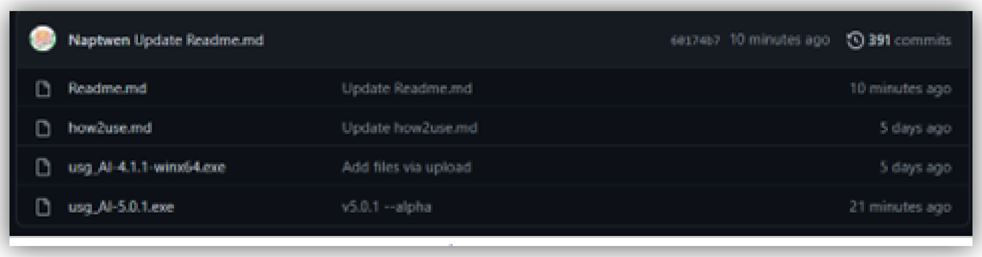
USG AI ENGINE USER GUIDE V6

How to intall

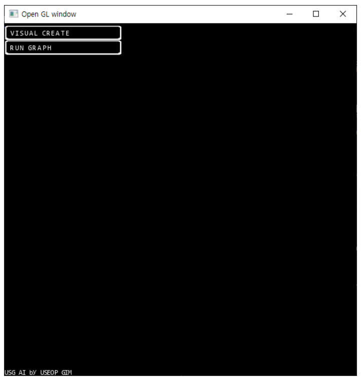
1. For working on the GUI and GPGPU. the pre-installing is required for the OpenGL 3.0 (for GUI interface), OpenCV 2.0 (for testing), OpenCL 2.0 (for gpgpu calcualtion)
2. Access the https://github.com/Naptwen/usgAI then download the newest version



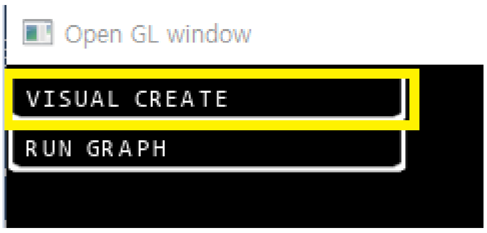
Naptwen Update Readme.md 6817487 10 minutes ago 391 commits

Readme.md	Update Readme.md	10 minutes ago
how2use.md	Update how2use.md	5 days ago
usg_AI-4.1.1-winx64.exe	Add files via upload	5 days ago
usg_AI-5.0.1.exe	v5.0.1 --alpha	21 minutes ago

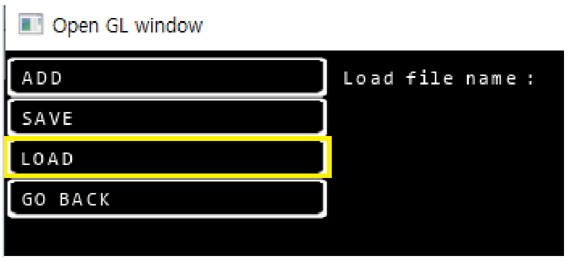
3. After installing for the test, double click the usg_AI.exe file

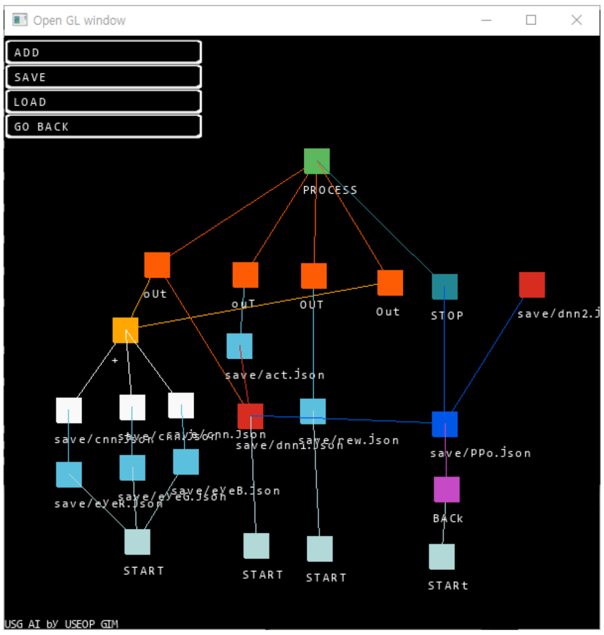


4. For testing, click the VISUAL CREATE button



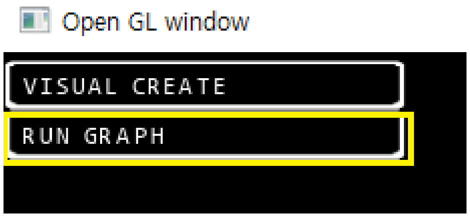
5. Next click the LOAD button then type "save/test.text'"



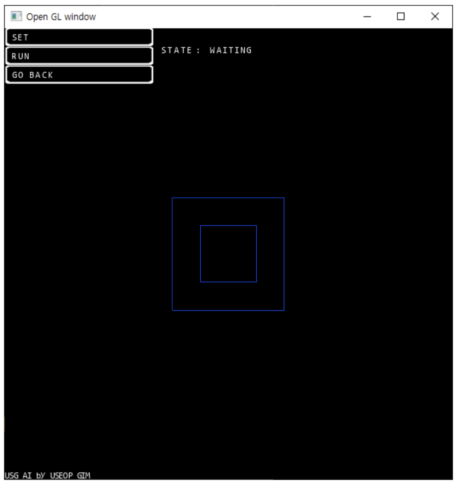


- If you can't see like above, please re-installing the file or checking the save folder in usg_AI

