Guide

Two main parts here are the neural network and the electronics part. The code for both these parts and files of the UI are present in this repo - https://github.com/ayvak/PDP-Plexus.

The Neural Network and UI

All the files for the neural network (NN) are in the directory `PDP-Plexus/NN`. The NN is already trained and the model is in the directory `PDP-Plexus/NN/PDP-NN`. The UI and NN are integrated together along with the the server.

Installation

If this is to be installed in another system, the directory `PDP-Plexus/NN` has to be copied to that system where Python is installed. Below are the packages to be installed. If version numbers are specified along the packages, that particular version has to be installed. Latest versions of other packages can be installed.

- Python - 3.8.2
- Tensorflow - 2.4
- Gradio - 1.5.3
- Scikit-learn
- Scipy
- PySerialTransfer
- Opencv2
- Matplotlib
- Numpy

UI changes

We have changed the default UI of the Gradio interface. Corresponding files are present in the attached zip file. This zip file is the compressed Gradio library. After installing the Gradio library, the static folder of the library with the one inside the zip file (at location `gradio/static`). The installation location of library can be found using the command `pip show gradio`. In case gradio is updated, the `static` folder has to be replaced with the given files.

Note:

1. It is required to give a manual IP for the system as we have to SSH to it to start and stop the NN server.
2. We have created a Python environment and installed packages in the Mac Mini. It is not necessary to do that if this is replicating in another system.

Starting the server and NN

Since the computer is inside the structure, we have to SSH to the system before any operations. Start a screen session and start the NN server in it. Below are the commands. If the screen is connected to the system inside the structure, no need to do SSH, start from the second line.

- ssh to the system - `ssh <user>@<address>` (user details are in the attached file)
- start screen session - `screen -s NN`

- start environment - `conda activate miniforge` (only needed if a Python environment is created)
- change to working directory - `cd pdp-plexus/pdp-plexus/nn`
- start the server `python testing.py`
- - Once the server is started, a public and a local link will be printed on the console. Copy that before detatching from the screen session
- detatch screen session `cmd+A, D`

Go to above link in the browser to access the UI.

Electronics

The LED grids are controlled using a Teensy 4.0. LED grids are connected to Teensy as below.

- Input layer (bottom) - PIN 6
- Hidden layer (middle) - PIN 7
- Output layer (top) - PIN 8

Main code to be uploaded to Teensy is `pdp-led-control.ino` which is in the directory `PDP-Plexus/Arduino/pdp-led-control`. Below Arduino/Teensiduino packages are required before uploading.

- FastLED
- SerialTransfer

Parameters

Properties of the layers are defined in the beginning of the file `pdp-led-control.ino`. Below are the parameters, some of them are adjustable.

Parameter	Default value	Remark
PIN_INPUT`	6	Input layer pin
PIN_HIDDEN	7	Hidden layer pin
PIN_OUTPUT	8	Output layer pin
NETWORK_LAYER_SIZE	28	Size of layer in the neural network. It is always 28.
SCALE_FACTOR	1	This indicates the size of the physical layer. If the physical layer is doubled (56 x 56 LEDs), SCALE_FACTOR is 2.
HIDDEN_ACTUAL_SIZE	12	Size of the grid where the actual values of hidden layers are shown. It is 12 in the neural network, which is not changed.
ENABLE_PATTERN	1	Enables rainbow pattern when no one uses the system
TIMEOUT	50000	Sets timeout in mimcroseconds after which the pattern to be displayed (disabled in the code)

Note: The hidden layer is inverted when it is fixed on the structure. Correspondingly the code is adjusted to incorporate that.