

Lab Cortex-M4

*Machine Learning (Deep Learning, Artificial Neural Network) using
Microcontroller : Embedded AI*



Human Brain vs Computer



History of AI

S/Z/Z/G/Y

A.I. TIMELINE

1950

TURING TEST

Computer scientist Alan Turing proposes a test for machine intelligence. If a machine can trick humans into thinking it is human, then it has intelligence

1955

A.I. BORN

Term 'artificial intelligence' is coined by computer scientist, John McCarthy to describe "the science and engineering of making intelligent machines"

1961

UNIMATE

First industrial robot, Unimate, goes to work at GM replacing humans on the assembly line

1964

ELIZA

Pioneering chatbot developed by Joseph Weizenbaum at MIT holds conversations with humans

1966

SHAKY

The 'first electronic person' from Stanford, Shakey is a general-purpose mobile robot that reasons about its own actions

A.I. WINTER

Many false starts and dead-ends leave A.I. out in the cold

1997

DEEP BLUE

Deep Blue, a chess-playing computer from IBM defeats world chess champion Garry Kasparov

1998

KISMET

Cynthia Breazeal at MIT introduces Kismet, an emotionally intelligent robot insofar as it detects and responds to people's feelings



1999

AIBO

Sony launches first consumer robot pet dog AIBO (AI robot) with skills and personality that develop over time



2002

ROOMBA

First mass produced autonomous robotic vacuum cleaner from iRobot learns to navigate and clean homes



2011

SIRI

Apple integrates Siri, an intelligent virtual assistant with a voice interface, into the iPhone 4S



2011

WATSON

IBM's question answering computer Watson wins first place on popular \$1M prize television quiz show *Jeopardy*



2014

EUGENE

Eugene Goostman, a chatbot passes the Turing Test with a third of judges believing Eugene is human



2014

ALEXA

Amazon launches Alexa, an intelligent virtual assistant with a voice interface that completes shopping tasks



2016

TAY

Microsoft's chatbot Tay goes rogue on social media making inflammatory and offensive racist comments



2017

ALPHAGO

Google's A.I. AlphaGo beats world champion Ke Jie in the complex board game of Go, notable for its vast number (2^{170}) of possible positions

Neural Net CPU

"My CPU is a neural-net processor; a learning computer."

- T-800 - *Terminator 2: Judgment Day*

The **Neural Net CPU** is a "learning computer" and one of the most powerful [microprocessors](#) ever built. All of the [battle units](#) deployed by [Skynet](#) contain a Neural Net CPU.

Housed within inertial shock dampers within each battle unit, the CPU gives Skynet the ability to control it's units directly, or allow them to function by themselves, learning from a pre-programmed knowledge base as they go. This means that each battle unit has the potential to adapt to its situation, and literally reason through problems and tactical maneuvers. In the case of the various [Terminator series](#), this means that they can learn to behave more like humans in order to be better equipped for [infiltration](#).

It is developed by [Miles Bennett Dyson](#), director of Special Projects at [Cyberdyne Systems Corporation](#), via reverse engineering on the wreckage of a [T-800 Terminator](#) in 1984.



Cameron's CPU



ARTIFICIAL INTELLIGENCE

IS NOT NEW

ARTIFICIAL INTELLIGENCE

Any technique which enables computers to mimic human behavior



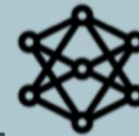
MACHINE LEARNING

AI techniques that give computers the ability to learn without being explicitly programmed to do so



DEEP LEARNING

A subset of ML which make the computation of multi-layer neural networks feasible



1950's

1960's

1970's

1980's

1990's

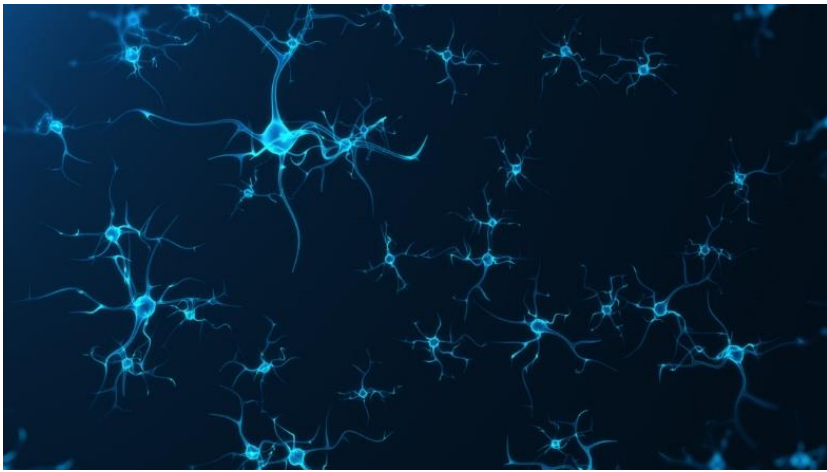
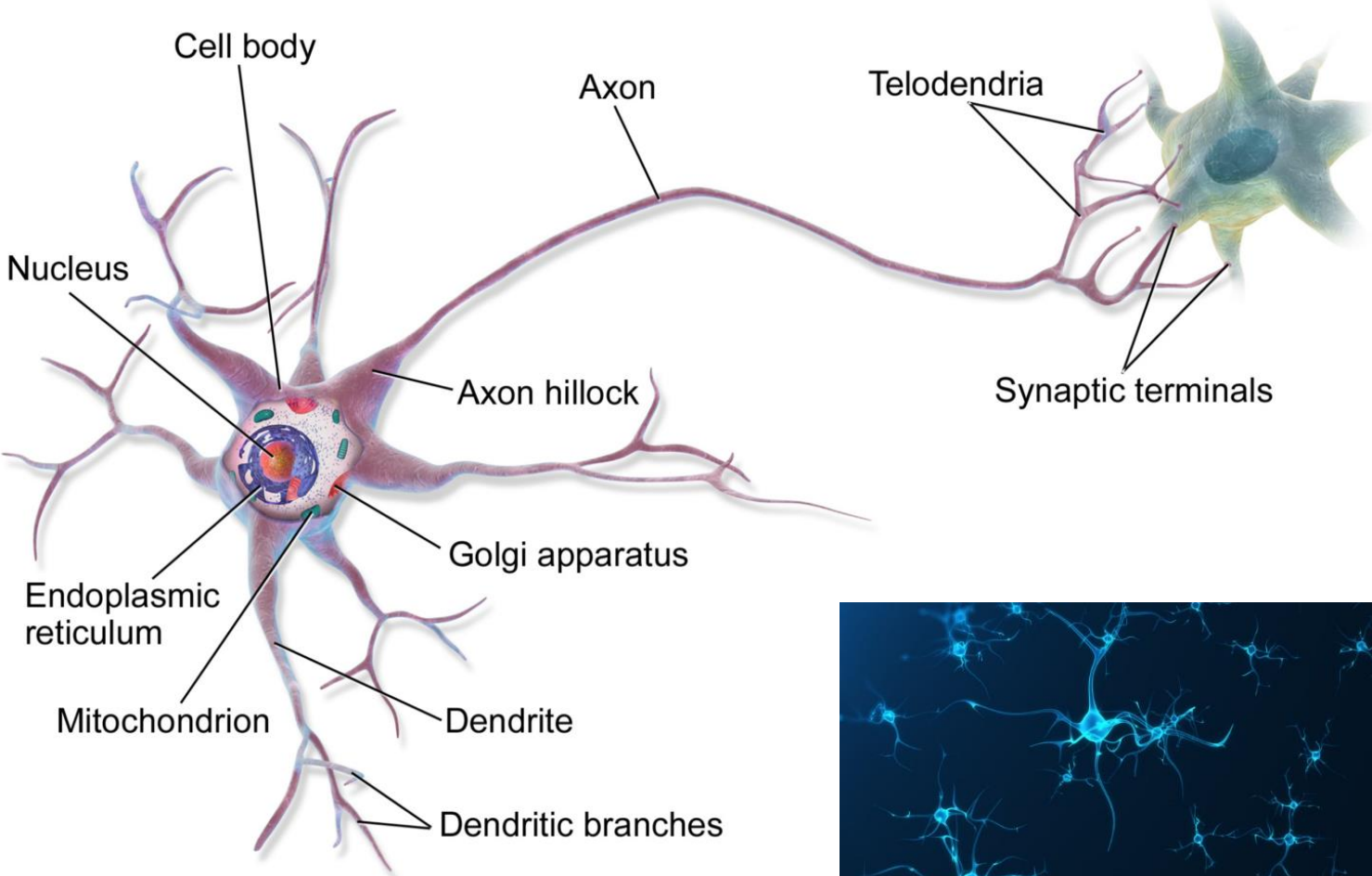
2000's