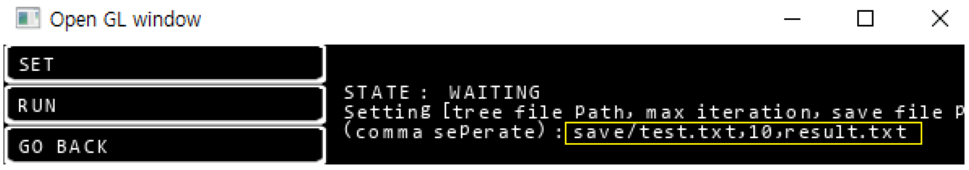
7. Click the GOBACK button then click the RUN GRAPH



8. In the RUN GRAPH screen, click the SET button



9. After click the SET button type “save/test.txt, 10, result.txt”

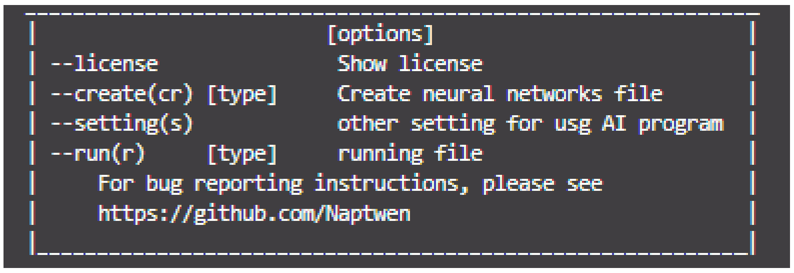


```
STATE : WAITING
Setting [tree file Path, max iteration, save file P
(comma sePerate) : save/test.txt,10,result.txt
```

10. Now click the RUN button
11. If the center double square is moving, it means that the programe is running.
12. If the moving is stopped, you can exit the program

Console Interface User Guide

Although the console interface is provided, you can write down by other third party text edit program. In this case, please reference the format that already in the save file.

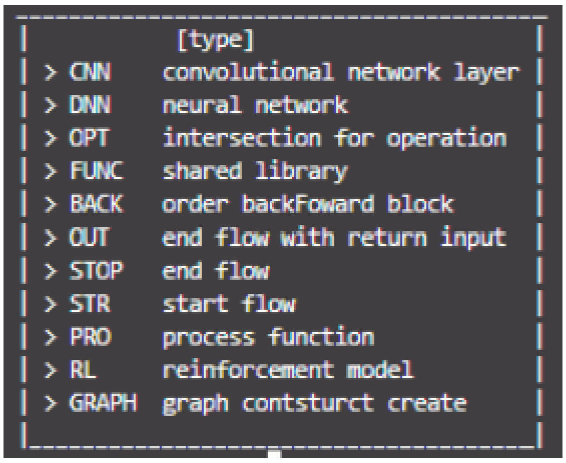


```
 -------------------------------------------------------------
|                         [options]                           |
| --license                 Show license                      |
| --create(cr) [type]       Create neural networks file       |
| --setting(s)              other setting for usg AI program  |
| --run(r)     [type]       running file                      |
|     For bug reporting instructions, please see              |
|     https://github.com/Naptwen                              |
|_____________________________________________________________|
```

--help : show the basic options
--license : show license
--create “type”: writing the model using console interface
--setting : setting the usg_AI_setting.txt file
- **gpgpuon(off)** : on and off the gpgpu (OpenCL3.0)
- **verboseon(off)** : on and off the detail for the runtime message for the program
- **multion(off)** : on and off the multi thread

--run “graph file path” “max iteration” “save file name” : running the program

TYPE

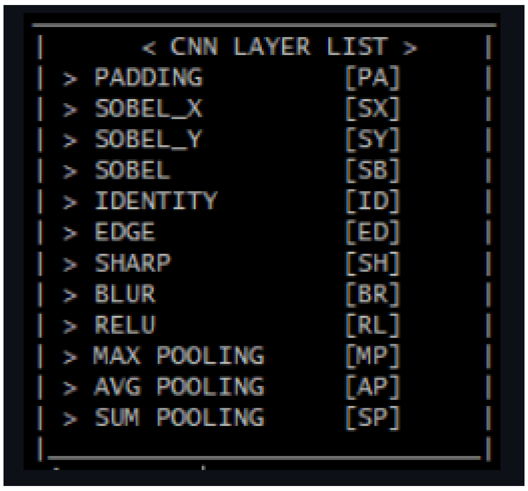


```
----------------------------------------
|            [type]                    |
| > CNN    convolutional network layer |
| > DNN    neural network              |
| > OPT    intersection for operation  |
| > FUNC   shared library              |
| > BACK   order backFoward block      |
| > OUT    end flow with return input  |
| > STOP   end flow                    |
| > STR    start flow                  |
| > PRO    process function            |
| > RL     reinforcement model         |
| > GRAPH  graph contsturct create     |
|______________________________________|
```

CNN : Convolutional Neural Network node
DNN: Deep Neural Network node
OPT : Operation node (version 5 provides only sum and multiplication)
FUNC : The shared library node (shared library node is a extra function for runtime)
BACK : Send backward message to the connected parents Model
OUT: The end of one cycle of flow graph node, it return the input value to parents
STOP : The end of one cycle of flow graph node, it doesn’t return the input value to parents
START: The start of one cycle of flow graph node
PRO: The control the cycle(process) of the flow graph node
RL: The reinforcement model node
RNN : The RNN model node
GRAPH: The graph flow chart file(not recommend for edit in CLI, please use GUI program)

You don’t actually write the file OPT, BACK, OUT, STOP, START, if you type the name on the GUI, it automatically converts the node to the right type.

