RayBNN

RayBNN: A 3-D Biological Neural Network Transfer Learning Model

System Requirements

- RTX 3090 or more powerful with at least 24GB VRAM
- 32GB RAM
- 20 GB of disk space
- Docker (https://www.docker.com/)
- Rust (https://www.rust-lang.org/)
- Arrayfire (https://github.com/arrayfire/arrayfire)
- Arrayfire Rust (https://github.com/arrayfire/arrayfire-rust)
- Pytorch (https://pytorch.org/)
- Pytorch geometric (https://github.com/pyg-team/pytorch_geometric)
- Matlab

Important Functions

Sphere Cell Collision Functions to generate sphere and delete collided cells

`RayBNN/src/physics/initial_f32.rs`

- `sphere_cell_collision_batch()` : Generates a sphere and detects cell collisions in batch. Where all cells are check at the same time
- `sphere_cell_collision_serial()` : Generates a sphere and detects cell collisions in serial. Where each cell is checked one by one
- `sphere_cell_collision_minibatch()` : Generates a sphere and detects cell collisions in minibatch. Where groups/minibatches of cells are checked

Input Neuron Assignments

`RayBNN/src/physics/initial_f32.rs`

- `create_spaced_input_neuron_on_sphere()` : Creates input neurons on the surface of a sphere for 2D images of size (Nx,Ny)
- `create_spaced_input_neuron_on_sphere_1D()` : Creates input neurons on the surface of a sphere for 1D data with random neuron position assignment

Raytracing Algorithms

`RayBNN/src/physics/raytrace_f32.rs`

- `RT1_random_rays()` : Raytracing algorithm 1 for creating neural connections. Randomly generates rays of random directions with variable number of random rays
- `RT2_directly_connected()` : Raytracing algorithm 2 for creating neural connections. Connects all neurons within the neural network sphere at the same time
- `RT3_distance_limited_directly_connected()` : Raytracing algorithm 3 for creating neural connections. Connects all neurons within minibatches/groups of neurons

Neural Network Training Algorithms

`RayBNN/src/neural/network_f32.rs`

- `state_space_forward_batch()` : Forward pass using CSR weighted adjacency sparse matrices and UAF. Generates all internal states and the neural network output
- `state_space_backward_group2()` : Backward pass using CSR weighted adjacency sparse matrices and UAF. Generates the gradients of the sparse weighted adjacency matrix

Installation Guide

1. On the Host Machine. Place RayBNN.zip into $RAYBNN_DIR and unzip it.

Set the $RAYBNN_DIR enviromental variable to a folder that will store all the RayBNN files

For example, setting $RAYBNN_DIR to /opt/

```
export RAYBNN_DIR=/opt/
```

Place RayBNN.zip into $RAYBNN_DIR and unzip it to produce:

- $RAYBNN_DIR/RayBNN/src/
- $RAYBNN_DIR/RayBNN/examples/
- $RAYBNN_DIR/RayBNN/matlab_plot/
- $RAYBNN_DIR/RayBNN/python_verify/

2. Make Sure Matlab is Installed On the Host Machine.

Make sure Matlab is installed on the host system. Tested on Matlab 2023a

3. On the Host Machine. Download CUDA Docker Container from Nvidia

This will download a CUDA Docker Container and link $RAYBNN_DIR directory in the Host Machine to the /workspace/ directory in the Docker Container.

```
docker run --name raybnn \
--gpus all \
-v $RAYBNN_DIR:/workspace \
-w /workspace  \
-it  nvcr.io/nvidia/cuda:12.1.1-cudnn8-devel-ubuntu22.04  bash
```

4. Inside the Docker Container, Verify the GPU is detected and CUDA is 12.1

Note that you need a GPU with 24 GB or more VRAM to run all of the code. RayBNN was tested on RTX3090. Verify the GPU is detected

```
nvidia-smi
```

5. Inside the Docker Container, Install Dependencies and Install RayBNN

`./install.sh` installs all of the dependencies inside the docker container.

```
cd /workspace/RayBNN

chmod 700 ./install.sh

bash ./install.sh

exit
```