2010's

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

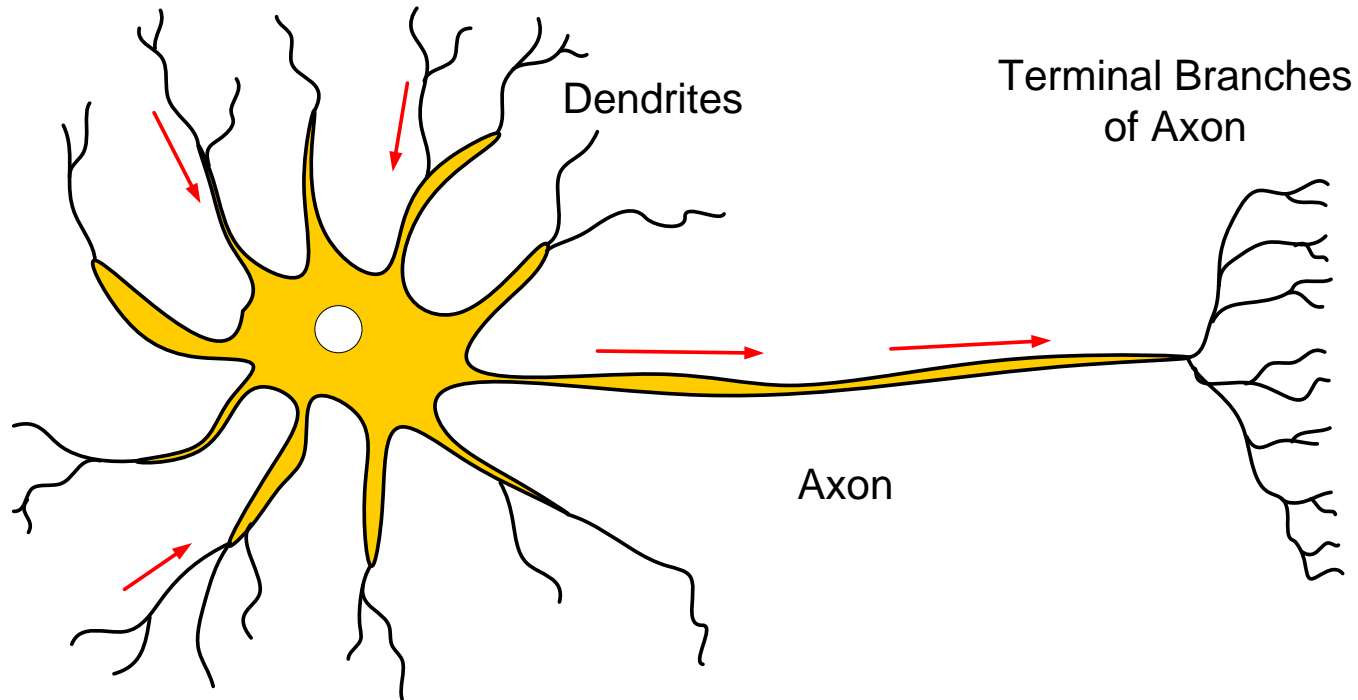
Neuron



Neuron structure taken from Wikipedia

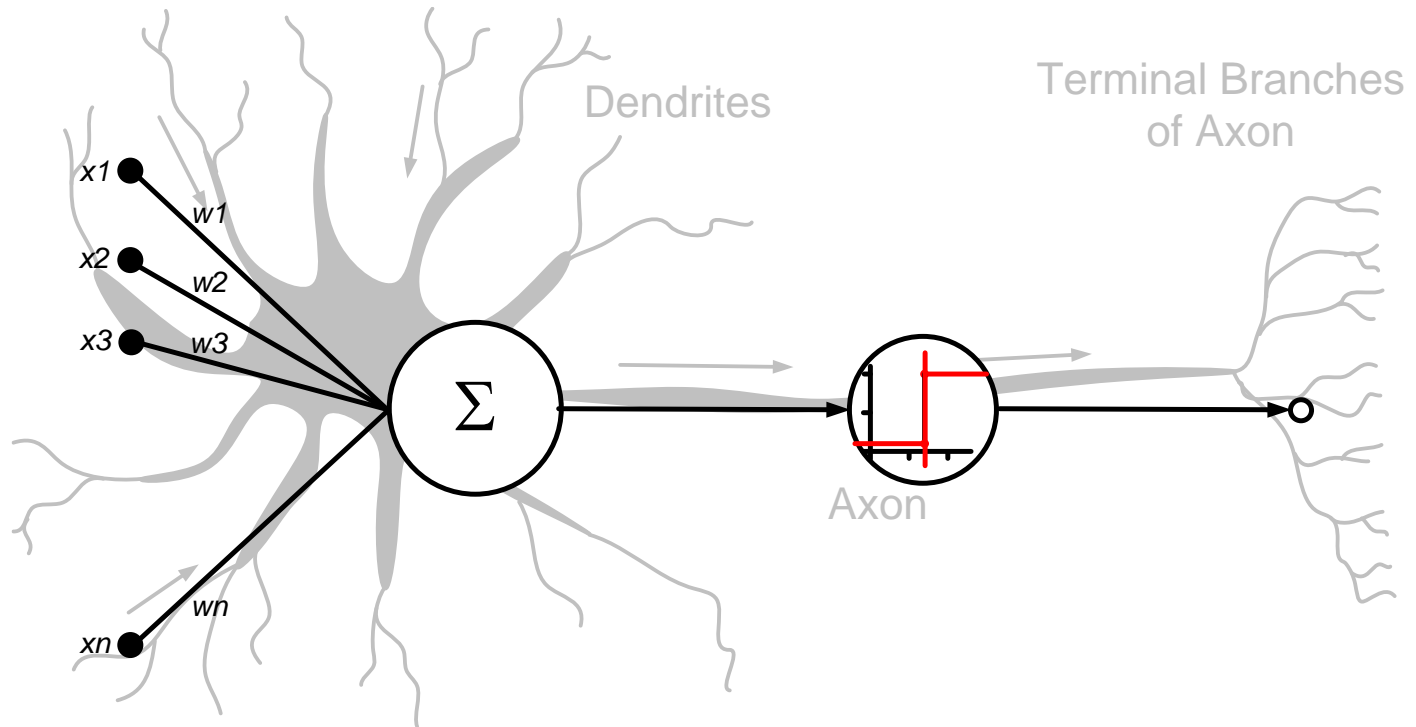
Biologically Inspired

- Electro-chemical signals
- Threshold output firing



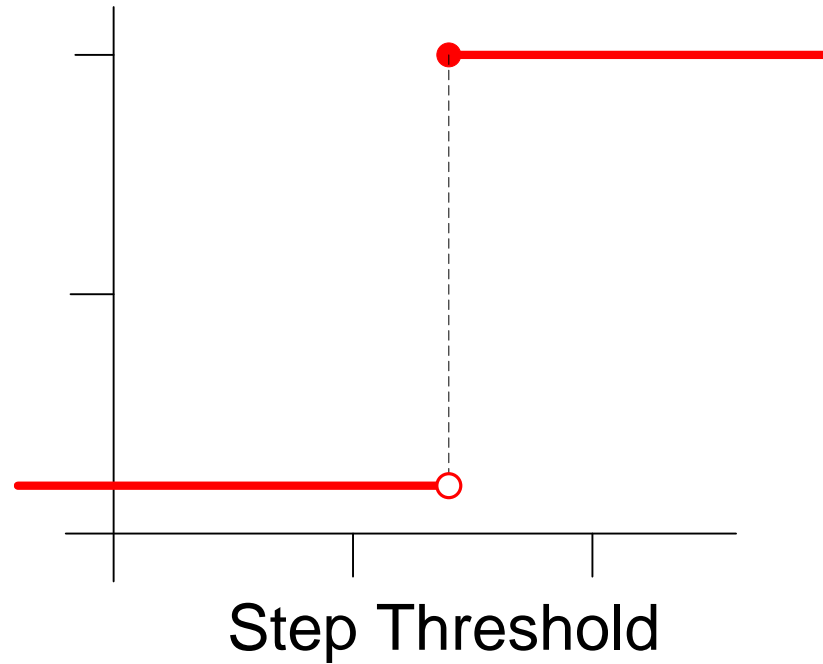
The Perceptron

- Binary classifier functions
- Threshold activation function



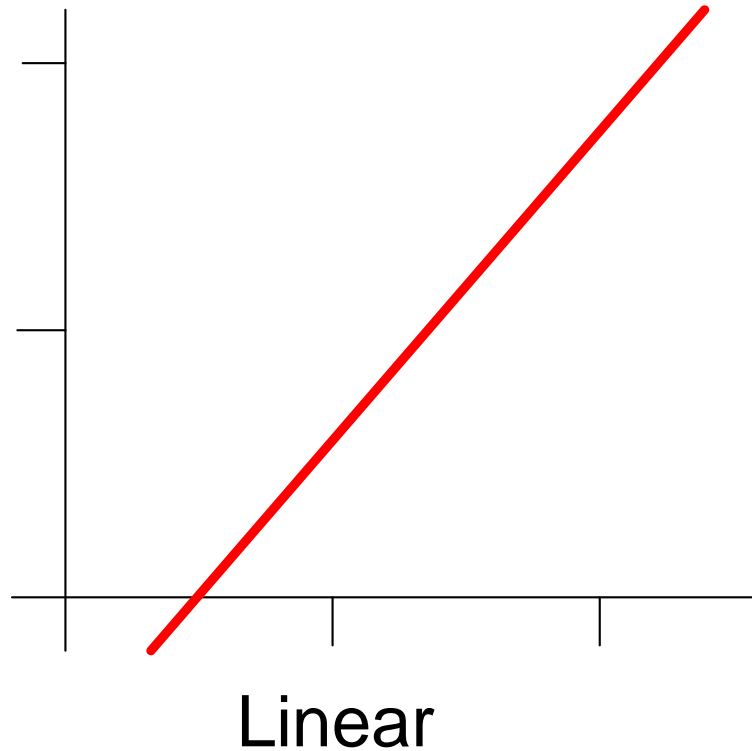
The Perceptron: Threshold Activation Function

- Binary classifier functions
- Threshold activation function



Linear Activation functions

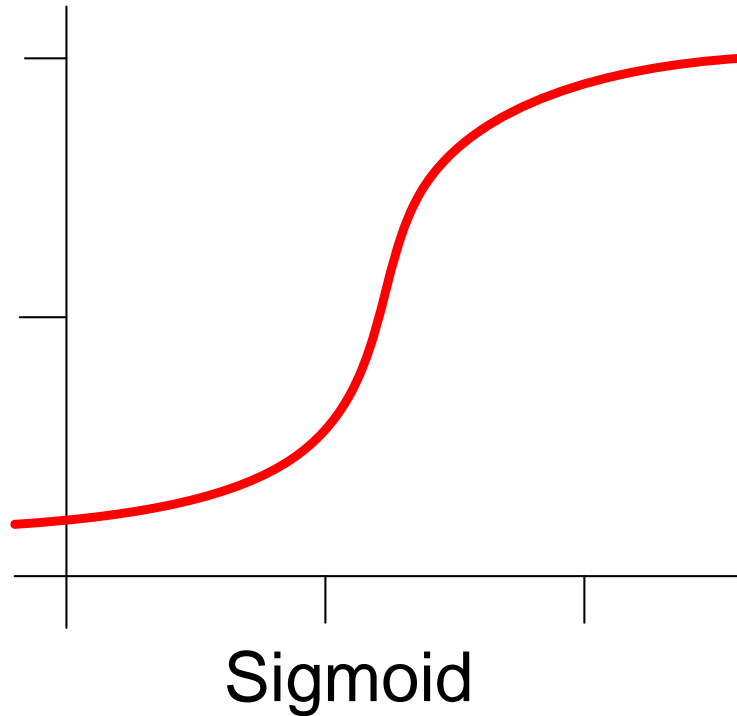
- Output is scaled sum of inputs



$$y = u = \sum_{n=1}^N w_n x_n$$

Nonlinear Activation Functions

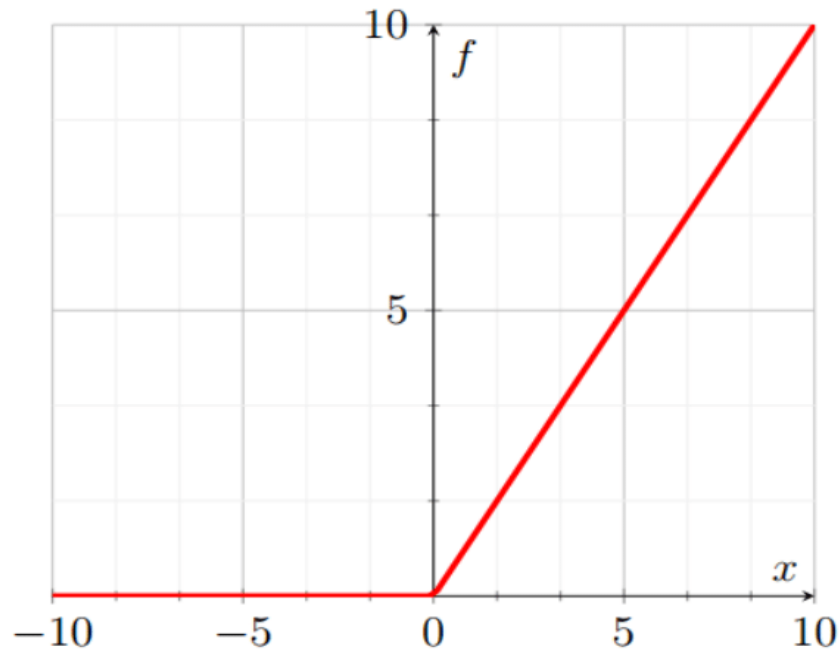
- Sigmoid Neuron unit function



$$y_{hid}(u) = \frac{1}{1 + e^{-u}}$$

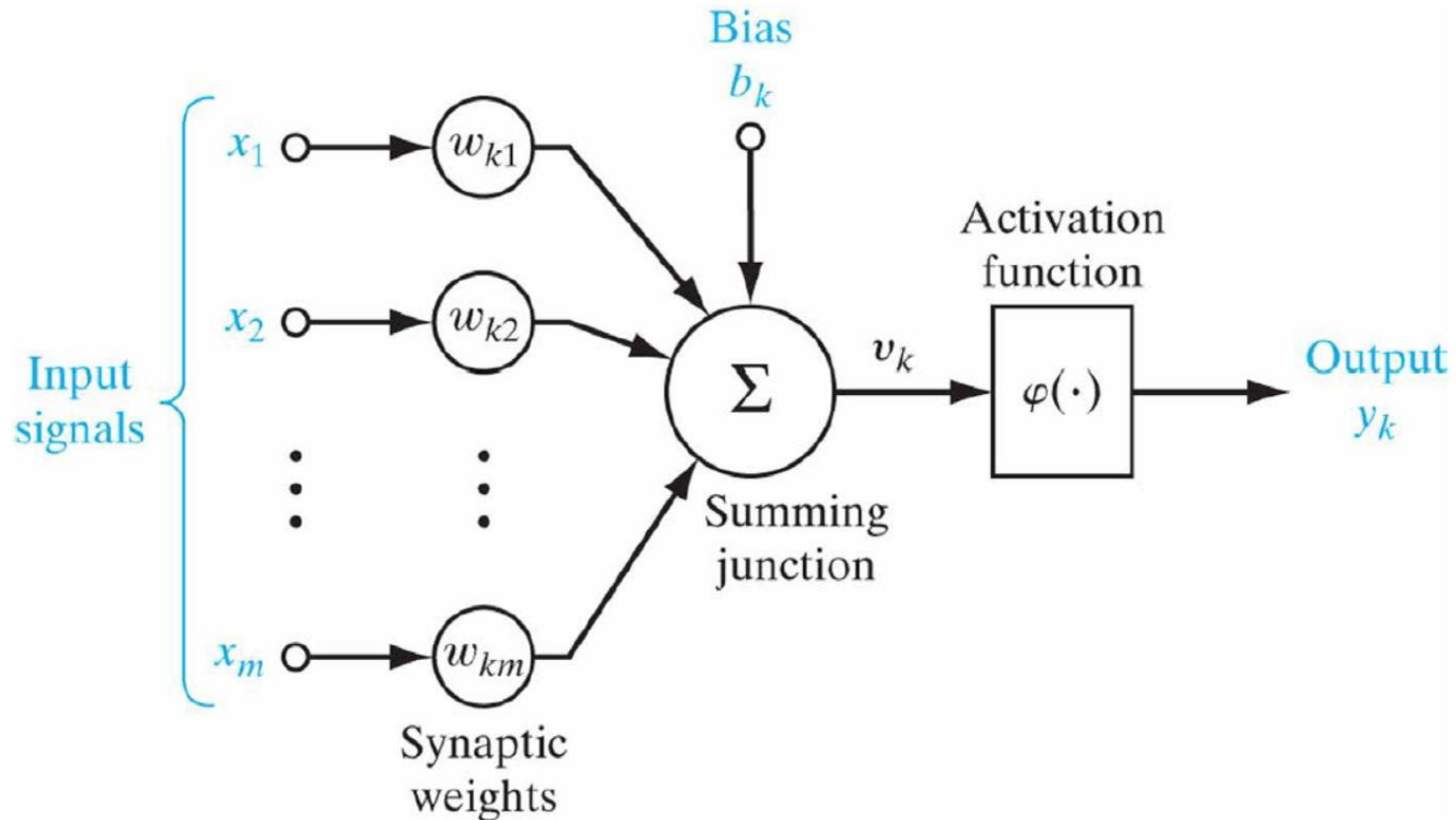
Nonlinear Activation Functions

- ReLU (Rectified Linear Unit)