CNN OPTIONS



```
|       < CNN LAYER LIST >        |
| > PADDING               [PA]    |
| > SOBEL_X               [SX]    |
| > SOBEL_Y               [SY]    |
| > SOBEL                 [SB]    |
| > IDENTITY              [ID]    |
| > EDGE                  [ED]    |
| > SHARP                 [SH]    |
| > BLUR                  [BR]    |
| > RELU                  [RL]    |
| > MAX POOLING           [MP]    |
| > AVG POOLING           [AP]    |
| > SUM POOLING           [SP]    |
```

- *All paddings are size 1*

  PADDING : padding the image basic setting is zero padding.
- *All kernels are 3 by 3 with stride 1 (v5.x.x version don't allow to change it)*

  SOBEL_X : Sobel mask X axis basic setting

  SOBEL_Y : Sobel mask Y axis basic setting

  IDENTITY : Identity mask

  EDGE : Edge mask

  SHARP : Sharp mask

  BLUR : Blur mask
- *All poolings are 2 by 2 with stride 2*

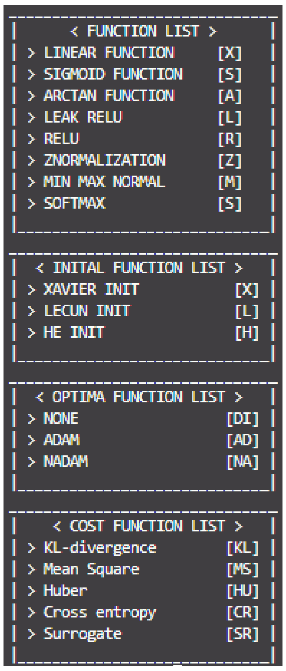
  MAX POOLING : max pooling

  AVG POOLING: average pooling

  SUM POOLING : summation pooling

** When the allocated image by stride size is not enough the allocated part is discarded.*

Neural Network options



```
 _____________________________
|      < FUNCTION LIST >      |
| > LINEAR FUNCTION     [X]   |
| > SIGMOID FUNCTION    [S]   |
| > ARCTAN FUNCTION     [A]   |
| > LEAK RELU           [L]   |
| > RELU                [R]   |
| > ZNORMALIZATION      [Z]   |
| > MIN MAX NORMAL      [M]   |
| > SOFTMAX             [S]   |
|_____________________________|
 _____________________________
|  < INITAL FUNCTION LIST >   |
| > XAVIER INIT           [X] |
| > LECUN INIT            [L] |
| > HE INIT               [H] |
|_____________________________|
 _____________________________
|  < OPTIMA FUNCTION LIST >   |
| > NONE                 [DI] |
| > ADAM                 [AD] |
| > NADAM                [NA] |
|_____________________________|
 _____________________________
|    < COST FUNCTION LIST >   |
| > KL-divergence        [KL] |
| > Mean Square          [MS] |
| > Huber                [HU] |
| > Cross entropy        [CR] |
| > Surrogate            [SR] |
|_____________________________|
```

Active function and normalization function

LINEAR FUNCTION : f(x) = x

SIGOMID FUNCTION : sigomid (x)

ARCTAN FUNCTION : arctan(x)

LEAK RELU : f(x) = max(x,0.3)

RELU : f(x) = max(x,0)

ZNORMALIZATION : z-norm function

MIN MAX NORMAL : min max norm function if min == max it return 0

SOFTMAX : softmax(x)

Initial function

XAVIER INIT : xavier uniformal initialization

LECUN INIT : lecun uniformal initialization

HE INIT : He uniformal intialization

Optimization

NONE : no optimization for back propagation

ADAM : adam optimization for back propagation

NADAM : Nestrov adam optimization for back propagation

Cost function

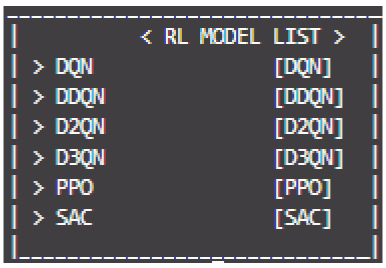
KL-divergence : KL divergence cost function

Mean square : mean square with power of 2 function

Surrogate : surrogate cost function (it means depends on the model's cost evaluation)

HUBER : Huber cost function the |delta| = 1

Cross Entropy : Cross entropy cost function



```
 ________________________________
|         < RL MODEL LIST >      |
| > DQN               [DQN]      |
| > DDQN              [DDQN]     |
| > D2QN              [D2QN]     |
| > D3QN              [D3QN]     |
| > PPO               [PPO]      |
| > SAC               [SAC]      |
|________________________________|
```

DQN: It requires 2 neural moddels with output 1 for both Q model

DDQN : It requires 2 neural moddels with output 1 for both Q model
and the size of input of target Q is state + 1

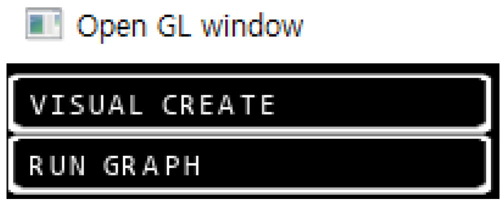
D2QN : It requires 2 neural moddels with output 2 for both Q model

D3QN : It requires 2 neural moddels with output 2 for both Q model
and the size of input of target Q is state + 1

PPO: It requires 2 neural moddels, one is policy model and other is Q model

SAC: It requires 3 neurla modles and main model is policy and others are Q model

GUI User Guide



VISUAL CREATE : Creating the graphical flow chart
RUN GRAPH : Running the graphical flow chart

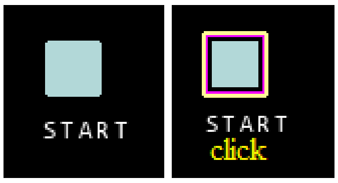
VISUAL CREATE GUI Guide

First of all, the node is the unit for saving and running each algorithm
This program's one of main point is that using the connections of nodes for easy to create the machine learning algorithm.