6. Install the Docker Container to run other models

Download pytorch docker container Tested on RTX 3090 with i5-8400

```
docker run --name othermodel \
--gpus all \
-v $RAYBNN_DIR:/workspace \
-w /workspace  \
-it  pytorch/pytorch:1.13.1-cuda11.6-cudnn8-devel  bash

apt update

apt install wget git  curl git-lfs build-essential


pip install scikit-learn matplotlib pandas pytorch-lightning==1.9.0

pip install torch_geometric

pip install pyg_lib torch_scatter \
torch_sparse torch_cluster \
torch_spline_conv \
-f https://data.pyg.org/whl/torch-1.13.0+cu116.html



exit
```

Reproducing the Results in the Manuscript

Reproducing results in batch

On the Host Machine, Restart the Docker Container

```
docker restart raybnn

docker exec -it raybnn bash

cd /workspace/RayBNN

bash run_results_rust.sh

exit

docker restart othermodel

docker exec -it othermodel bash

cd /workspace/RayBNN

bash run_results_fig4_other_models.sh

exit

bash plot_results_matlab.sh
```

Reproducing inidividual results

On the Host Machine, Restart the Docker Container

```
docker restart raybnn

docker exec -it raybnn bash

cd /workspace/RayBNN
```

Plot Figure 1b, an example of a simple neural network

Related scripts at

- `RayBNN/examples/figure1b.rs`
- `RayBNN/matlab_plot/figure1b_plot.m`

Figure 1b is an example of a simple neural network with raytraced connections

Inside Docker Container, generate `./figure1_neural_network.csv` that contains the entire neural network

```
cd /workspace/RayBNN
cargo run --example figure1b --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./figure1_neural_network.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure1b_plot.m
```

Plot Figure 2a Measuring the Cell Density and Probability of Collisions

Related scripts at

- `RayBNN/examples/figure2a.rs`
- `RayBNN/matlab_plot/figure2a_plot.m`

The neural network sphere radius is constant, while the number of cells changes. It allows us to plot cell density vs the probability of cell collisions

Inside Docker Container, generate ./initial_cell_num.csv ./final_neuron_num.csv ./final_glia_num.csv ./collision_run_time.csv

```
cd /workspace/RayBNN
cargo run --example figure2a --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./initial_cell_num.csv ./final_neuron_num.csv ./final_glia_num.csv ./collision_run_time.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2a_plot.m
```

Plot Figure 2b Measuring runtime of various collision detection algorithms

Related scripts at

- `RayBNN/examples/figure2b.rs`
- `RayBNN/matlab_plot/figure2b_plot.m`

The code runs serial, mini-batch, and batch versions of cell collision detection. It compares the runtimes of those algorithms

Inside Docker Container, generate ./collision_run_time.csv ./collision_run_time_serial.csv ./collision_run_time_batch.csv

```
cd /workspace/RayBNN
cargo run --example figure2a --release
cargo run --example figure2b --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./collision_run_time.csv ./collision_run_time_serial.csv ./collision_run_time_batch.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2b_plot.m
```

Plot Figure 2c Distribution of Cells as a function of radius

Related scripts at

- `RayBNN/examples/figure2c.rs`
- `RayBNN/matlab_plot/figure2c_plot.m`

This code generates 240,000 neurons and 240,000 glial cells in a 739.81 radius network It is intended to plot the distribution of cells as a function of radius

Inside Docker Container, generate ./neuron_pos.csv ./glia_pos.csv

```
cd /workspace/RayBNN
cargo run --example figure2c --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./neuron_pos.csv ./glia_pos.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2c_plot.m
```

Plot Figure 2d Runtimes of Various Raytracing Algorithms

Related scripts at

- `RayBNN/examples/figure2d.rs`
- `RayBNN/matlab_plot/figure2d_plot.m`

This code benchmarks the runtimes of RT-1,RT-2, and RT-3. RT-3 has variable 20, 40, and 60 neuron radii

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure2d --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./RT1_run_time.csv ./RT2_run_time.csv ./RT3_20_run_time.csv ./RT3_40_run_time.csv ./RT3_60_run_time.csv ./neuron_num_list.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2d_plot.m
```

Plot Figure 2e Probability Density Function of the Ray Lengths

Related scripts at

- `RayBNN/examples/figure2e.rs`
- `RayBNN/matlab_plot/figure2e_plot.m`

This code uses RT-3 to plot the probability density function of the raylengths compared to the density of the neural network sphere

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure2e --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./WRowIdxCOO_*.csv ./WColIdx_*.csv ./neuron_pos_*.csv ./neuron_idx_*.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2e_plot.m
```