Geoffrey E Hinton
University of Toronto

Model of a single neuron



Neuron Model

$$u_k = \sum_{j=1}^m w_{kj} x_j$$

Adder, weighted sum, linear combiner

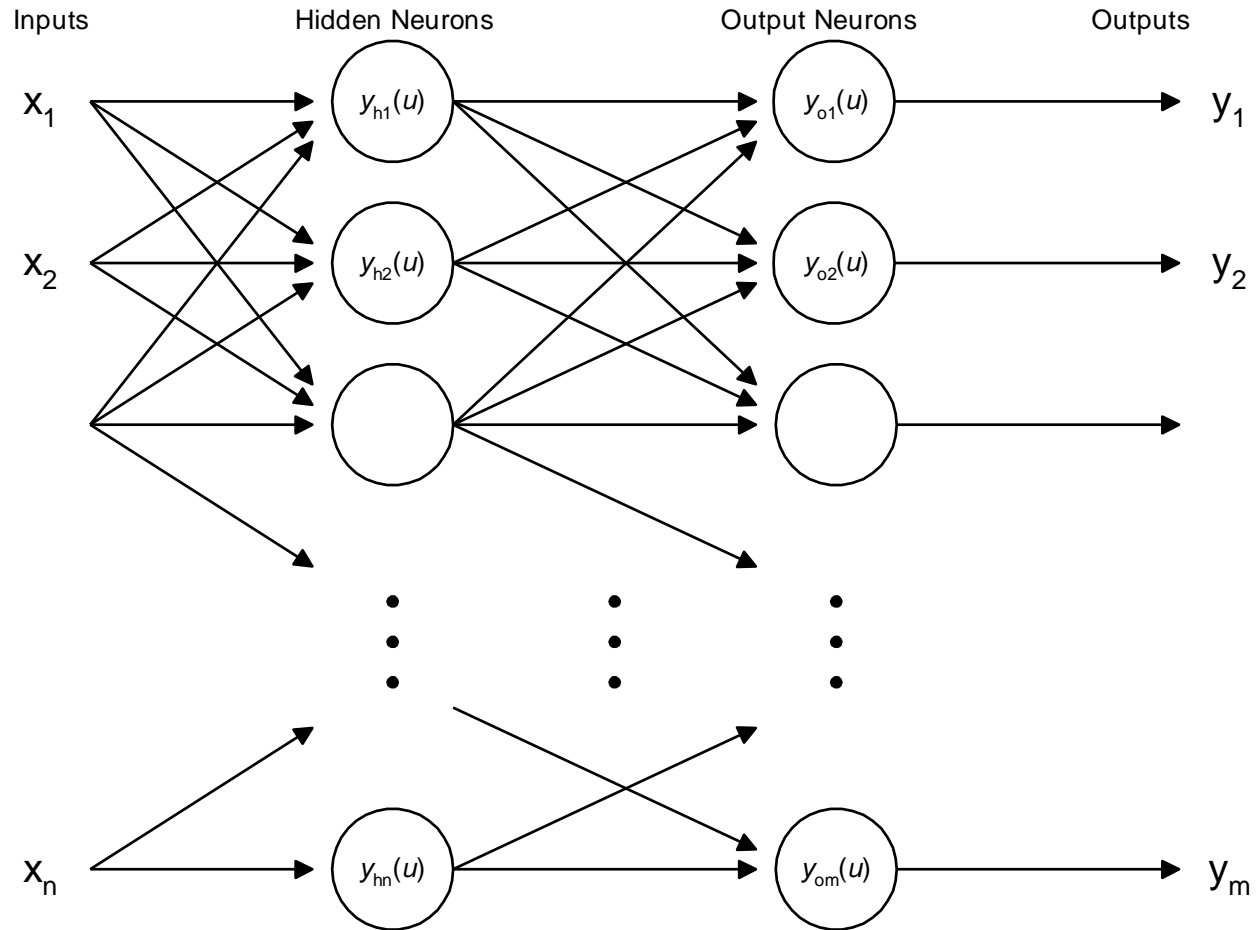
$$v_k = u_k + b_k$$

Activation potential; b_k : bias

$$y_k = \varphi(v_k)$$

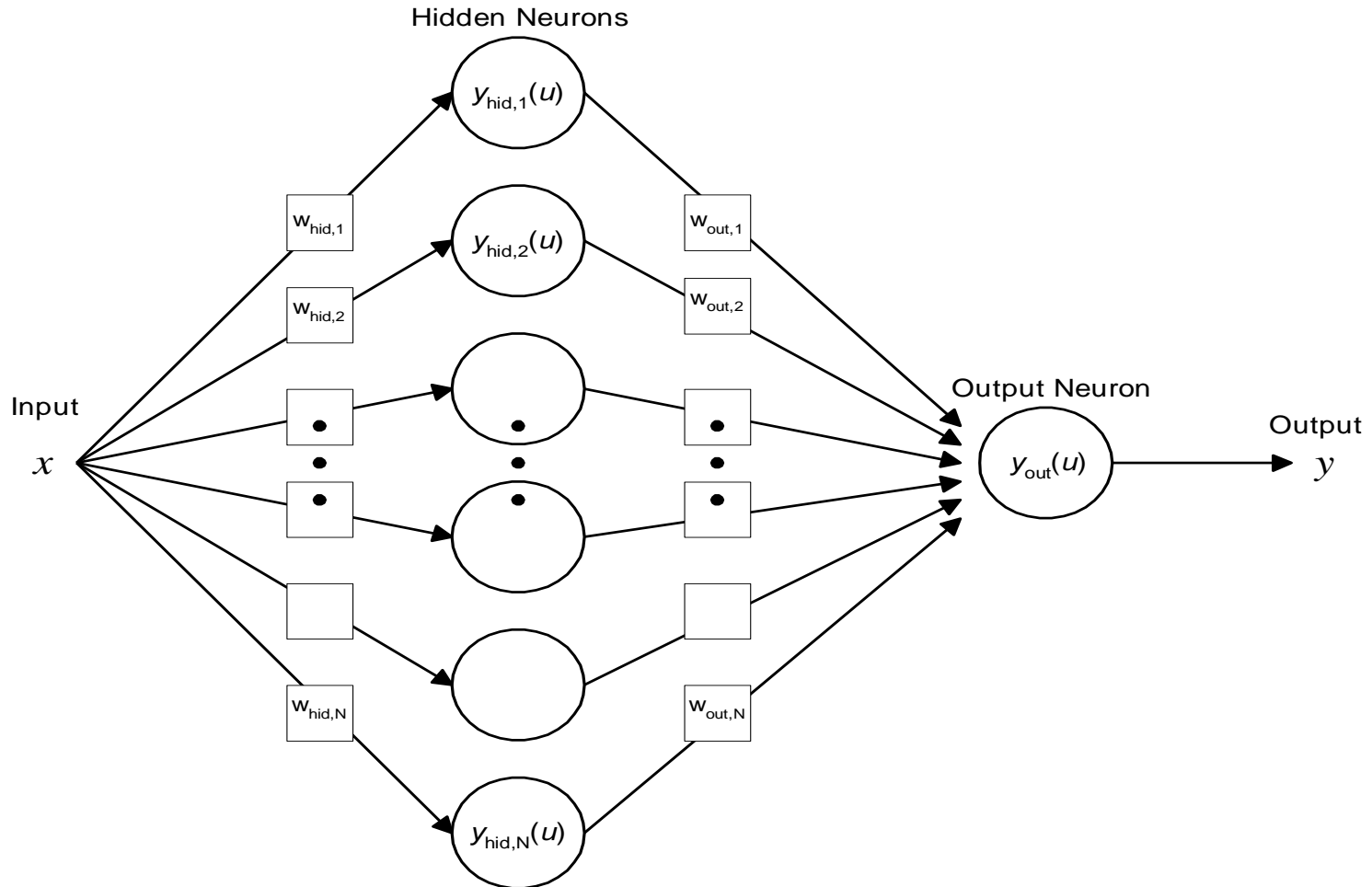
Output; φ : activation function

Layered Networks



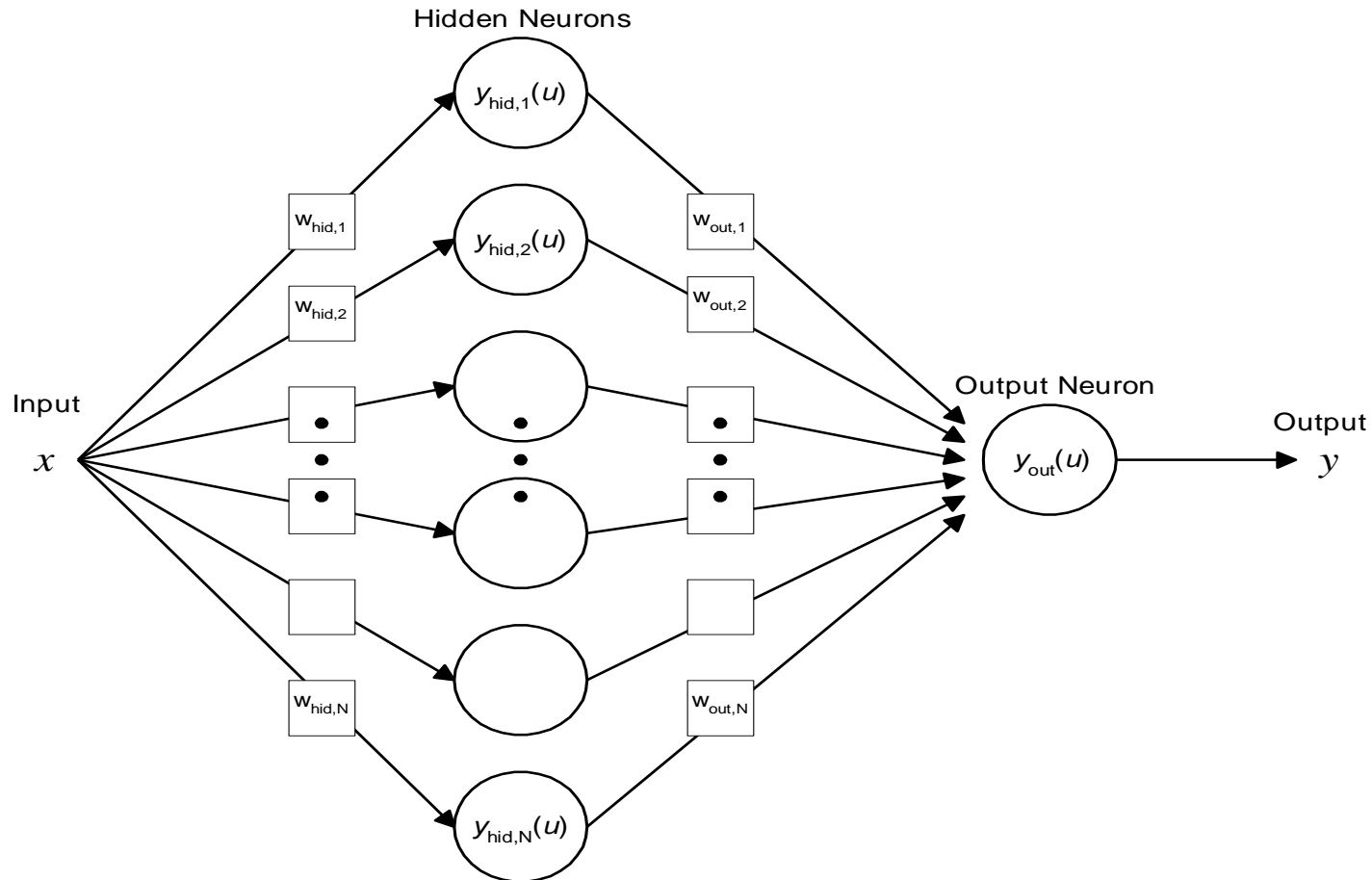
SISO Single Hidden Layer Network

- Can represent and single input single output functions: $y = f(x)$



Training Data Set

- Adjust weights (w) to learn a given target function: $y = f(x)$
- Given a set of training data $X \rightarrow Y$



Training Weights: Error Back-Propagation (BP)

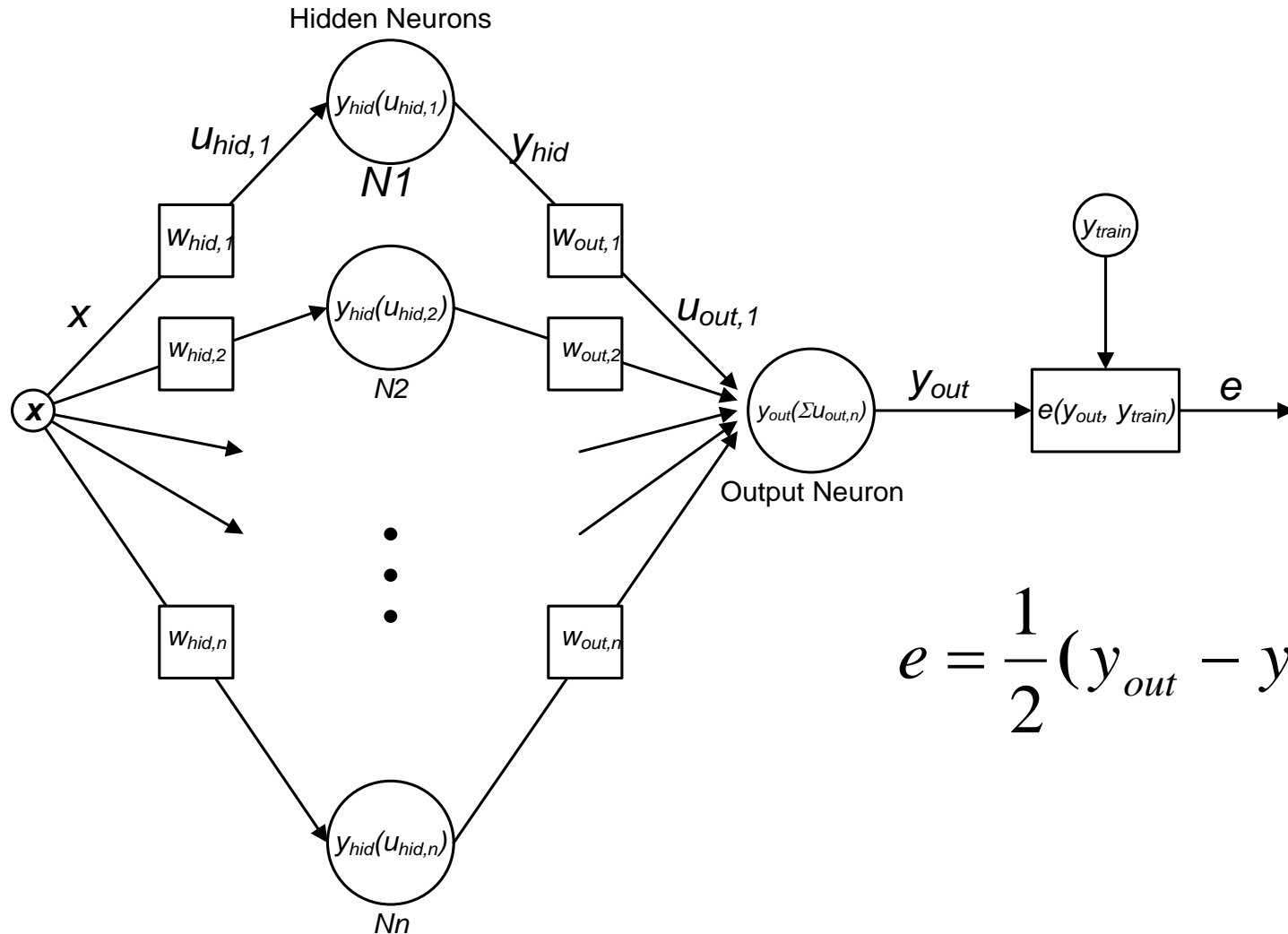
- Weight update formula:

$$w(k + 1) = w(k) + \Delta w$$

$$\Delta w(i) = \eta * \frac{\partial e(i)}{\partial w}$$

Error Back-Propagation (BP)