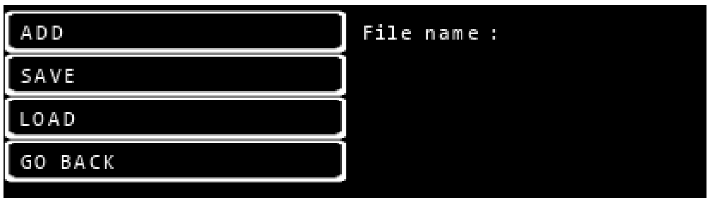
ADD : Adding a new node
SAVE : Save current selected node and its child nodes
LOAD : Loading the graph flow chart

Guide for using graphical interface

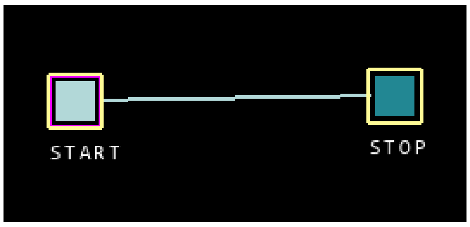
1. When click the ADD button, empty node is created.
   The total node is limited in 2^32-1
2. When click the node, the selected node is highlighted.



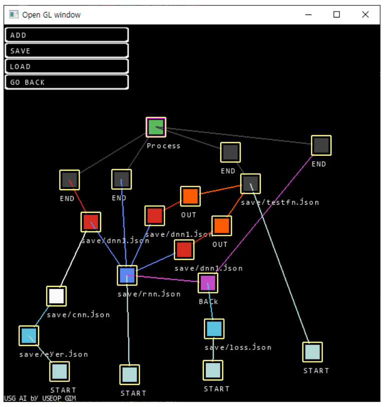
3. When right click the node, you can link the node with file by file name



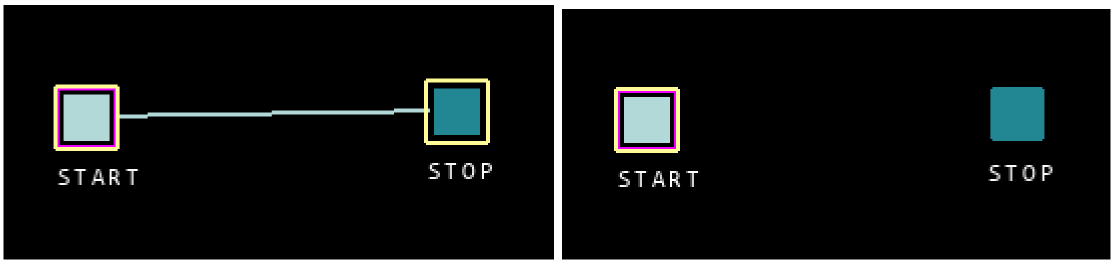
4. From selected node to another, the first node be child and the second be parent for the selected node



5. When click the connected node, it also highlighted the selected node and all others.

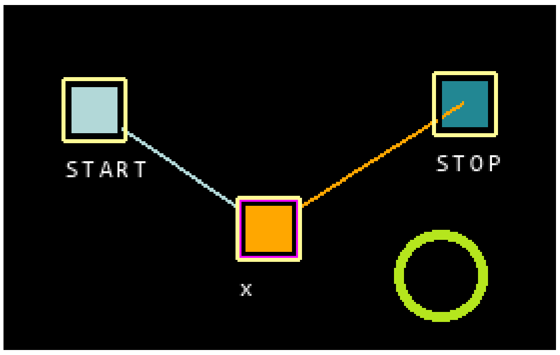


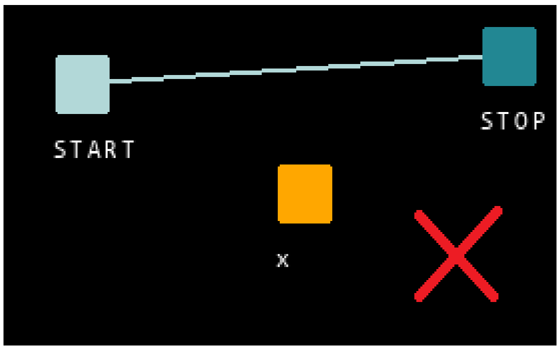
6. When double click the selected node, its connections are removed.