Plot 2f Probability Density Function of the Number of Connections per Neuron

Related scripts at

- `RayBNN/examples/figure2f.rs`
- `RayBNN/matlab_plot/figure2f_plot.m`

This code uses RT-3 to plot the probability density function of the number of neural connections per neuron compared to the density of the neural network sphere

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure2f --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./WRowIdxCOO_*.csv ./WColIdx_*.csv ./neuron_pos_*.csv ./neuron_idx_*.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure2f_plot.m
```

Plot Figure 3b and 3c Probability Distribution of Weights and Deleted Weights

Related scripts at

- `RayBNN/examples/figure3b.rs`
- `RayBNN/matlab_plot/figure3b_plot.m`
- `RayBNN/matlab_plot/figure3c_plot.m`

This code trains a neural network and probabilistically deletes 5% of the smallest weights. The weight distribution and deleted weights are plotted

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure3b --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./before_delete_WRowIdxCOO.csv ./before_delete_WColIdx.csv ./before_delete_WValues.csv ./after_delete_WRowIdxCOO.csv ./after_delete_WColIdx.csv ./after_delete_WValues.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure3b_plot.m
matlab -r figure3c_plot.m
```

Plot Figure 3d,3e, and 3f Sparsity, UAF, and Weighted Adjancency matrix

Related scripts at

- `RayBNN/examples/figure3d.rs`
- `RayBNN/matlab_plot/figure3d_plot.m`
- `RayBNN/matlab_plot/figure3e_plot.m`
- `RayBNN/matlab_plot/figure3f_plot.m`

This code trains a neural network and plots the sparsity of weights. It also plots UAF and the weighted adjancency matrix

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure3d --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./sparsenetwork_*.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure3d_plot.m
matlab -r figure3e_plot.m
matlab -r figure3f_plot.m
```

Plot Figure 4

Running RayBNN for the Alcala Dataset

Alcala Dataset from IndoorLoc Platform

https://web.archive.org/web/20211130114720/http://indoorlocplatform.uji.es/

Related scripts at

- `RayBNN/examples/figure4_raybnn.rs`
- `RayBNN/matlab_plot/figure4a_plot.m`
- `RayBNN/matlab_plot/figure4b_plot.m`
- `RayBNN/matlab_plot/figure4c_plot.m`
- `RayBNN/matlab_plot/figure4d_plot.m`
- `RayBNN/matlab_plot/figure4e_plot.m`
- `RayBNN/matlab_plot/figure4f_plot.m`

Run the 10 Fold Testing for the Alcala Dataset in Figure 4 Note that CUDA has compile the kernels at runtime so the first run is slower. Tested on RTX 3090 with i5-8400

Inside Docker Container, generate all the csv files

```
cd /workspace/RayBNN
cargo run --example figure4_raybnn --release
mv *.csv ./matlab_plot/
```

On the Host Machine, Plot ./info_*.csv ./test_act_*.csv ./test_pred_*.csv using Matlab

```
cd $RAYBNN_DIR/RayBNN/matlab_plot/
matlab -r figure4a_plot.m
matlab -r figure4b_plot.m
matlab -r figure4c_plot.m
matlab -r figure4d_plot.m
matlab -r figure4e_plot.m
matlab -r figure4f_plot.m
```

Running other Pytorch code for the Alcala Dataset

On the Host Machine, Restart the Docker Container

```
docker restart othermodel
docker exec -it othermodel bash
cd /workspace/RayBNN
```

Running a CNN model for the Alcala dataset

Example running the CNNRSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
 cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py CNNRSSI.py

python3 ./getresult.py
```

Running a GCN2 model for the Alcala dataset

Example running the GCN2RSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
 cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py GCN2RSSI.py

python3 ./getresult.py
```

Running a LSTM model for the Alcala dataset

Example running the LSTMRSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
 cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py LSTMRSSI.py

python3 ./getresult.py
```