Training error term: e



Example: The XOR Problem

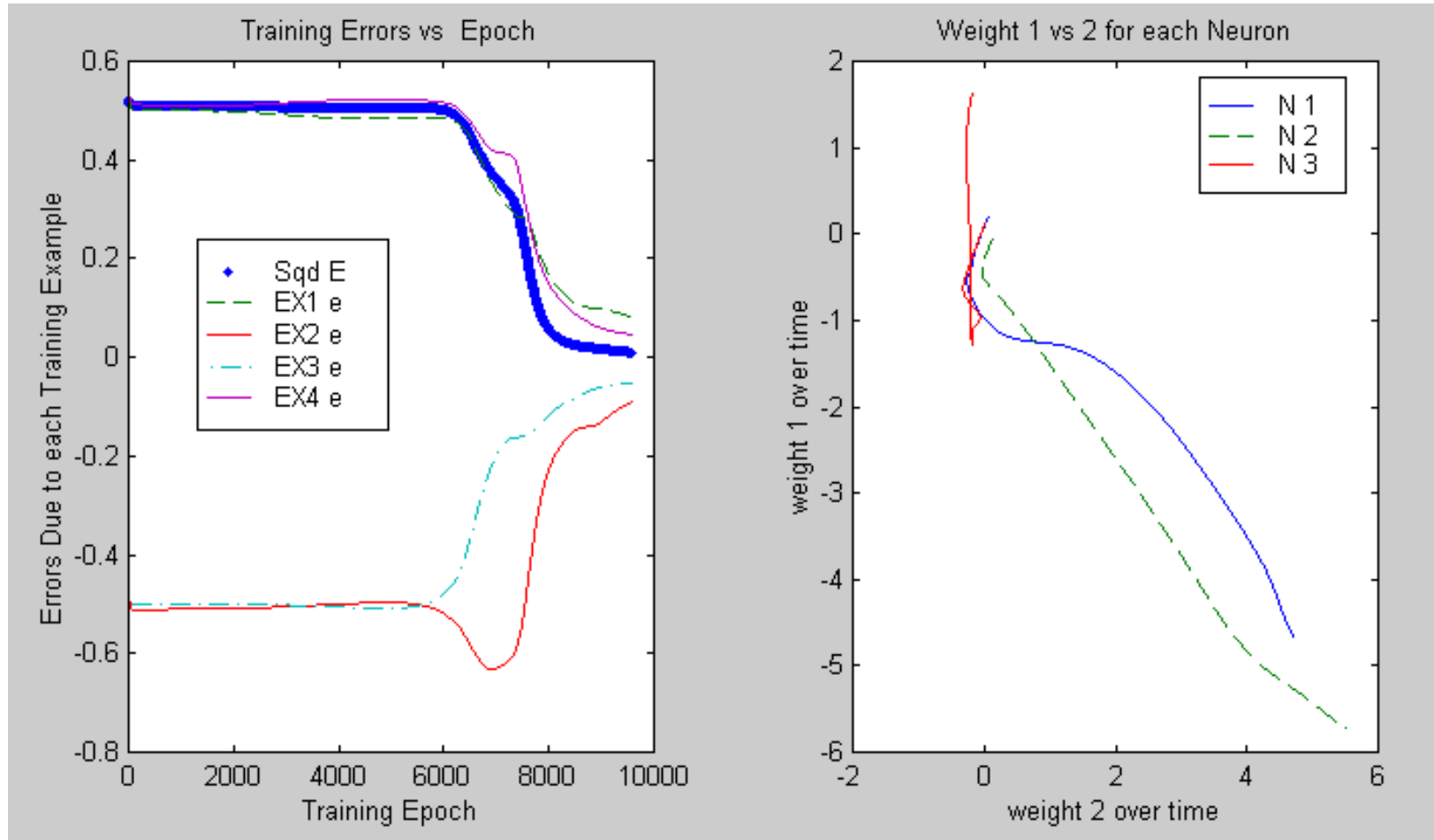
- Single hidden layer: 3 Sigmoid neurons
- 2 inputs, 1 output

Desired I/O table (XOR):

	x1	x2	y
Example 1	0	0	0
Example 2	0	1	1
Example 3	1	0	1
Example 4	1	1	0

Example: The XOR Problem

- Training error over epoch



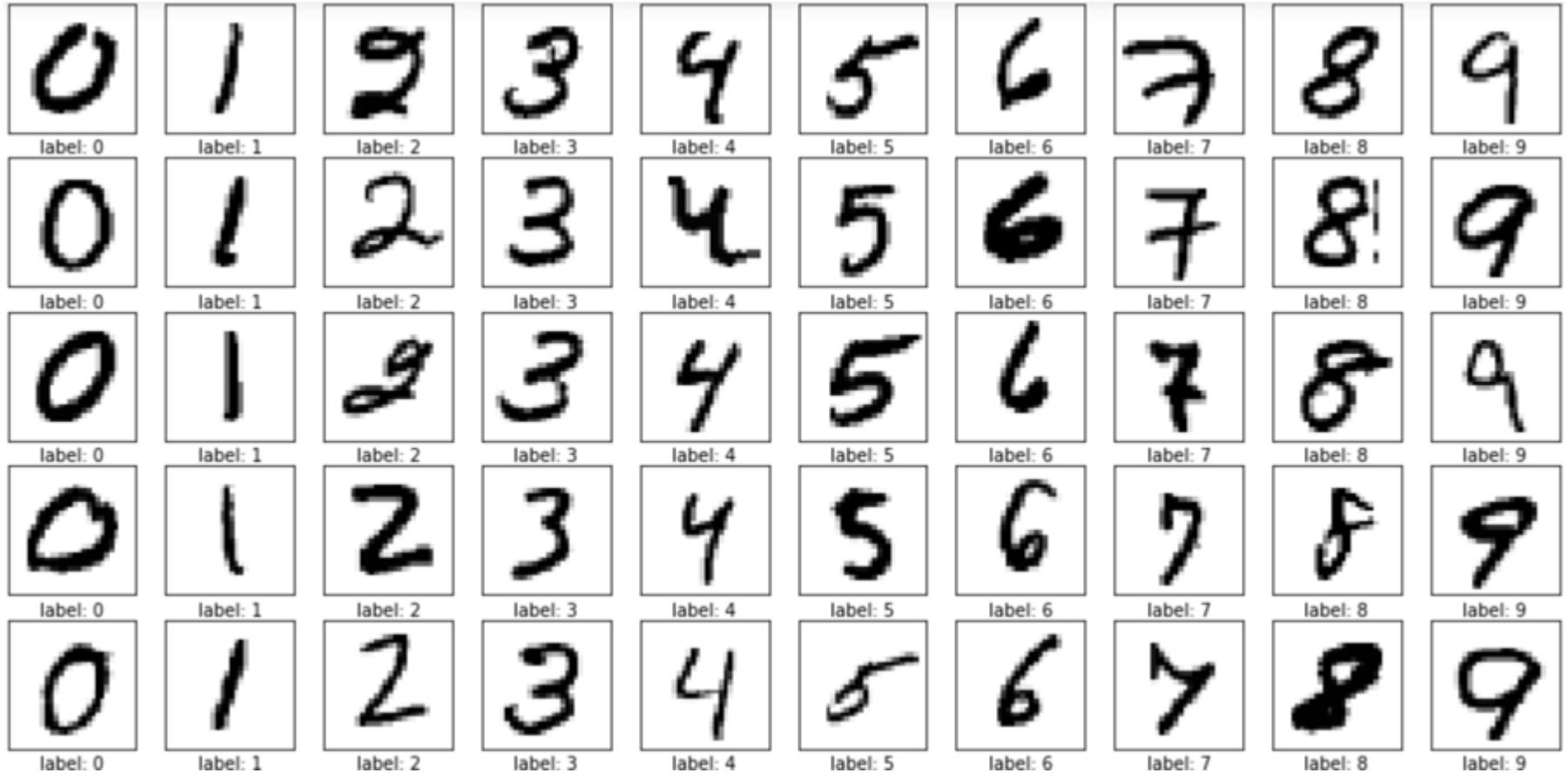
Example: The XOR Problem

- Mapping produced by the trained neural net:

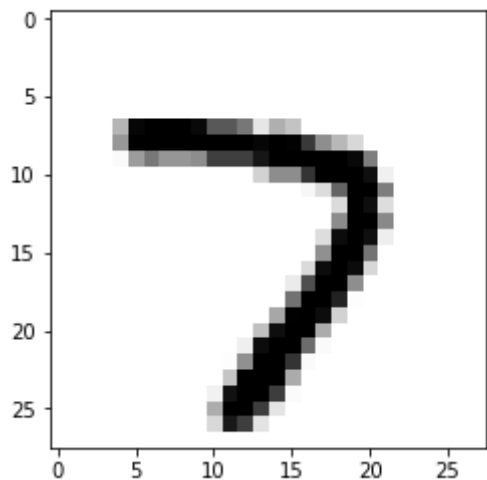
	x1	x2	y
Example 1	0	0	0.0824
Example 2	0	1	0.9095
Example 3	1	0	0.9470
Example 4	1	1	0.0464

Embedded AI Example

- MNIST Data Set



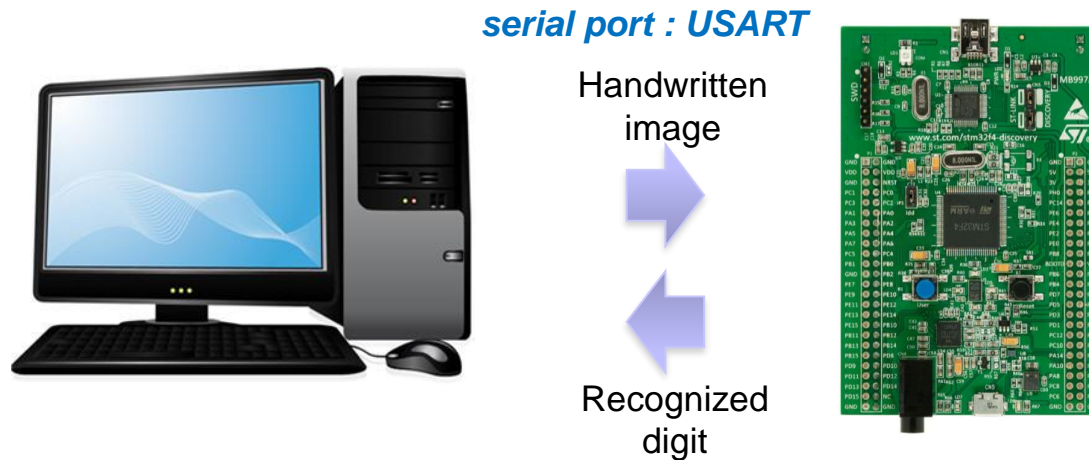
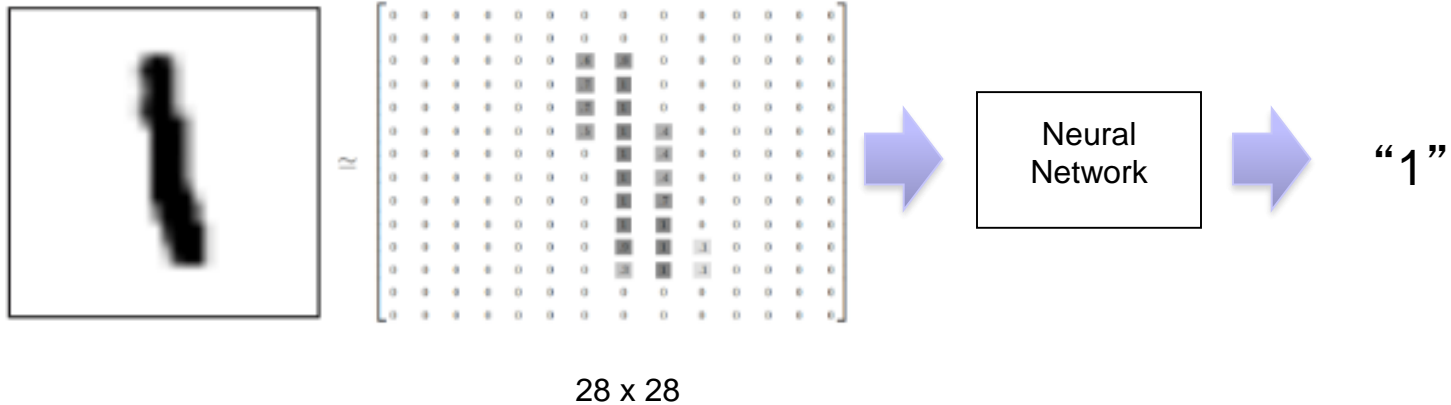
MNIST Data



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4a	f9	fe	fe	fe	f5	a7	a7	88	19	50	3c	0	0	0	0	0	0	0	0	0	0	
0	68	fe	fe	fe	fe	fe	fe	fe	fe	f9	fe	fc	c5	71	47	27	0	0	0	0	0	0	
0	5	63	87	69	69	72	c0	c0	c0	e9	fe	fe	fe	fe	fe	f6	81	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	2d	72	72	cb	fe	fe	fe	f0	f	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	8	23	9b	fe	fe	82	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	fe	f1	22	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	fe	fe	76	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	f3	fe	f0	11	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	6f	fe	fe	8b	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	25	f3	fe	f4	28	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	14	b0	fe	fe	71	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	8c	fe	fe	dc	2	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	58	fd	fe	f3	2d	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	3f	f1	fe	fe	53	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	10	f3	fe	fe	93	5	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	2	6f	fe	fe	cb	5	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	3a	fe	fe	fe	54	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	e	ed	fe	ff	c2	4	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	52	fe	fe	c2	1b	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	27	e6	c1	1c	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Embedded AI Example using MNIST Data Set

0–9 handwritten digit recognition:



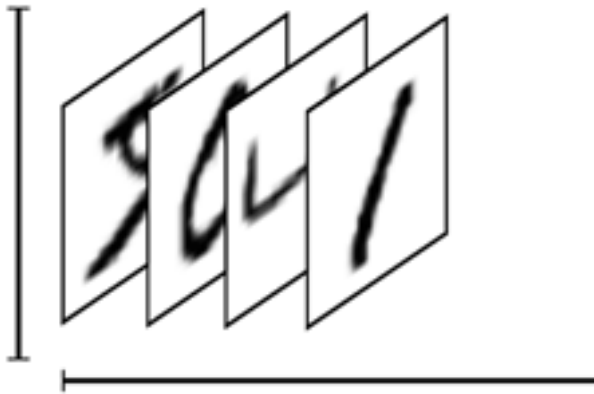
MNIST Data maintained by Yann LeCun: <http://yann.lecun.com/exdb/mnist/>
Keras provides data sets loading function at <http://keras.io/datasets>