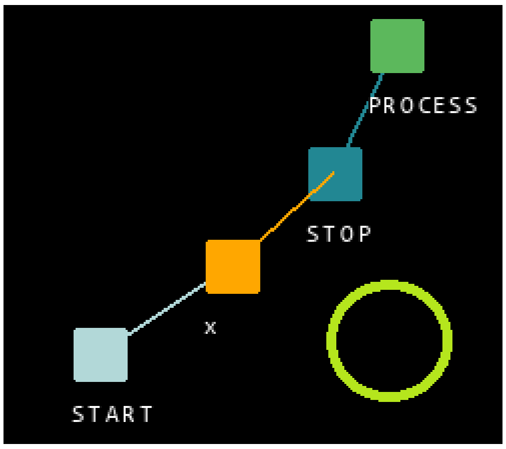
User Guide for connection

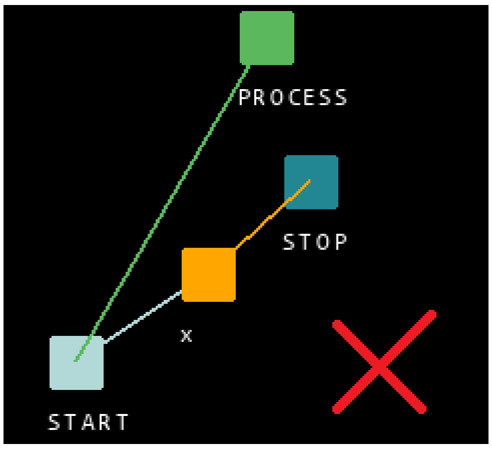
1. The START and END node must be exist at the end of both side in the node that want to be run.



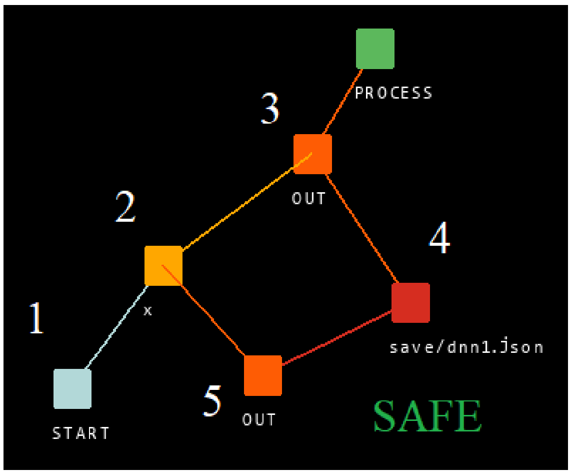


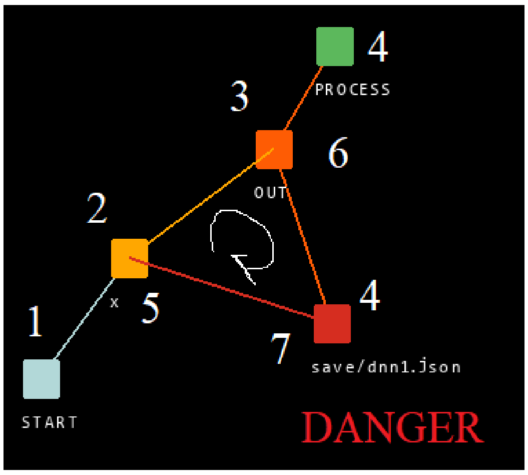
2. The PROCESS node must have the END node as child.



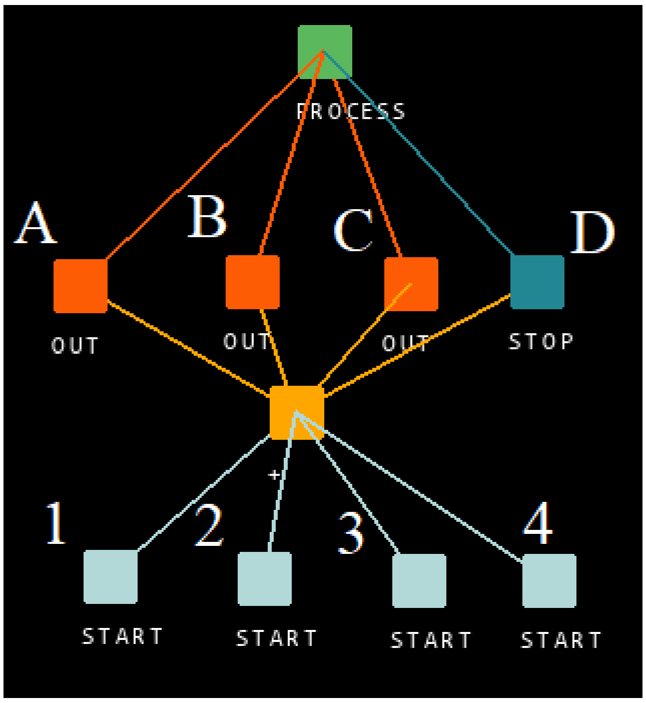


3. After V6.0.0 can make a recursive node but please does not forget to make a break point for that if not the program infinitely loops.