Running a MLP model for the Alcala dataset

Example running the MLPRSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
 cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py MLPRSSI.py

python3 ./getresult.py
```

Running a GCN2LSTM model for the Alcala dataset

Example running the GNNLSTMRSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
 cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py GNNLSTMRSSI.py

python3 ./getresult.py
```

Running a BILSTM model for the Alcala dataset

Example running the BILSTMRSSI.py code for 10 fold testing

Inside Docker Container, generate all the .dat files

```
cd /workspace/RayBNN/python_verify/RSSI2/

python3 ./All_run.py BILSTMRSSI.py

python3 ./getresult.py
```

Results for Table 1

Download the 210 GB EEG dataset

http://gigadb.org/dataset/100542

Follow the preprocessing steps

Preprocessing steps to obtain the processed `KU_mi_smt.h5` dataset

https://github.com/zhangks98/eeg-adapt

Running CSP_LDA

Running CSP LDA for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./CSP_info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/CSP_LDA/
python3 ./train_CSP_LDA.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./parseData.py
```

Running CSP_LR

Running CSP LR for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./CSP_info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/CSP_LR/
python3 ./train_CSP_LR.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./parseData.py
```

Running Xdawn MDM

Running Xdawn MDM for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./MDM_info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Xdawn_MDM/
python3 ./train_Xdawn_MDM.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./parseData.py
```

Running Xdawn LR

Running Xdawn LR for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./LR_info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Xdawn_LR/
python3 ./train_Xdawn_LR.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./parseData.py
```

Running Deep4Net

Running Deep4Net for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./CNN_info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Deep4Net/
python3 ./train_base_Deep4Net.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./parseData.py
```

Running Xdawn_Deep4Net_MLP

Running Xdawn_Deep4Net_MLP for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_MLP/
python3 ./preprocess_Xdawn_Deep4Net.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
python3 ./MLPEEG.py  ./  42
python3 ./getresult.py
```

Running Deep4Net_RayBNN

Running Deep4Net_RayBNN for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Deep4Net_RayBNN/
python3 ./preprocess_Deep4Net_RayBNN.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
cargo run --example table1_deep4net_raybnn    --release  ./  ./  42
python3 ./getresult.py
```

Running Xdawn_Deep4Net_RayBNN

Running Xdawn_Deep4Net_RayBNN for a specific fold number 42 and GPU number 1

Input: /path_to/KU_mi_smt.h5 Dataset Output: ./info_*.txt Containing Accuracy, F1, AUC ROC

```
cd /workspace/RayBNN/python_verify/EEG/Deep4Net_RayBNN/
python3 ./preprocess_Deep4Net_RayBNN.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
cargo run --example table1_deep4net_raybnn    --release  ./  ./  42
cp ./* /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_RayBNN/



cd /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_RayBNN/
python3 ./preprocess_Xdawn_Deep4Net_RayBNN.py   /path_to/KU_mi_smt.h5   ./   -fold  42 -gpu 1
cargo run --example table1_transfer_xdawn_deep4net_raybnn    --release  ./  ./  42
python3 ./getresult.py
```

Plotting Figure 5b

Compare RayBNN with MLP+Dropout

```
cd /workspace/RayBNN/
cargo run --example table1_raybnn_optim     --release  /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_RayBNN/   41
cp ./*.csv /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_MLP/


cd /workspace/RayBNN/python_verify/EEG/Xdawn_Deep4Net_MLP/
python3 ./MLPEEG_optim.py  ./   41
matlab  plotcmp.m
```

Plotting Figure 5c

Plot the accuracy for all algorithms

```
 cd /workspace/RayBNN/python_verify/EEG/
cp ./Deep4Net/acc*  ./
cp ./Xdawn_Deep4Net_MLP/acc*  ./
cp ./Xdawn_MDM/acc*  ./
cp ./CSP_LDA/acc*  ./
cp ./CSP_LR/acc*  ./
cp ./Deep4Net_RayBNN/acc*  ./
cp ./Xdawn_Deep4Net_RayBNN/acc*  ./
cp ./Xdawn_LR/acc*  ./

matlab ./plot_EEG_MI.m
```