Training

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

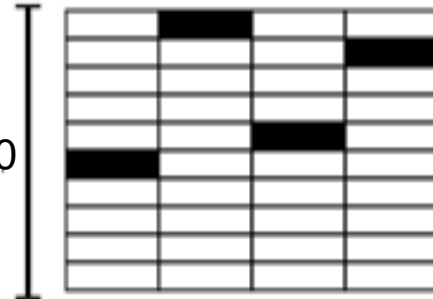
numpy array

28 x 28
=784



Number of training examples

10



Number of training examples

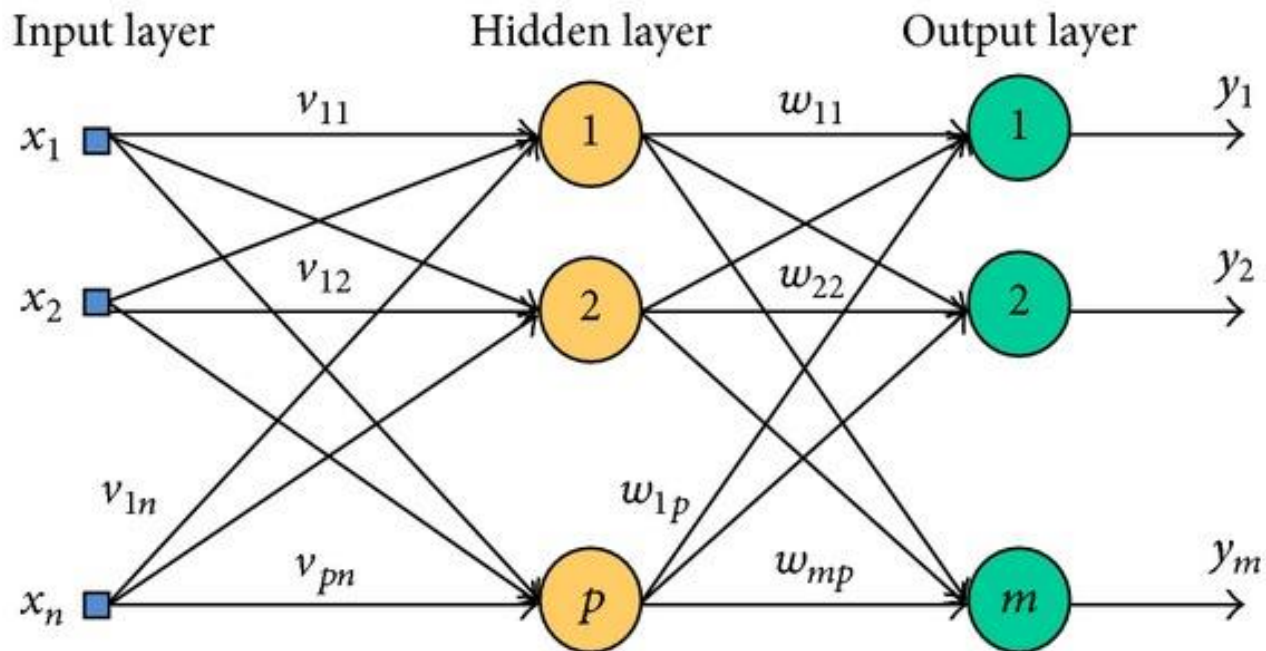


- Training on PC
- Save neural network model
- Convert model to C program
- Compile and download to target

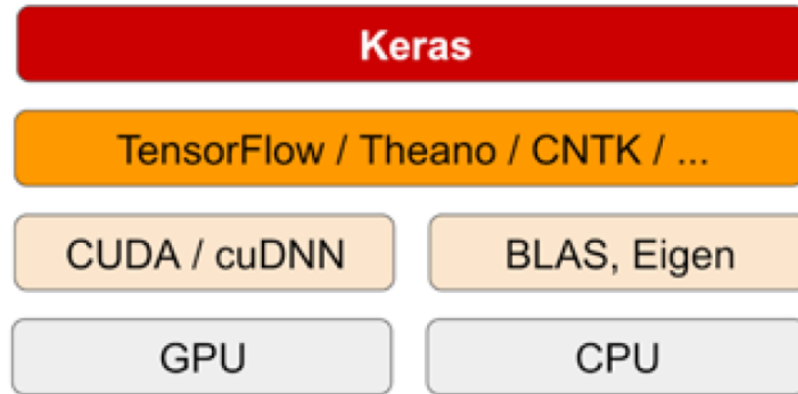


Neural Network Model

- $n=28 \times 28$
- $m=10$

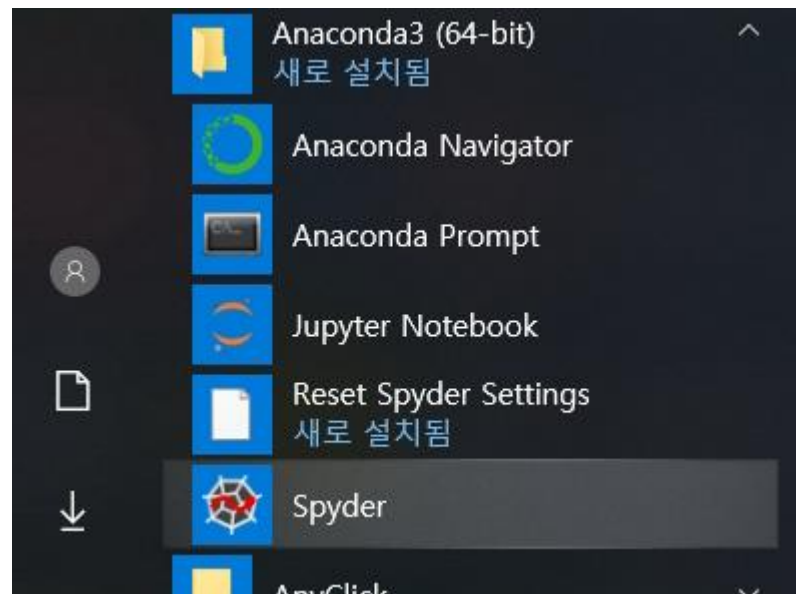


Deep-Learning Software and Hardware Stack

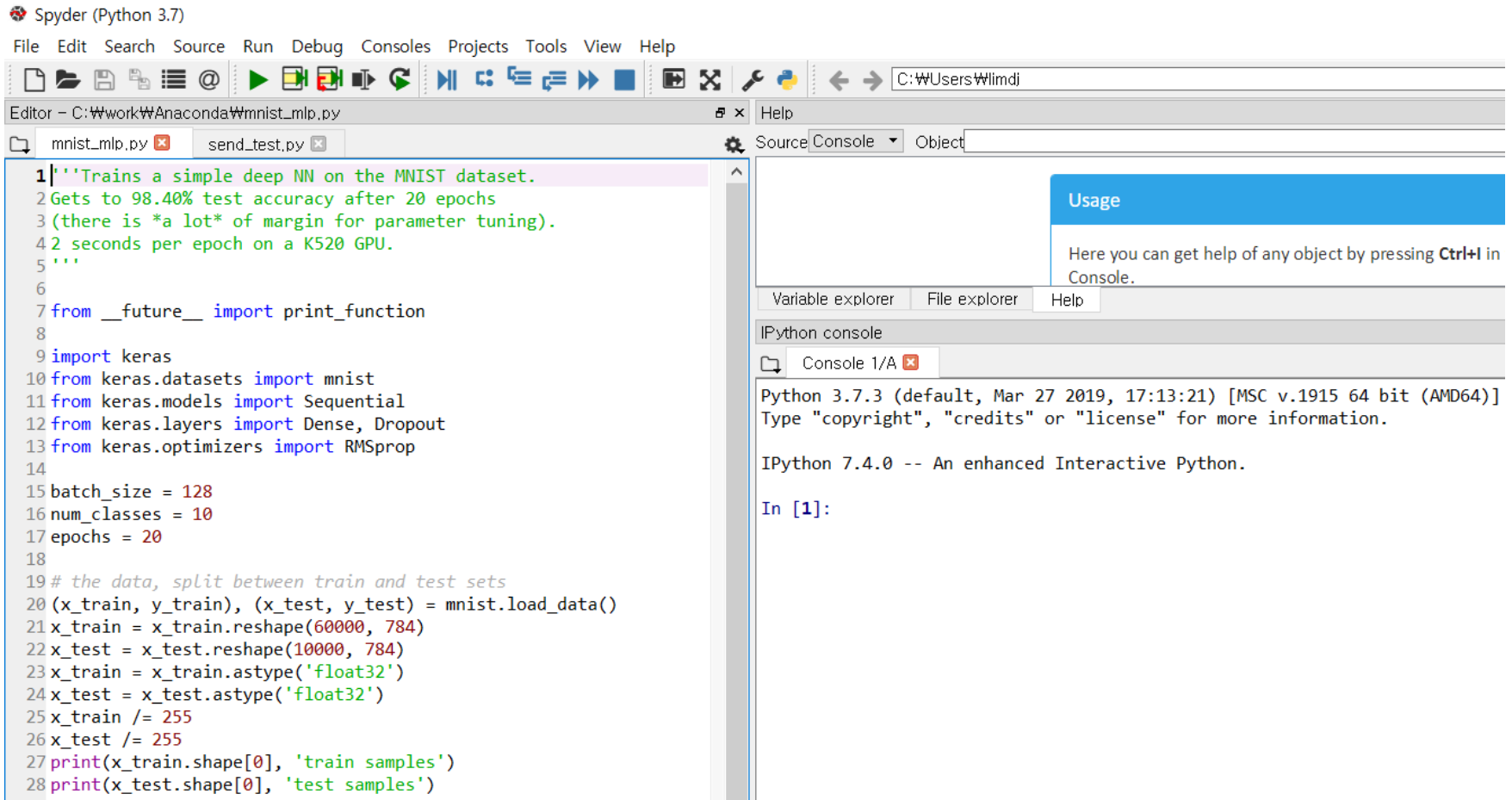


Anaconda : Python Data Science Platform

- Copy mnist_mlp.py, send_test.py to C:\work\Ananconda
- Run Spyder



■ Open mnist_mlp.py



The image shows the Spyder Python IDE interface. The main editor window displays the following Python code:

```
1|'''Trains a simple deep NN on the MNIST dataset.
2|Gets to 98.40% test accuracy after 20 epochs
3|(there is *a lot* of margin for parameter tuning).
4|2 seconds per epoch on a K520 GPU.
5|'''
6|
7|from __future__ import print_function
8|
9|import keras
10|from keras.datasets import mnist
11|from keras.models import Sequential
12|from keras.layers import Dense, Dropout
13|from keras.optimizers import RMSprop
14|
15|batch_size = 128
16|num_classes = 10
17|epochs = 20
18|
19|# the data, split between train and test sets
20|(x_train, y_train), (x_test, y_test) = mnist.load_data()
21|x_train = x_train.reshape(60000, 784)
22|x_test = x_test.reshape(10000, 784)
23|x_train = x_train.astype('float32')
24|x_test = x_test.astype('float32')
25|x_train /= 255
26|x_test /= 255
27|print(x_train.shape[0], 'train samples')
28|print(x_test.shape[0], 'test samples')
--
```

The right-hand side of the IDE shows the IPython console output:

```
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

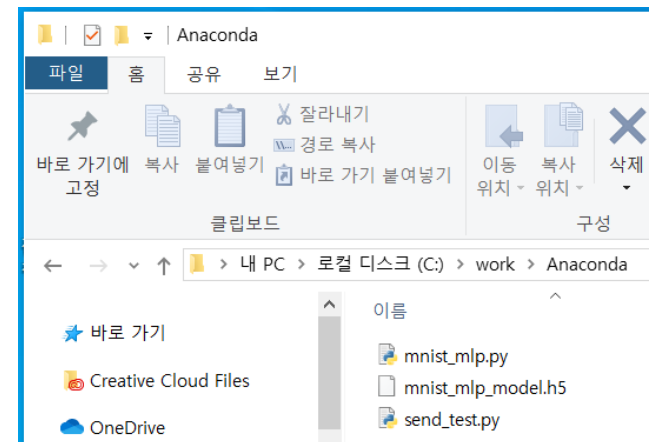
In [1]:
```

Below the console output are tabs for 'Usage', 'Variable explorer', 'File explorer', and 'Help'. The 'Usage' tab is currently active, displaying the text: 'Here you can get help of any object by pressing Ctrl+I in Console.'

■ Saved model: mnist_mlp_model.h5

```
20 print(x_train.shape[0], 'train samples')
29 print(x_test.shape[0], 'test samples')
30
31 # convert class vectors to binary class matrices
32 y_train = keras.utils.to_categorical(y_train, num_classes)
33 y_test = keras.utils.to_categorical(y_test, num_classes)
34
35 model = Sequential()
36 model.add(Dense(512, activation='relu', input_shape=(784,)))
37 model.add(Dropout(0.2))
38 model.add(Dense(512, activation='relu'))
39 model.add(Dropout(0.2))
40 model.add(Dense(num_classes, activation='softmax'))
41
42 model.summary()
43
44 model.compile(loss='categorical_crossentropy',
45               optimizer=RMSprop(),
46               metrics=['accuracy'])
47
48 history = model.fit(x_train, y_train,
49                    batch_size=batch_size,
50                    epochs=epochs,
51                    verbose=1,
52                    validation_data=(x_test, y_test))
53 score = model.evaluate(x_test, y_test, verbose=0)
54 print('Test loss:', score[0])
55 print('Test accuracy:', score[1])
56
57 model.save('mnist_mlp_model.h5')
```

```
Variable explorer | File explorer | Help
IPython console
Console 1/A
Epoch 11/20
60000/60000 [=====] - 6s 92us/step - loss: 0.0265 - acc: 0.9921 - val_loss: 0.0972 - val_acc: 0.9823
Epoch 12/20
60000/60000 [=====] - 6s 92us/step - loss: 0.0248 - acc: 0.9929 - val_loss: 0.1022 - val_acc: 0.9815
Epoch 13/20
60000/60000 [=====] - 5s 91us/step - loss: 0.0255 - acc: 0.9934 - val_loss: 0.0917 - val_acc: 0.9832
Epoch 14/20
60000/60000 [=====] - 5s 92us/step - loss: 0.0228 - acc: 0.9937 - val_loss: 0.1059 - val_acc: 0.9828
Epoch 15/20
60000/60000 [=====] - 6s 92us/step - loss: 0.0209 - acc: 0.9942 - val_loss: 0.0999 - val_acc: 0.9853
Epoch 16/20
60000/60000 [=====] - 6s 92us/step - loss: 0.0203 - acc: 0.9948 - val_loss: 0.1001 - val_acc: 0.9846
Epoch 17/20
60000/60000 [=====] - 5s 91us/step - loss: 0.0202 - acc: 0.9947 - val_loss: 0.1012 - val_acc: 0.9846
Epoch 18/20
60000/60000 [=====] - 5s 92us/step - loss: 0.0183 - acc: 0.9950 - val_loss: 0.0983 - val_acc: 0.9850
Epoch 19/20
60000/60000 [=====] - 5s 91us/step - loss: 0.0195 - acc: 0.9951 - val_loss: 0.1137 - val_acc: 0.9826
Epoch 20/20
60000/60000 [=====] - 6s 92us/step - loss: 0.0187 - acc: 0.9950 - val_loss: 0.0983 - val_acc: 0.9850
Test loss: 0.09829121294636527
Test accuracy: 0.985
In [2]:
IPython console | History log
```



New STM32 Project: ai

IDE STM32 Project

Project Setup
Setup STM32 project

Project Name: ai

Use default location

Location: C:/Users/limdj/STM32CubeIDE/workspace_1.1.0 Browse...