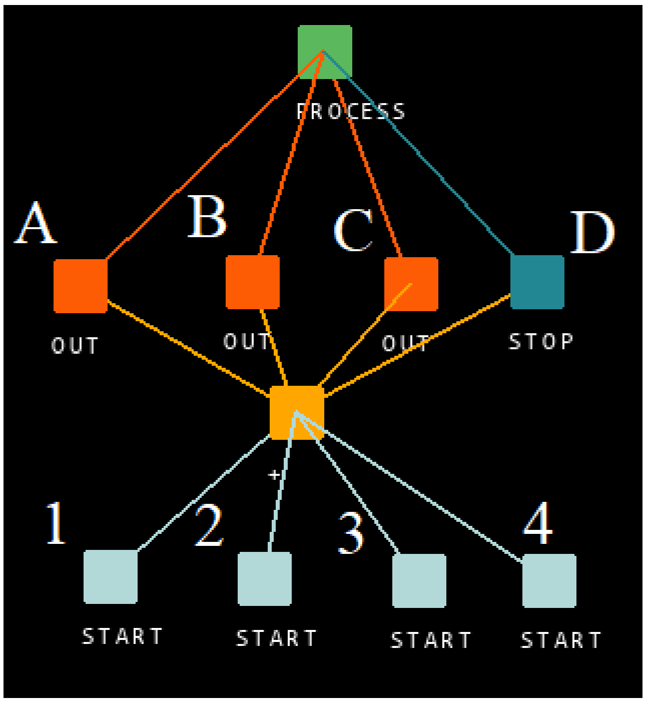


4. There are unlimited for the number of child and parents



When save the node, all connected node are saved together but not disconnected node

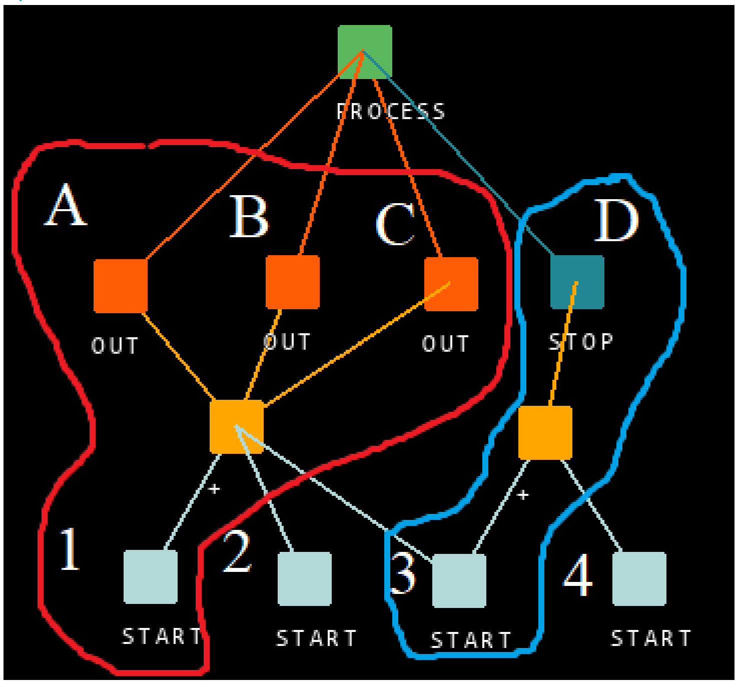
5. The order of running the program is base on ***the most left*** OUT or STOP node, connecting PROCESS node.



For example, the first start process is node is A ->B->C->D

6. As same as above, the order of the START node is also the most left node 1->2->3->4
However, in the above case, all start node is connected to the A,B,C,D
Therefore, in the start node, only number 1 START node is working but 2,3,4 are not.

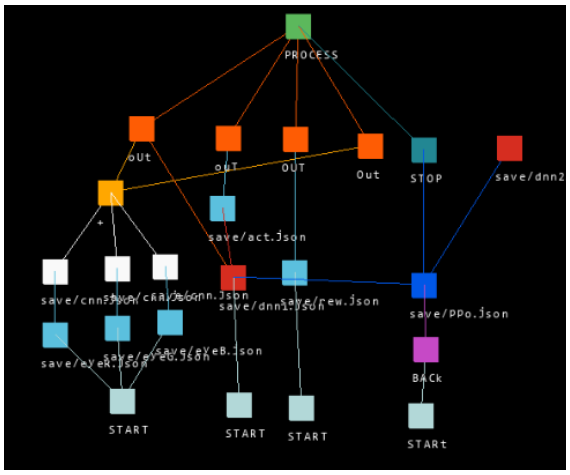
For example, see below

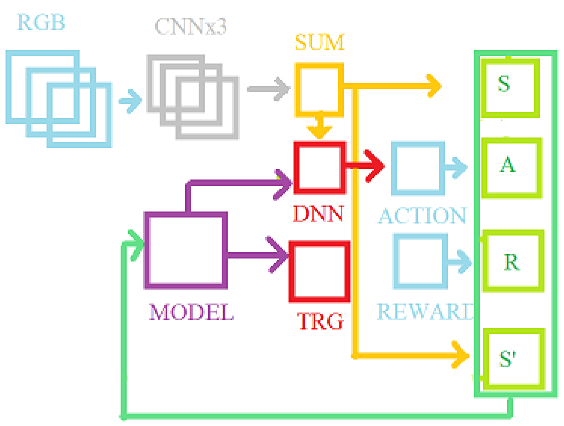


The order of process is A,B,C,and D then the START node for A,B and C is 1 but D is 3 Therefore the order of process is 1,1,1,3

How to make Reinforcement node

The reinforcement node must have a single PROCESS node with four END node and multiple neural network node (the number of neural network node depeneds on the reinforcement learning model)





