Robin Bird™ Machine Learning Library

Version 1.0, September 25, 2019

Table of Contents

Chapter 1. INTRODUCTION

Robin Bird™ (or RB™) is a small and fully experimental *Machine Learning* library, 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 from low to medium-scale *Machine Learning* solutions.

1.1. LICENSE

RB™ 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. RB™

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. *RB™* 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*. *RB™* 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. *RB™* include algorithms for recognition problems, it can be integrated with *ML* solutions from low to medium scale.

Chapter 2. GETTING STARTED

To contribute to the *RB™* library, follow the instructions in the Building RB™ on GNU/Linux Host PC section, otherwise to run the sample applications on *i.MX board* use the instructions in the Building RB™ on eIQ™ section.

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

2.1. Prerequisites

The list of the required packages/libraries for running *RB™* on a regular *GNU/Linux Host PC* are:

Package/Library	Description
gtkmm	*C++ interfaces for GTK+ and GNOME*
libpng	*C functions for handling PNG images*

This list of dependencies may increase over time.

2.1.1. 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.

2.2. Building RB™ on GNU/Linux Host PC

2.2.1. RB™ Source Code

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

   Host GNU/Linux PC Terminal

   ```
   $ git clone https://bitbucket.sw.nxp.com/scm/imxs/robin-bird.git 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

```
$ 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:

Name	Description	Default
ROBIN_BIRD	*enables RB™ library*	*on*
SAMPLES	*enables Robin Bird™ samples*	*on*
INSTURMENT_FOR_FUZZING	*enables fuzzing tests*	*off*
ROBIN_BIN_INSTALL_DIR	*defines samples directory*	*on*

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

Host GNU/Linux PC Terminal

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

① *configuring the project using cmake and the choosen flags*.

② *building 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 to the define directory on ROBIN_BIN_INSTALL_DIR*.

② *configuring the library*.

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

2.2.2. Helper Tool

1. Use *pkg-config* helper tool to check if *RB™* is correctly installed:

Host GNU/Linux PC Terminal

```
$ pkg-config robin --libs --cflags ①
```

① *running pkg-config to retrieve library path*.

The *pkg-config* retrieves the following output:

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

If all goes well, go the RB™ Applications section.

2.3. Building RB™ on eIQ™

1. Download the latest *GNU/Linux BSP* documentation available on IMX-SW.
2. Follow the steps from the Yocto Project User's Guide to create a build environment.

At the time this tutorial was written, the latest release available was **L4.14.98-2.0.0_ga**.

3. Apply the this patch to the *meta-fsl-bsp-release* meta layer:

Host GNU/Linux PC Terminal

```
$ cd /sources/meta-fsl-bsp-release ①
$ git am <downloaded_patch> ②
```

① *entering the meta-fsl-bsp-release folder.*

② *applying the patch.*

4. Create the build directory:

Host GNU/Linux PC Terminal

```
$ DISTRO=fsl-imx-xwayland MACHINE=imx8qmmek source fsl-setup-release.sh -b build-xwayland ①
```

① *creating the build directory.*

5. Include the following line at the end of the *build-xwayland/conf/local.conf* file:

Host GNU/Linux PC Terminal

```
IMAGE_INSTALL_append += "robin-bird" ①
```

① *including RB™ recipe.*

6. Build the image:

Host GNU/Linux PC Terminal

```
$ bitbake fsl-image-qt5 ①
```

① *generating the image.*

7. Generating a bootable *SD Card*:

Host GNU/Linux PC Terminal

```
$ bunzip /tmp/deploy/image/<image_name>.sdcard.bzip2 ①
$ dd if=<image_name>.sdcard of=/dev/sd<x> status=progress bs=1M && sync ②
```

① *uncompressing the image.*

② *burning the image to the SD Card.*

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.

<*image_name*> refers to the respective board image.

Chapter 3. RB™ Applications

3.1. RB™ Hello World Example

The next instructions can be used to test if the library is working properly:

1. Use the following *C++* *source code* to create a simple example using *RB™* functions:
2. To compile the code, use the next command-line:

   Host GNU/Linux PC Terminal

   ```
   $ g++ example.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.

3.2. RB™ HandWriting Digit Recognition Application

RB™ has a simple *Neural Network* implemented for recognizing handwritten digits. The next sections approach how to run the *UI* application using a *RB™* model, and how to train a new model and populate the dataset.

- *Running inference on a new input data using command line* RB™ Running Inference using Command-line.
- *Running inference on a new input data using UI application* RB™ Running Inference using UI Application.
- *Using a dataset to train a new model and test it* RB™ Training a New Model.
- *Populating the dataset using RB™ web page* RB™ Webpage Populating the Dataset.

3.2.1. RB™ Running Inference using Command-line

1. Go the board and run the following command-line:

   i.MX8 Board Terminal

   ```
   # hwd-digit-input <image> ①
   ```

   ① *running inference on a single image.*

   

   The image must be a *32x32* (1024 pixels) gray scale.

2. The output is a vector from 0 to 9 and it position has the estimated result percent.

3.2.2. RB™ Running Inference using UI Application

1. To run inference in a real time handwriting digit:

   i.MX8 Board Terminal

   ```
   # handWrittenDigitInputUI <model> ①
   ```

   ① *running inference on a single image.*

2. It opens an UI and just write the number from 0 to 9:

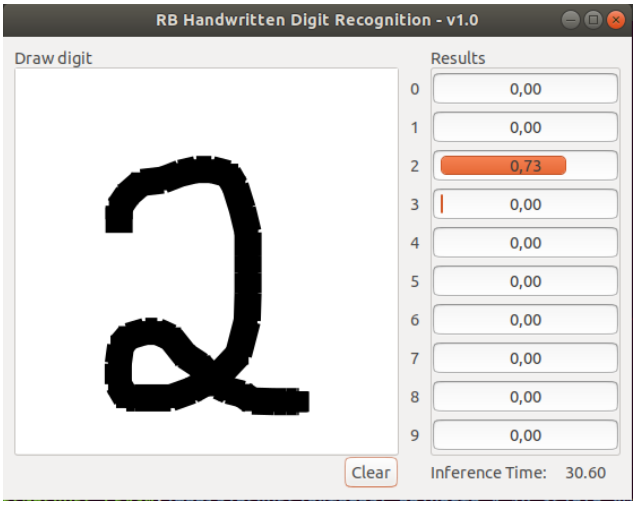

Figure 1. UI Application

The inference time is around *30 milliseconds*.

3.2.3. RB™ Training a New Model

The following instructions explain how to get a *dataset*, how the *dataset* works and how to train the model.

This step is **NOT** required. Download the model on *RB™ webpage*

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

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

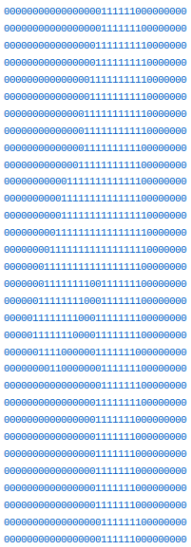

Figure 2. Dataset Example

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

Training the Dataset

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 *RB™ webpage*:

```
$ hwd-digit hwd-digits-train.rb 100 hwd-digits-test.rb 10 100 0.7 ①
```

① *training a new model.*

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)

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

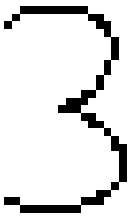

Figure 3. Gray Scale Number

3.2.4. RB™ Webpage Populating the Dataset

To download a *handwritten digit* model, use the internal *RB™* 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, then click send.

The *RB™* 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:

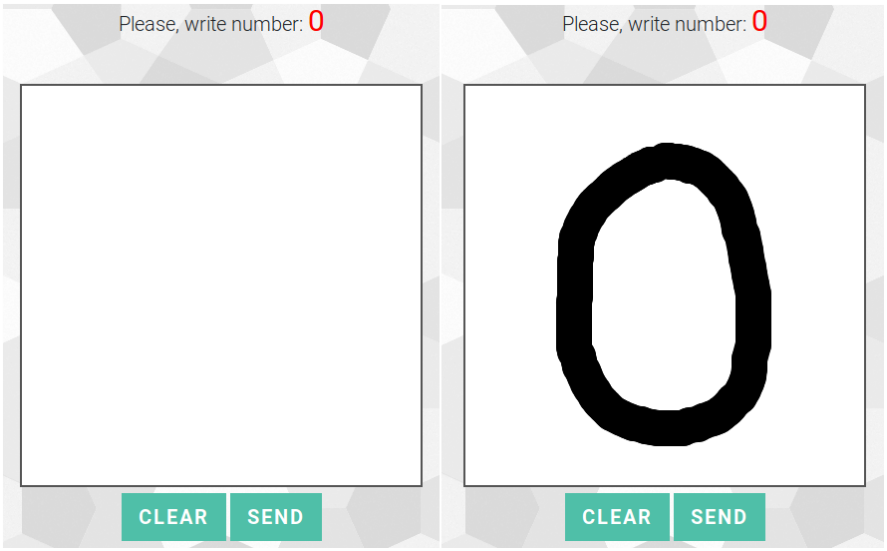

Figure 4. Collecting Data for Training

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

Chapter 4. Revision History

Version	Author	Changes	Date
1.0	Diego Dorta, Marco Franchi, Rogerio Pimental and Vanessa Maegima	Original release 1.0	10/29/2019