Options

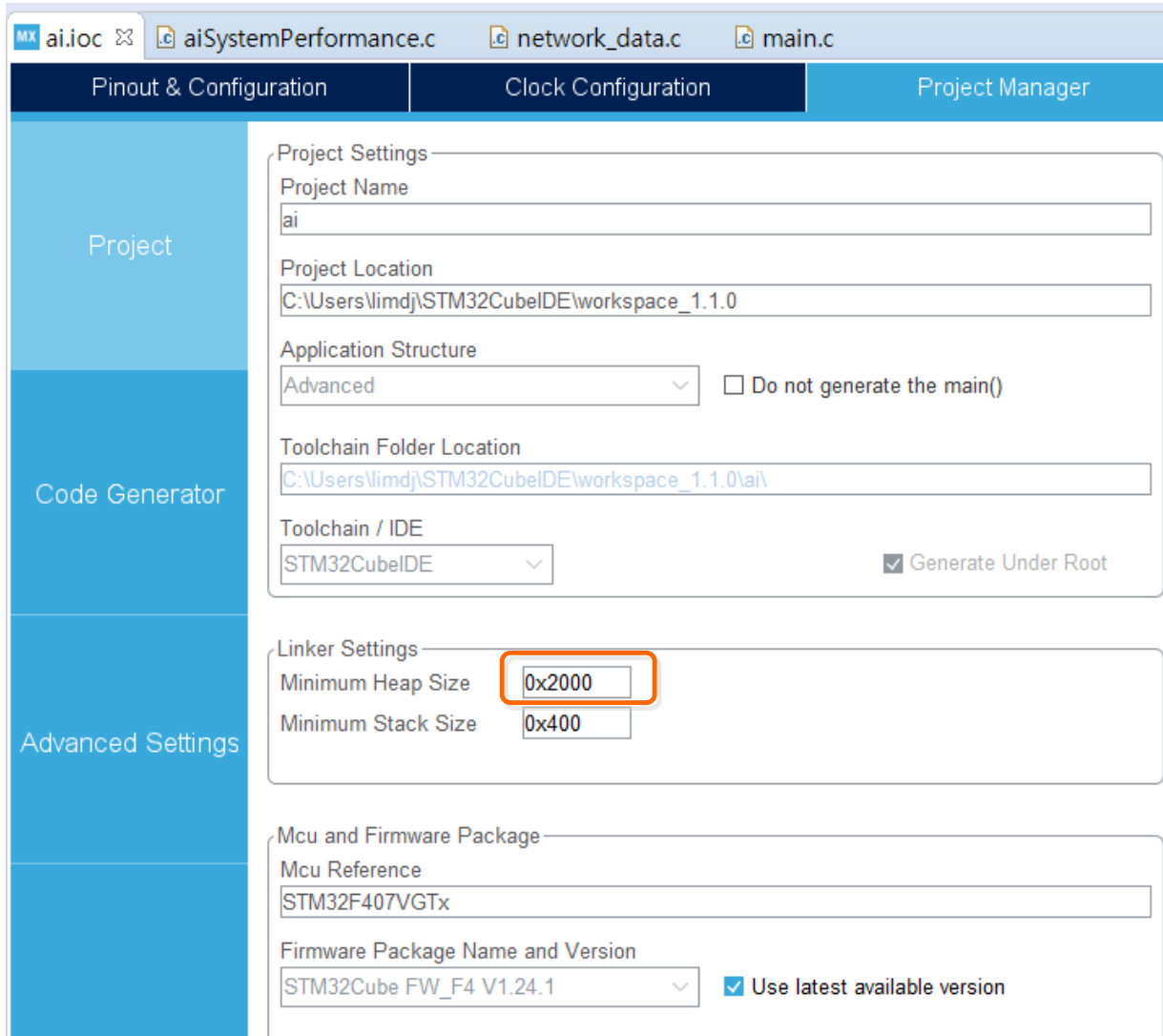
Targeted Language
 C C++

Targeted Binary Type
 Executable Static Library

Targeted Project Type
 STM32Cube Empty

? < Back Next > Finish Cancel

■ Minimum Heap Size: 0x2000



The screenshot displays the STM32CubeIDE Project Manager interface. The top navigation bar includes tabs for 'Pinout & Configuration', 'Clock Configuration', and 'Project Manager'. The left sidebar is divided into four sections: 'Project', 'Code Generator', 'Advanced Settings', and an unlabeled bottom section. The main content area is titled 'Project Settings' and contains the following fields:

- Project Name:** ai
- Project Location:** C:\Users\vimdj\STM32CubeIDE\workspace_1.1.0
- Application Structure:** Advanced (dropdown menu) with a checkbox for 'Do not generate the main()' which is unchecked.
- Toolchain Folder Location:** C:\Users\vimdj\STM32CubeIDE\workspace_1.1.0\ai\
- Toolchain / IDE:** STM32CubeIDE (dropdown menu) with a checked checkbox for 'Generate Under Root'.

Below the Project Settings is the 'Linker Settings' section, which includes:

- Minimum Heap Size:** 0x2000 (highlighted with an orange box)
- Minimum Stack Size:** 0x400

At the bottom is the 'Mcu and Firmware Package' section:

- Mcu Reference:** STM32F407VGTx
- Firmware Package Name and Version:** STM32Cube FW_F4 V1.24.1 (dropdown menu) with a checked checkbox for 'Use latest available version'.

■ Enable USART2

The screenshot displays the STM32CubeMX Pinout & Configuration window. The left sidebar shows a tree view of components, with USART2 selected and highlighted in blue. The main area is titled "USART2 Mode and Configuration" and is divided into two sections: "Mode" and "Configuration".

Mode Section:

- Mode: Asynchronous
- Hardware Flow Control (RS232): Disable

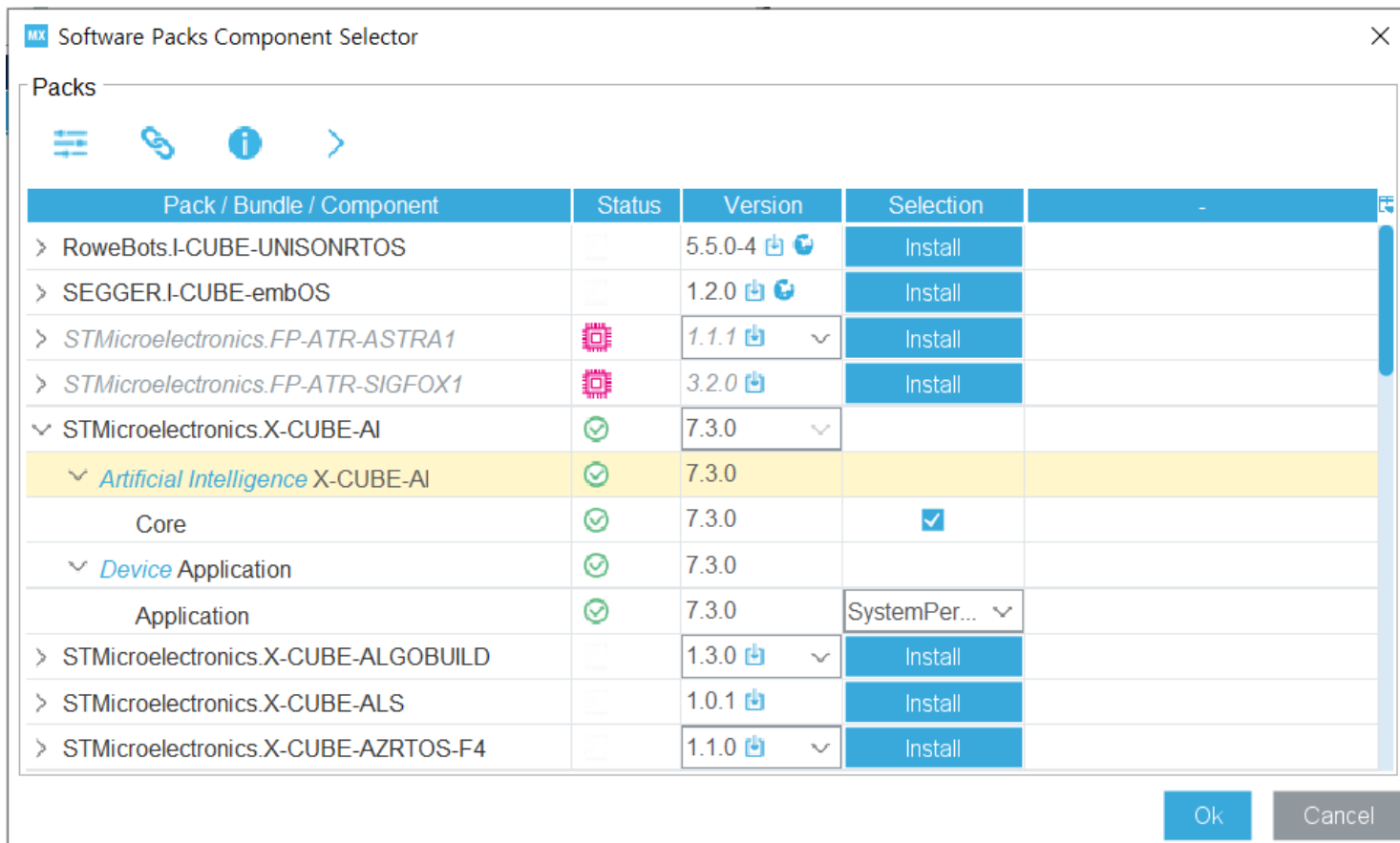
Configuration Section:

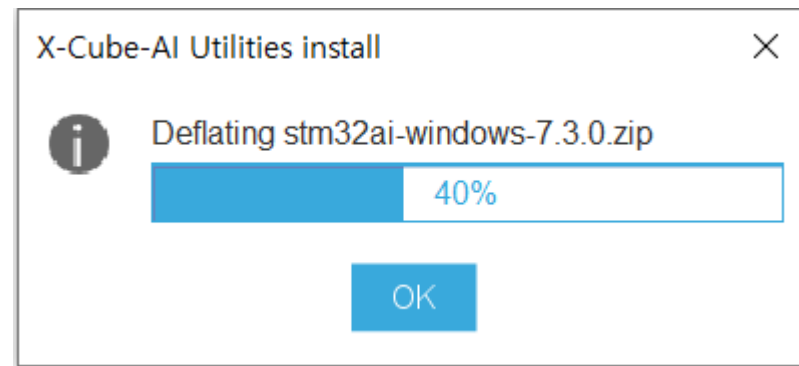
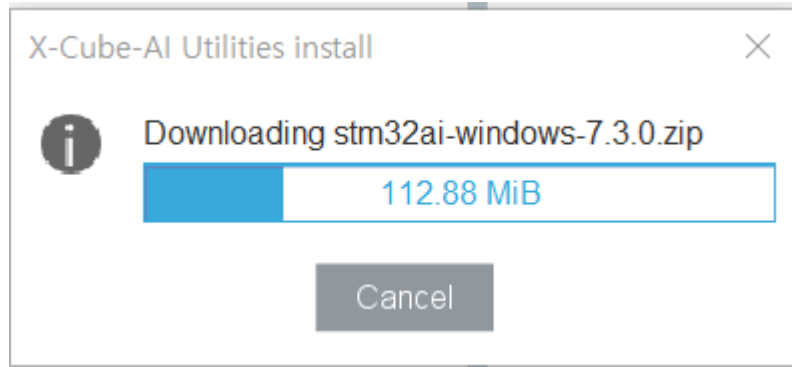
- Reset Configuration button
- Checkmarks for NVIC Settings, DMA Settings, and GPIO Settings.
- Checkmarks for Parameter Settings and User Constants.
- Text: "Configure the below parameters :"
- Search bar: "Search (Ctrl+F)"
- Basic Parameters:
 - Baud Rate: 115200 Bits/s
 - Word Length: 8 Bits (including Parity)
 - Parity: None
 - Stop Bits: 1
- Advanced Parameters:
 - Data Direction: Receive and Transmit

■ Select Components from Software Packs Menu



- Select Application: SystemPerformance
- Select X-CUBE-AI: Core
- Then, Click OK





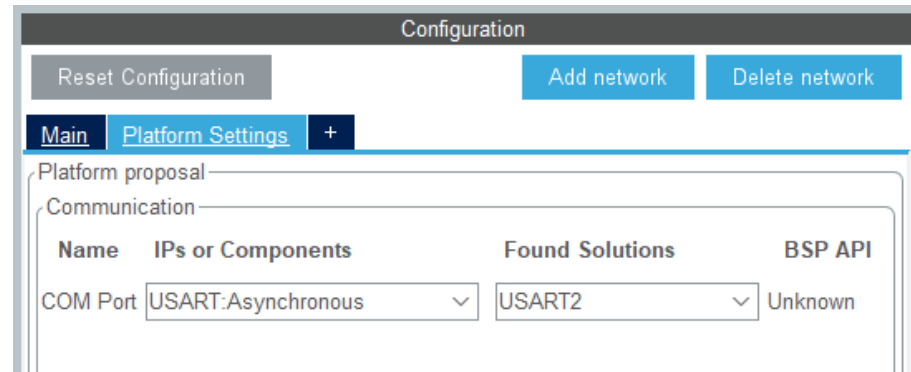
■ Click Software Packs and click

The screenshot displays a software configuration interface with the following elements:

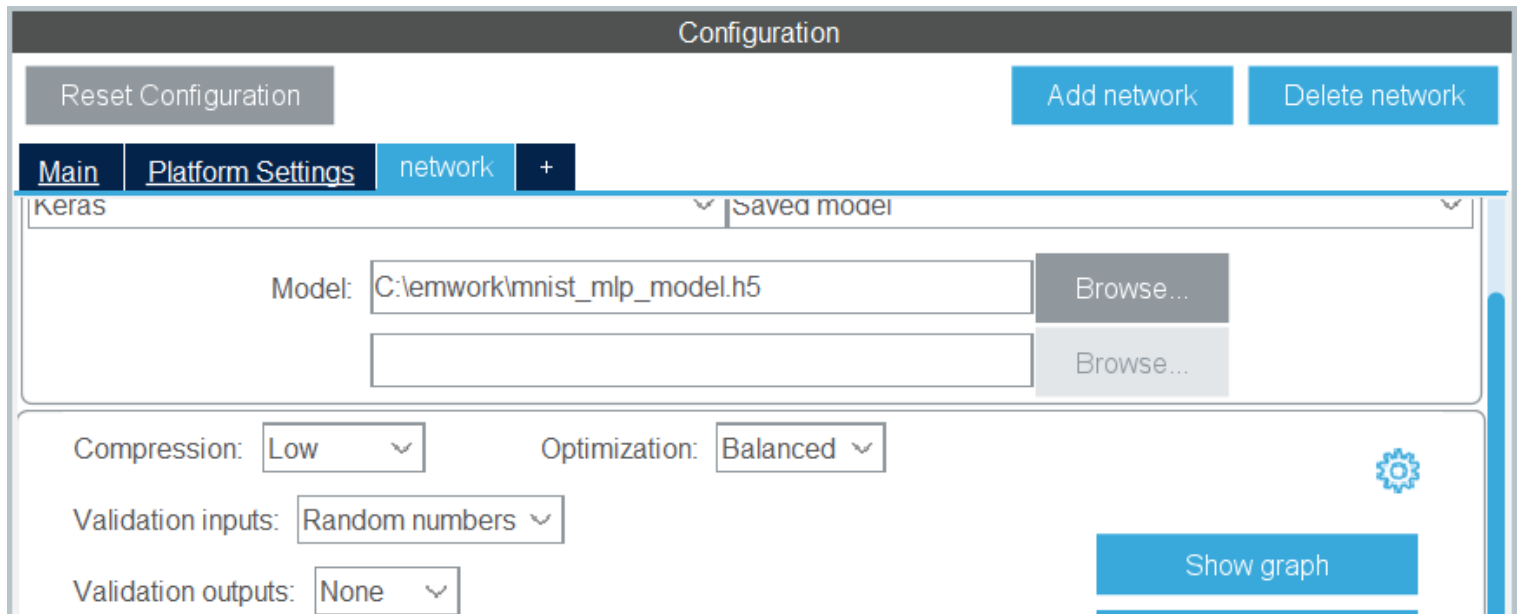
- Navigation Bar:** Pinout & Configuration, Clock Configuration, Project Manage
- Section Headers:** Software Packs, Pinout
- Search and Filter:** Search bar, Categories A->Z
- Left Sidebar (Categories):** System Core, Analog, Timers, Connectivity, Multimedia, Security, Computing, Middleware, Software Packs (selected), STMicroelectronics.X-CUBE-AI...
- Main Content Area:**
 - Mode:** Artificial Intelligence X-CUBE-AI (checked), Artificial Intelligence Application (checked)
 - Configuration:** Reset Configuration, Add network, Delete network
 - Platform Settings:** Main, Platform Settings (+)
 - Platform proposal:** Communication
 - Table:**

Name	IPs or Components	Found Solutions	BSP API
COM Port	Undefined	No solution	Unknown

■ Platform Settings



■ Add network



- Click Analyze and check memory

The screenshot displays a software configuration window titled "Configuration". At the top, there are buttons for "Reset Configuration", "Add network", and "Delete network". Below this is a navigation bar with tabs for "Main", "Platform Settings", "network", and a "+" icon. The "network" tab is active, showing a dropdown menu with "Keras" selected and "Saved model" as a label. The "Model:" field contains the path "C:\lemwork\mnist_mlp_model.h5" and has a "Browse..." button next to it. Below this is another empty field with a "Browse..." button. The configuration section includes "Compression: Low" and "Optimization: Balanced", both with dropdown arrows. "Validation inputs:" is set to "Random numbers" and "Validation outputs:" is set to "None", both with dropdown arrows. A gear icon is visible to the right of these settings. On the right side, there are four stacked buttons: "Show graph", "Analyze" (with a green checkmark icon), "Validate on desktop", and "Validate on target". At the bottom left, the analysis results are displayed: "Complexity: 670880 MACC", "Used Flash: 686.89 KiB (686.89 KiB over 1024.00 KiB Internal)", "Used Ram: 7.12 KiB (7.12 KiB over 192.00 KiB Internal)", "Achieved compression: 3.9", and "Analysis status: done".

■ Validate on desktop

MX Please wait... ×

i Validation on desktop

NOTE: the output of the reference model is used as ground truth/reference value

acc=100.00%, rmse=0.006813521, mae=0.001394981, l2r=0.022370711
10 classes (10 samples)

C0	0
C1	.	0
C2	.	.	6
C3	.	.	.	0
C4	0
C5	0	.	.	.
C6	3	.	.
C7	1	.
C8	0
C9	0

Evaluation report (summary)

Mode	acc	rmse	mae	l2r	tensor
X-cross #1	100.00%	0.006813521	0.001394981	0.022370711	dense_3_nl, ai_float, [(1, 1, 10)], m_id=[4]

X-cross (l2r) #1 error : 2.23707110e-02 (expected to be < 0.01)
Creating txt report file C:\Users\#lmdjw.stm32cubemx\network_validate_report.txt
elapsed time (validate): 10.767s

OK

Generate Code and Build

The screenshot displays the STM32CubeIDE interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows a project named 'ai' with various components like Binaries, Includes, Core, Drivers, Middlewares, USB_HOST, X-CUBE-AI, Debug, and several test files. The main workspace is divided into three panes: Pinout & Configuration, Clock Configuration, and Project Manager. The Pinout & Configuration pane shows a list of categories such as System Core, Analog, Timers, Connectivity, Multimedia, Security, and Computing. The Project Manager pane shows the 'Software Packs' section with 'STMicroelectronics X-CUBE-AI 6.0.0 Mode and Configuration' selected. The 'Configuration' section includes buttons for 'Reset Configuration', 'Add network', and 'Delete network'. The 'Main' tab is active, showing 'Platform Settings' and 'network' sub-tabs. The 'Model' field is set to 'C:\emwork\mnist_mlp_model.h5'. The 'Compression' is set to 4, 'Validation inputs' to 'Random numbers', and 'Validation outputs' to 'None'. The 'Complexity' is 670880 MACC. The Console pane at the bottom shows the output of the build process:

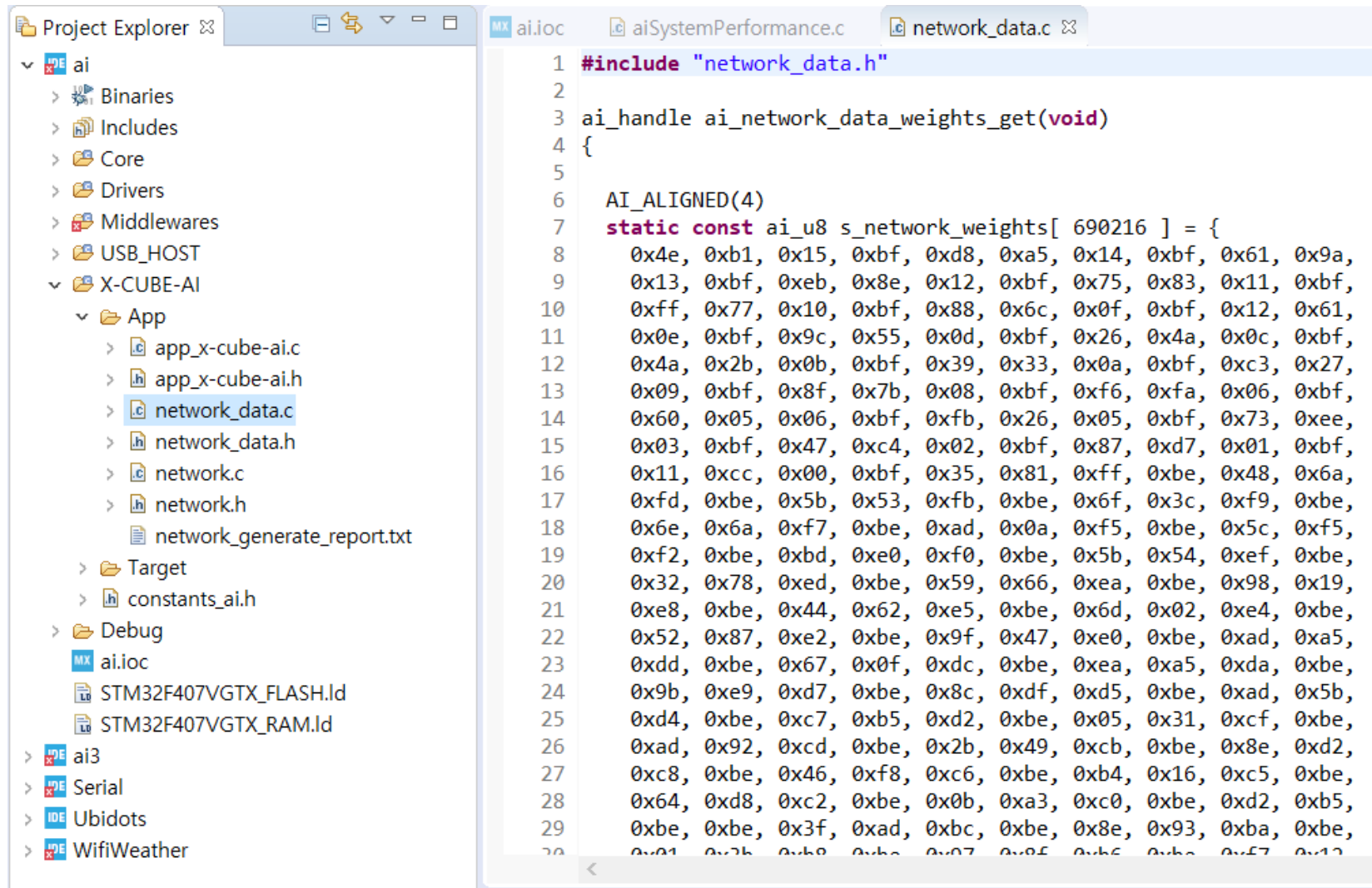
```
CDT Build Console [ai]
arm-none-eabi-size ai.elf
arm-none-eabi-objdump -h -S ai.elf > "ai.list"
arm-none-eabi-objcopy -O binary ai.elf "ai.bin"
text data bss dec hex filename
764040 2224 22128 788392 c07a8 ai.elf
Finished building: default.size.stdout

Finished building: ai.bin

Finished building: ai.list

12:32:26 Build Finished. 0 errors, 0 warnings. (took 16s.305ms)
```

■ Trained weight



The image shows a screenshot of an IDE with a Project Explorer on the left and a code editor on the right. The Project Explorer shows a project named 'ai' with several subfolders and files. The code editor shows the file 'network_data.c' with the following content:

```
1 #include "network_data.h"
2
3 ai_handle ai_network_data_weights_get(void)
4 {
5
6     AI_ALIGNED(4)
7     static const ai_u8 s_network_weights[ 690216 ] = {
8         0x4e, 0xb1, 0x15, 0xbf, 0xd8, 0xa5, 0x14, 0xbf, 0x61, 0x9a,
9         0x13, 0xbf, 0xeb, 0x8e, 0x12, 0xbf, 0x75, 0x83, 0x11, 0xbf,
10        0xff, 0x77, 0x10, 0xbf, 0x88, 0x6c, 0x0f, 0xbf, 0x12, 0x61,
11        0x0e, 0xbf, 0x9c, 0x55, 0x0d, 0xbf, 0x26, 0x4a, 0x0c, 0xbf,
12        0x4a, 0x2b, 0x0b, 0xbf, 0x39, 0x33, 0x0a, 0xbf, 0xc3, 0x27,
13        0x09, 0xbf, 0x8f, 0x7b, 0x08, 0xbf, 0xf6, 0xfa, 0x06, 0xbf,
14        0x60, 0x05, 0x06, 0xbf, 0xfb, 0x26, 0x05, 0xbf, 0x73, 0xee,
15        0x03, 0xbf, 0x47, 0xc4, 0x02, 0xbf, 0x87, 0xd7, 0x01, 0xbf,
16        0x11, 0xcc, 0x00, 0xbf, 0x35, 0x81, 0xff, 0xbe, 0x48, 0x6a,
17        0xfd, 0xbe, 0x5b, 0x53, 0xfb, 0xbe, 0x6f, 0x3c, 0xf9, 0xbe,
18        0x6e, 0x6a, 0xf7, 0xbe, 0xad, 0x0a, 0xf5, 0xbe, 0x5c, 0xf5,
19        0xf2, 0xbe, 0xbd, 0xe0, 0xf0, 0xbe, 0x5b, 0x54, 0xef, 0xbe,
20        0x32, 0x78, 0xed, 0xbe, 0x59, 0x66, 0xea, 0xbe, 0x98, 0x19,
21        0xe8, 0xbe, 0x44, 0x62, 0xe5, 0xbe, 0x6d, 0x02, 0xe4, 0xbe,
22        0x52, 0x87, 0xe2, 0xbe, 0x9f, 0x47, 0xe0, 0xbe, 0xad, 0xa5,
23        0xdd, 0xbe, 0x67, 0x0f, 0xdc, 0xbe, 0xea, 0xa5, 0xda, 0xbe,
24        0x9b, 0xe9, 0xd7, 0xbe, 0x8c, 0xdf, 0xd5, 0xbe, 0xad, 0x5b,
25        0xd4, 0xbe, 0xc7, 0xb5, 0xd2, 0xbe, 0x05, 0x31, 0xcf, 0xbe,
26        0xad, 0x92, 0xcd, 0xbe, 0x2b, 0x49, 0xcb, 0xbe, 0x8e, 0xd2,
27        0xc8, 0xbe, 0x46, 0xf8, 0xc6, 0xbe, 0xb4, 0x16, 0xc5, 0xbe,
28        0x64, 0xd8, 0xc2, 0xbe, 0x0b, 0xa3, 0xc0, 0xbe, 0xd2, 0xb5,
29        0xbe, 0xbe, 0x3f, 0xad, 0xbc, 0xbe, 0x8e, 0x93, 0xba, 0xbe,
30        0x01, 0x7b, 0xb8, 0xbe, 0x07, 0x8f, 0xb6, 0xbe, 0xf7, 0x13
```

- Open aiSystemPerformance.c
- Find from Edit menu: “input tensors” or go to the line 492

workspace_1.4.0 - 407ai/X-CUBE-AI/App/aiSystemPerformance.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