From the left hand side

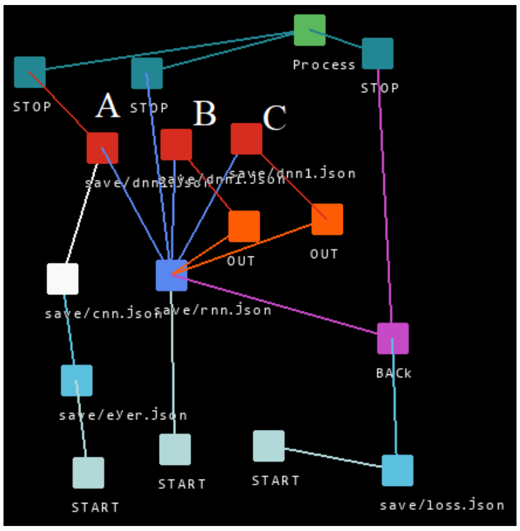
- OUT [1] : the first end node recieve the enviroment state.
- OUT [2] : the second end node recieve the action state.
- OUT [3] : the third end node recieve the reward state.
- OUT [4] : the last end node recieve the enviroment state after the action.
- STOP : The most left side neural network node be the main agent node for the learning, action(or policy) in model. In other words, it is the main neural network decides the action.
-

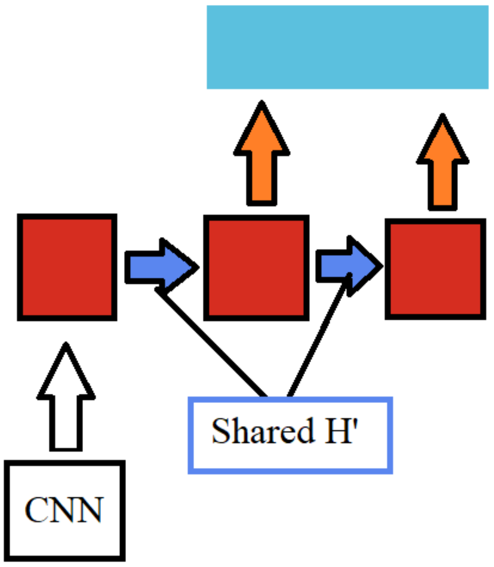
More detail of structure.

1. Get the display area pixel RGB data seperately by eyeR,eyeG, eyeB shared lib
2. Then CNN kenerl and distortion the image then merge it by + operation
3. Send the image to dnn1 and State node
4. using the pre pixel data dnn1 decide the action (it reduce the calcualtion time as directly get from first state)
5. Send the dnn1 output data to the action shared lib.
6. Get reward by reward shared lib
7. Get new state again (we reusing the same propation, we just connect from +
(don't worry about the overlapping it overwrite whenever start from the same start node)
8. Now by PPO add replay buffer and if the replay buffer is enough it calculate cost and update dnn1 and dnn2.

How to make RNN style

version 6 RNN model looks complicated because I tried to make it as be generalized as much as. However, it is simple as you thought, the left graph is the same as the right model.





version 6 recursive backpropagation for the RNN is working like below

1. the dnn1 get value from the CNN whose input is eyer.json
2. Next the process order to RNN working
3. RNN give the its weight times the linear output value of the most left dnn 1 (A) to the second dnn 1 (B).
4. Then B sums the values then do the forwarding.
5. After B forwarding it saves the out value to the RNN.
6. RNN using the B does the same propagation as B to C.
7. After C forwarding it saves the out value to the RNN.
8. Order the Lossfn get the true value(target value).
9. At the BACK node, get the true value from the Lossfn node then order to the RNN do backward.
10. From RNN, it calculates the loss from the saved values in the RNN compared with the input value in the BACK node in reverse order.

    1. Using the C output value calculate the cost value with the true value in the BACK node.

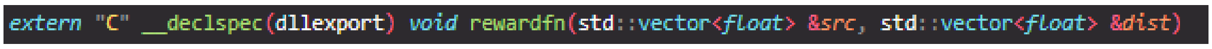
    2. Step by step do each layer do the following equations.

        a. 1. C does the backpropagation without the previous input value which is saved in RNN.
        b. It means subtracting the RNN weight time previous values.
        c. 2. Then do another backpropagation with the previous input value which is saved in RNN.
        d. 3. Update both the C layer and the RNN layer.
        e. At this time, the C layer updates the partial differential for the output value from the active function to deriver the difference in the cost function.
        f. But, the RNN updates only its weight layer since its summed value is dependent on the previous value.

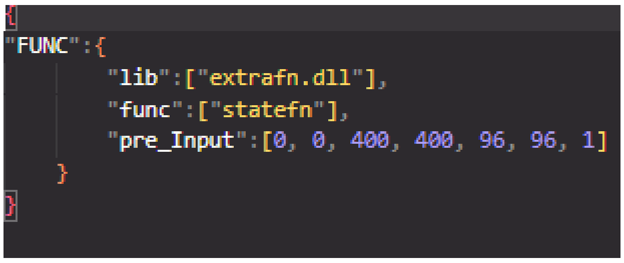
    3. Do the same propagation for the B.

    4. A doesn't have the saved output value for the RNN. So, its weight update is based on the cost value of B.

Shared library shape



```
extern "C" __declspec(dllexport) void rewardfn(std::vector<float> &src, std::vector<float> &dist)
```



```
{
"FUNC":{
        "lib":["extrafn.dll"],
        "func":["statefn"],
        "pre_Input":[0, 0, 400, 400, 96, 96, 1]
    }
}
```

1. **FUNC :** The node name
2. **lib :** shared library path
3. **func** : the specific function of shared library
4. **pre_Input:** the intial value for current function (if there no input for this node).

Please see the src/extrafn.cpp file and save/state.json, act.json and rew.json

Q & A

What is pre Input?

The preInput value is the initial value and if it dosen't have children, it keep that values until the program exit. Forexampel save/state.json has 7 pre vlauese, 1,2 are displace position 3,4 are capture display size, 5,6 are the resize for display, 7 is rgb channel option.

