CORE_EXTRA_IMAGE =+ "robin bird"
   ```

4. Run the following command:

   Host GNU/Linux PC Terminal

   ```
   $ bitbake robinbird
   ```

5. Flash the image into the *SD Card*:

   Host GNU/Linux PC Terminal

   ```
   $ bunzip /tmp/deploy/image/robinbird.sdcard.bzip2
   $ dd if=robinbird.sdcard of=/dev/sd<x> status=progress bs=1M && sync
   ```

   You need root privileges (*sudo*) to use `dd` for writing to `/dev/sd<x>`.

   <***x***> refers to the SD Card device. You can check it using `lsblk` command.

 © 2019 NXP Semiconductor, Inc. All rights reserved

Chapter 3. Robin Bird™ Applications

3.1. RB Hello World

Follow the next instructions to test if the library is working properly:

1. Create a simple *C++* and copy the following content:

   Host GNU/Linux PC Terminal

```
#include <iostream>
#include <time.h>

#include <robin/robin-nn.h>

static double random(double x);

int main(int argc, char *argv[])
{
    if (argc < 2) {
        std::cout << "Missing matrix size" << std::endl;
        return 0;
    }

    double size = atof(argv[1]);
    srand(time(NULL));
    Matrix <double> A(size, size), B(size, size);
    A = A.applyFunction(random);
    B = B.applyFunction(random);

    std::cout << "Matrix A:\n" << A << std::endl;
    std::cout << "Matrix B:\n" << B << std::endl;
    std::cout << "Matrix A + B:\n" << A + B << std::endl;
    std::cout << "Matrix A * B:\n" << A.dot(B) << std::endl;
    std::cout << "Matrix A(1,1): "<< A(0,0) << std::endl;
    std::cout << "Matrix A(" << size << "," << size << "): "
              << A(size - 1, size -1) << std::endl;

    return 0;
}

static double random(double x)
{
    return (double)(rand() % 10000 + 1)/10000;
}
```

2. To compile the code, use the next command-line:

   Host GNU/Linux PC Terminal

```
$ g++ sample.cpp `pkg-config robin --libs --cflags` ①
$ ./a.out ②
```

① compiling the code.

② running the generated binary.

This example demonstrates the basic *RB* operations on matrices.

Chapter 4. RB HandWritten Digit Application

Robin Bird™ has a *Neural Network* implemented for *handwritten digits recognition* and instead of explaining the code itself, the document focuses more on the how to get started with this application. The steps consist on:

- *Collecting the data* RB Webpage for Data Collect;
- *Training and testing the data* RB Training and Testing, *and then*;
- *Running inference on a new input data* RB Running Inference using UI.

4.1. RB Webpage for Data Collect

To download a *handwritten digit* model, use the internal *Robin Bird™* webpage. This page was create to collect data for training the model, to help the user only need to write the random number that is sorted on the white space and then, click send.

This step is **NOT** required. Skip it and go direct to RB Running Inference using UI section.

The *Robin Bird™* webpage has model to download and avoid spending time on training.

1. Follow a screenshot of the webpage where should be draw the random numbers:

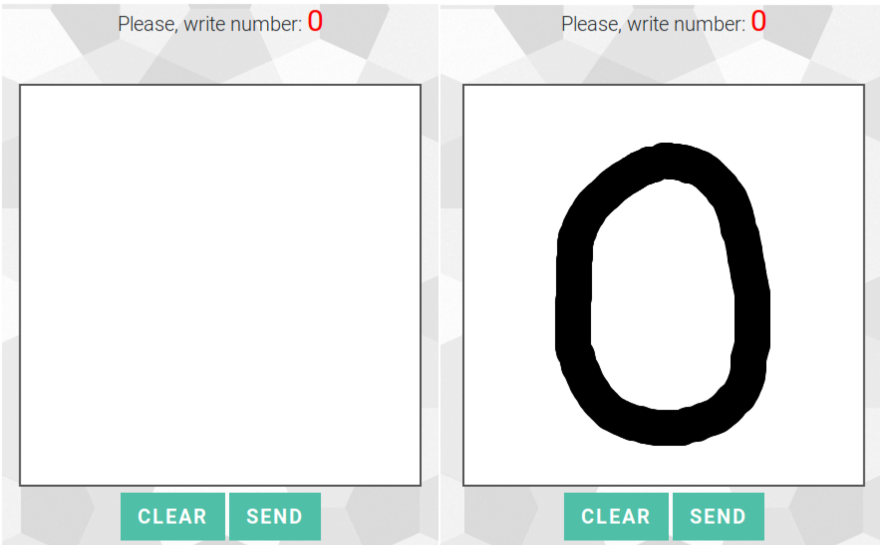

Figure 1. Collecting Data for Training

The *Robin Bird™* webpage trains model every *100* new inputs.

4.2. RB Training and Testing

The *Robin Bird™* webpage provides the model and also the *dataset* for training on any device that has the library installed.

To download the model and using it, go to RB Running Inference using UI section.

The following instructions explain how to get a dataset, how the *dataset* works and how to train the model on a Training on Host GNU/Linux PC or Training on i.MX Board.

4.2.1. RB Model Format

Robin Bird™ has a different format to save the data, the images collected in the *Robin Bird™* webpage are convert to *gray scale*, then all the *black spots* receive *ones* and the *white spots* receive *zero*, as follows:

```
00000000000000000111111000000000
00000000000000000111111100000000
00000000000000001111111110000000
00000000000000001111111110000000
00000000000000011111111110000000
00000000000000011111111110000000
00000000000000111111111110000000
00000000000000111111111110000000
00000000000000111111111100000000
00000000000001111111111100000000
00000000000111111111111100000000
00000000001111111111111100000000
00000000001111111111111110000000
00000000011111111111111110000000
00000000111111111111111110000000
00000001111111111111111100000000
00000001111111100111111100000000
00000011111111000111111100000000
00000111111110001111111100000000
00000111111100001111111100000000
00000011100000001111111000000000
00000000110000000111111100000000
00000000000000000111111100000000
00000000000000001111111100000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000000111111100000000
00000000000000000111111000000000
```

Figure 2. Dataset Example

The images used in this dataset have *32x32* (1024 pixels).

4.2.2. Training on Host GNU/Linux PC

Using the *dataset* containing the binary images it is possible to start computing the model.

1. Download the dataset for training and for testing from *Robin Bird™* webpage:

```
$ hwd-digit hwd-digits-train.rb 100 hwd-digits-test.rb 10 100 0.7
```

Table 1. Parameters

Binary Name	Train Dataset File	Number of Inputs	Test Dataset File	Number of Inputs	Number of Neurons	Learning Rate
hwd-digit	hwd-digits-train.rb	Use dataset size	hwd-digits-test.rb	Use 10	Use 100 (regular)	Use 0.7 (experimental)

Here is a simple 32x32 (1024 pixels) image that was convert into gray scale:

4.2.3. Training on i.MX Board

4.3. RB Running Inference using UI

Chapter 5. Revision History

Version	Author	Changes	Date
1.0	Diego Dorta	Original release 1.0	10/15/2015