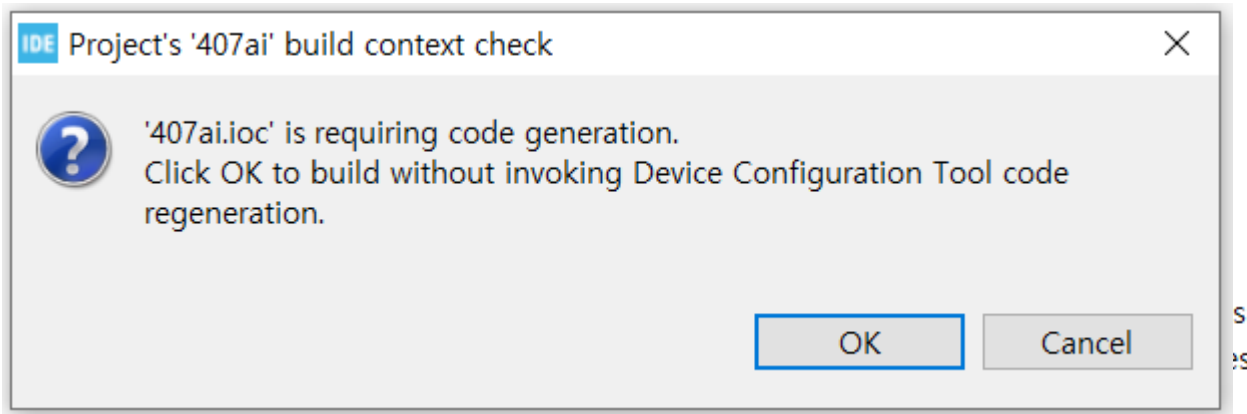
```
521     MON_ALLOC_ENABLE();
522     aiObserverInit(&net_exec_ctx[idx]);
523     observer_heap_sz = MON_ALLOC_MAX_USED();
524 }
525 #endif
526
527     MON_ALLOC_RESET();
528
529     /* Main inference loop */
530     for (iter = 0; iter < niter; iter++) {
531
532         /* Fill input tensors with random data */
533         for (int i = 0; i < net_exec_ctx[idx].report.n_inputs; i++) {
534             unsigned char string[28 * 28][3];
535             ioRawReadBuffer((unsigned char*)string, 28 * 28 * 3);
536             for (ai_size j = 0; j < 28 * 28; j++) {
537                 if (string[j][0] == ' ') string[j][0] = '0';
538                 if (string[j][1] == ' ') string[j][1] = '0';
539             }
540             const ai_buffer_format fmt = AI_BUFFER_FORMAT(&ai_input[i]);
541             ai_i8 *in_data = (ai_i8 *)ai_input[i].data;
542             for (ai_size j = 0; j < AI_BUFFER_SIZE(&ai_input[i]); ++j) {
543                 /* uniform distribution between -1.0 and 1.0 */
544                 //const float v = 2.0f * (ai_float) rand() / (ai_float) RAND_MAX - 1.0f;
```

Modify aiSystemPerformance.c

```
/* Fill input tensors with random data */
for (int i = 0; i < net_exec_ctx[idx].report.n_inputs; i++) {
    unsigned char string[28 * 28][3];
    ioRawReadBuffer((unsigned char*)string, 28 * 28 * 3);
    for (ai_size j = 0; j < 28 * 28; j++) {
        if (string[j][0] == ' ') string[j][0] = '0';
        if (string[j][1] == ' ') string[j][1] = '0';
    }
    const ai_buffer_format fmt = AI_BUFFER_FORMAT(&ai_input[i]);
    ai_i8 *in_data = (ai_i8 *)ai_input[i].data;
    for (ai_size j = 0; j < AI_BUFFER_SIZE(&ai_input[i]); ++j) {
        /* uniform distribution between -1.0 and 1.0 */
        //const float v = 2.0f * (ai_float) rand() / (ai_float) RAND_MAX - 1.0f;
        const float v = (100.0f*(ai_float)(string[j][0] - 0x30) + 10.0f*(ai_float)(string[j][1] - 0x30) +
            (ai_float)(string[j][2] - 0x30)) / 255.0f;
    }

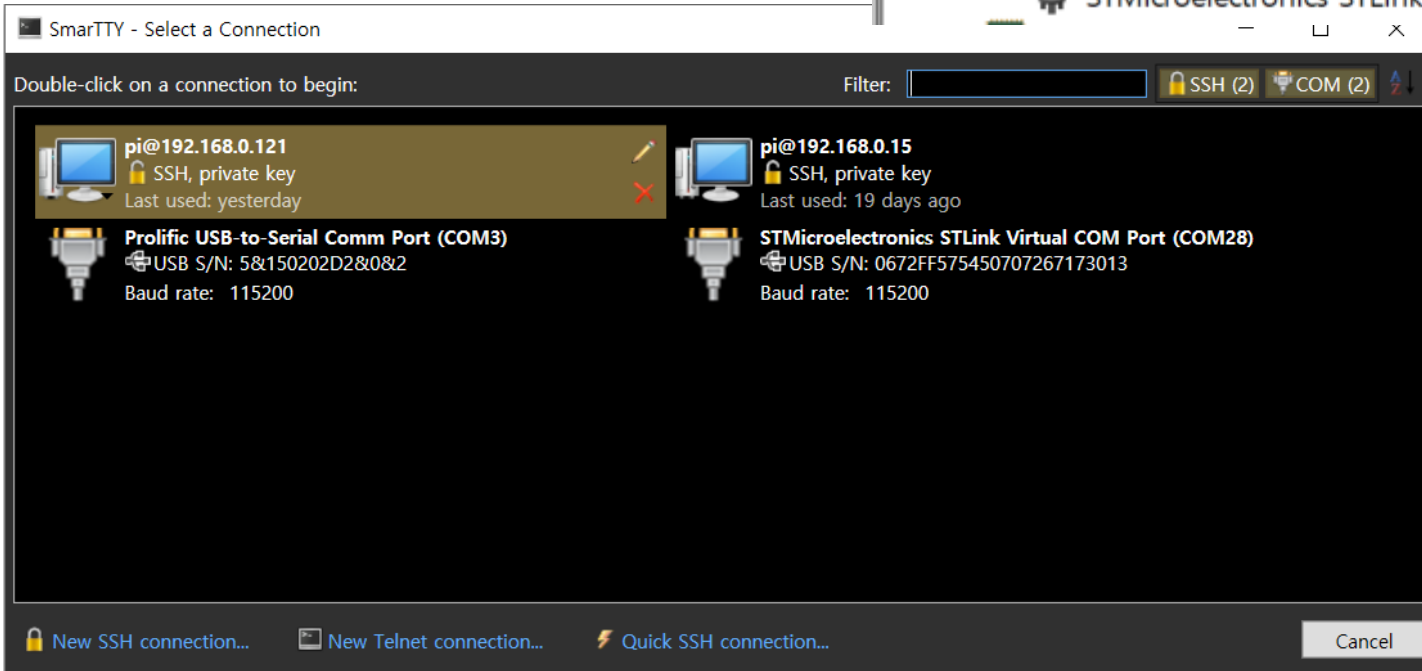
    batch = ai_mnetwork_run(net_exec_ctx[idx].handle, ai_input, ai_output);
    if (batch != 1) {
        aiLogErr(ai_mnetwork_get_error(net_exec_ctx[idx].handle),
            "ai_mnetwork_run");
        break;
    }
    unsigned char recognized_digit;
    ai_float out_data_float[10];
    for (int j = 0; j < 10; j++) out_data_float[j] = *(ai_float *) (ai_output[0].data + j * 4);
    recognized_digit = 0;
    for (int j = 0; j < 10; j++) if (out_data_float[j] > out_data_float[recognized_digit]) recognized_digit = j;
    printf("%d", recognized_digit);
    tend = cyclesCounterEnd();
}
```

- Build and Download to the target board
- Click OK



- > Drivers
- ▼ Middlewares
 - > ST
- > USB_HOST
- ▼ X-CUBE-AI
 - > App
 - > Target
 - > constants_ai.h
- > Debug
 - 407ai Debug.launch
 - 407ai.ioc [Code Generation is required]

- Connect USB-to-serial cable and find COM port number
- Do not open com port



Run send_test.py

- Press RESET button(black button) and run send_test.py
- Change port number

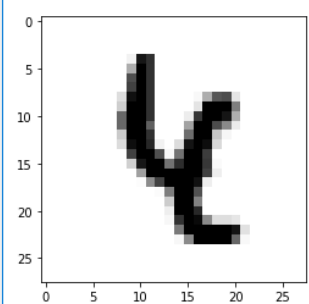
```
Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\work\Anaconda\send_test.py
Wenwork mnist_mlp.py send_test.py - C:\work\Anaconda
2 from keras.datasets import mnist
3 import serial
4
5 port = "COM3"
6 baud = 115200
7
8 ser = serial.Serial(port, baud, timeout=1)
9 # open the serial port
10 if ser.isOpen():
11     print(ser.name + ' is open...')
12
13 # 1. 데이터셋 생성하기
14
15 # 훈련셋과 시험셋 불러오기
16 (x_train, y_train), (x_test, y_test) = mnist.load_data()
17
18 print(x_test.shape)
19
20 for k in list(range(0,16)):
21     digit= x_test[160+k]
22     import matplotlib.pyplot as plt
23     plt.imshow(digit, cmap=plt.cm.binary)
24     plt.show()
25     #print( digit)
26
27     for i in list(range(0,28)):
28         for j in list(range(0,28)):
29             digit_string="{:3d}".format(digit[i][j])
30             ser.write(digit_string.encode('ascii'))
31             #print(digit[i][j])
32
33         out = ser.read(2)
34         print('Recognized digit:',out.decode('utf-8'))
35
36 out = ser.read(2000)
37 print(out.decode('utf-8'))
38
```

Variable explorer File explorer Hel

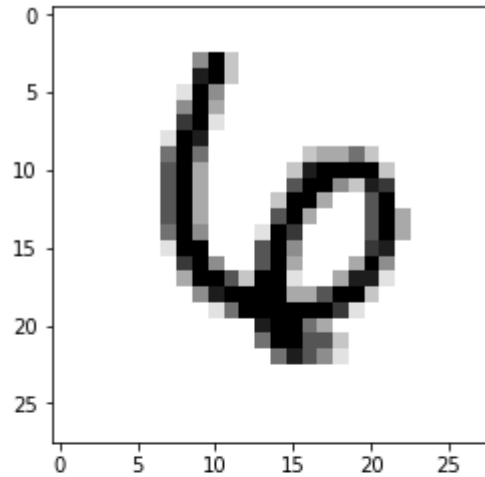
Python console

Console 1/A

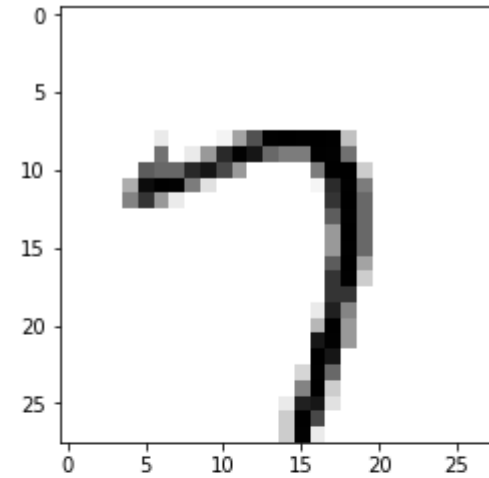
```
In [28]: runfile('C:/work/Anaco
COM3 is open...
(10000, 28, 28)
```



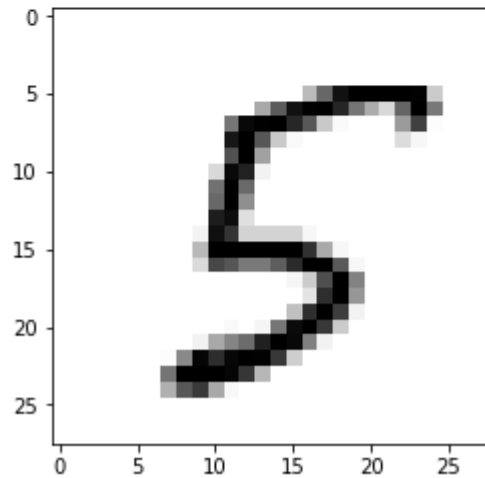
Recognized digit: 4.



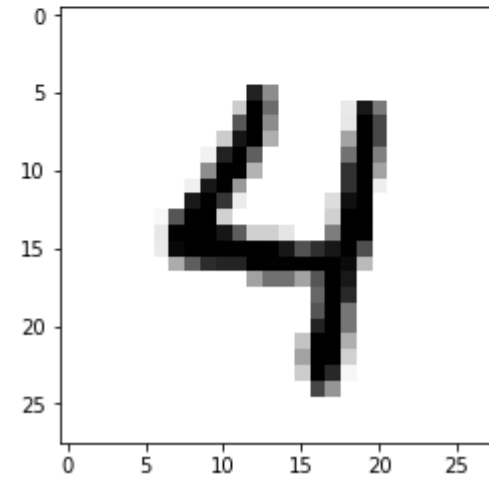
Recognized digit: 6.



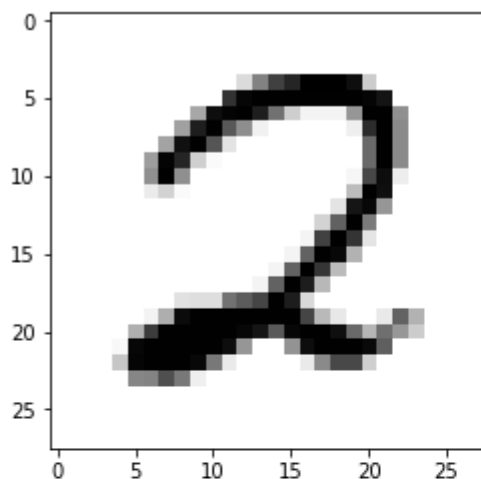
Recognized digit: 7.



Recognized digit: 5.



Recognized digit: 4.



Recognized digit: 2.

```
Results for "network", 16 inferences @168MHz/168MHz (complete)
duration      : 52.216 ms (average)
CPU cycles    : 8772339 -75/+87 (average, -/+ )
CPU Workload  : 5%
cycles/MACC   : 13 (average for all layers)
used stack    : NOT CALCULATED
used heap     : 0:0 0:0 (req:allocated,req:released) cfg=0
```

Exercise

- 주어진 예제를 실행하여 메모리 사용량 및 속도를 검토하고 실제 인식 성능을 확인한다.
- 모델에서 코드 생성 시 컴프레션을 하는 이유는 메모리 사용량을 줄여서 프로그램 메모리에 탑재 가능하도록 하기 위함이다. 컴프레션을 하지 않을 경우 아래 그림과 같이 플래쉬 메모리 용량 초과로 실행이 불가능하게 된다.

Configuration

Reset Configuration Add network Delete network

Main Platform Settings network +

Model inputs

network

Keras Saved model

Model: C:\emwork\mnist_mlp_model.h5 Browse... Browse...

Compression: None Optimization: Balanced

Validation inputs: Random numbers

Validation outputs: None

Complexity: 670880 MACC
Used Flash: 2.57 MiB (2.57 MiB over 1024.00 KiB Internal)
Used Ram: 7.12 KiB (7.12 KiB over 192.00 KiB Internal)

Show graph
Analyze
Validate on desktop
Validate on target

Exercise

- 뉴럴 네트워크 모델에서 뉴런의 숫자를 줄인다면 컴프레션을 하지 않아도 메모리에 탑재가 가능하게 할 수 있다. 뉴런의 개수를 줄여서 컴프레션을 하지 않고도 프로그램 메모리에 탑재할 수 있도록 한다. 이때, 뉴런을 줄이지 않고 컴프레션을 한 경우와, 뉴런을 줄여서 컴프레션을 한 경우의 체감 인식 성능을 비교해 본다.