Why the test is so slow?

It is not the algorithm problem the problem caused from extrafn shared library DLL file.

I wrote a very simple shared lib file for just testing the game, so don't use the test dll file for the real problems it is just for testing the program.

As referencing the form of the 'src/extrafn.cpp' resource file makes your own reward, action, and state functions.

As for giving a tip, most of taking time part is the OpenCV part and interface input part, if you directly get those data from your own program, the speed dramatically increases.

Only JSON file format allowed?

No!, I just wrote the file format as JSON, it is not required. you can change the extension name to anything but please keep the format.

Another tip Neural Network weight and bias value are based on base64 by Nyffenegger rene.nyffenegger@adp-gmbh.ch.

Why base64?

I have planned for multiplex socket TCP/IP code and also made it in the code but it is not used now. For that, It was preparation for networking.

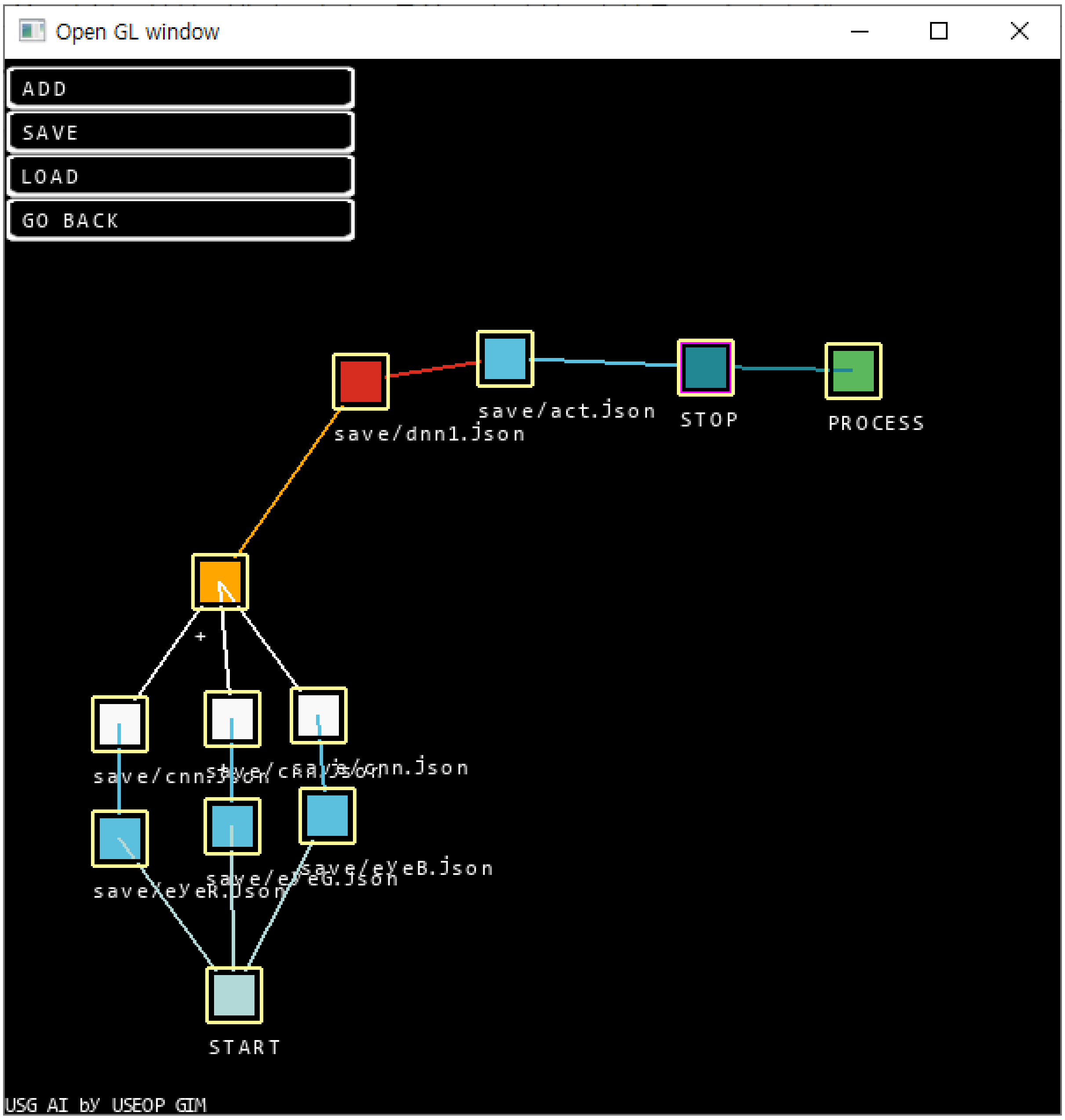
Program unexpectedly shut down what can I do?

If you don't have an idea please open the terminal then type '*usg_AI --setting verboseon*'

It shows every detail of the algorithm process.

On GUI, the node doesn't move and can't click!
Press Enter twice.

Can I use the graphical method for other programs?
Yes, you can make not only a Machine learning algorithm but also anything by the FUNC shared library node.
Here is the example simulation program.



1. PROCESS order to START block
2. START block order to run eyeR, G,B node
3. The eye R,G,B run the program and then give it the values for the each cnn node
4. Each cnn gives values to the operation node +
5. The operation node + sum of the input values then give to the dnn1 node
6. The dnn1 node gives it to the act node

7. The act node do the action then stop at stop node