Run CNN+RayBNN on CCDB

Package RayBNN_python

Source Code For RayBNN v2.0.1

https://github.com/BrosnanYuen/RayBNN_Neural

https://github.com/BrosnanYuen/RayBNN_DataLoader

https://github.com/BrosnanYuen/RayBNN_Sparse

https://github.com/BrosnanYuen/RayBNN_Raytrace

https://github.com/BrosnanYuen/RayBNN_Python

https://github.com/BrosnanYuen/RayBNN_Cell

https://github.com/BrosnanYuen/RayBNN_Optimizer

Git clone RayBNN_Python

```
# Load modules
#module --force purge
module load StdEnv/2020 gcc/9.3.0 cuda/12.2 fmt/9.1.0 spdlog/1.9.2 arrayfire/3.9.0 rust/1.70.0 python/3.11.2 openblas
```

Load necessary module(I add openblas)

University of Victoria

Package RayBNN_python(cont.)

```
virtualenv magic
source magic/bin/activate
pip install maturin numpy patchelf
```

install dependencies in the virtual environment

```
maturin develop
echo "maturin success"
```

use maturin to package Rust code into python wheel

```
(magic) [lain1385@narval3 CNN]$ pip list
Package           Version                   Editable project location
----------------- ------------------------- ---------------------------------------------------------
python_dateutil   2.9.0.post0+computecanada
raybnn_python     0.1.2                     /lustre07/scratch/lain1385/project/RayBNN_Python/Rust_Code
requests          2.32.3
```

Check that raybnn_python is installed successfully as python module

This process doesn't need to be submitted to compute cluster

University of Victoria

Package RayBNN_python(cont.)

```
project > RayBNN_Python > Rust_Code > example.py
1   import numpy as np
2   import raybnn_python
```

```
13  print("Rust")
14
15  z = raybnn_python.magic2(x)
16
17  print(z)
```

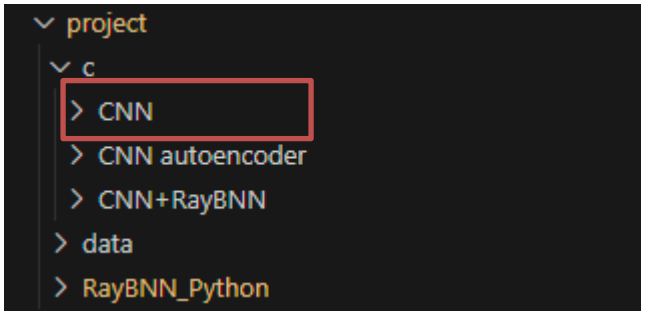
Try RayBNN_Python/Rust_Code/example.py to validate raybnn_python package

```
0.602763    0.891773    0.568045
0.832620    0.461479    0.944669

0.544883    0.963663    0.925597
0.778157    0.780529    0.521848

0.423655    0.383442    0.071036
0.870012    0.118274    0.414662


b
[1 1 1 1]
   0.778157
[[[0.5488135  0.71518934 0.60276335 0.5448832  0.4236548 ]
  [0.6458941  0.4375872  0.891773   0.96366274 0.3834415 ]
  [0.79172504 0.5288949  0.56804454 0.92559665 0.07103606]]

 [[0.0871293  0.0202184  0.83261985 0.77815676 0.87001216]
  [0.9786183  0.7991586  0.46147937 0.7805292  0.11827443]
  [0.639921   0.14335328 0.9446689  0.5218483  0.41466194]]]
0.77815676
Rust
{'dtype': 'F32', 'shape': {'dims': (2, 3, 5, 1)}, 'data': [0.54881352186203, 0.08712930232286453, 0.64
```

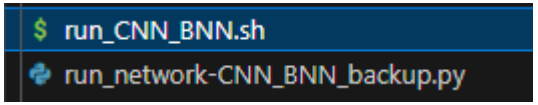
Submit jobs to run example.py and ouput is shown above

University of Victoria

Try code from Xuan



Download code of Xuan and unzip it (using Globus to transfer it onto CCDB)



Try to run C/CNN Code

```
if __name__ == '__main__':

    val_features, val_labels, train_features, train_labels = main()

    output_y = train_raybnn(train_features, train_labels, val_features, val_labels)
```

Use CNN model to extract features and labels

Then apply raybnn to train it

University of Victoria

Prepare Dataset

```python
import numpy as np
import raybnn_python
import torch
from torch import nn, optim
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.model_selection import KFold
import os
from torchvision import datasets, transforms,utils
from torch.utils.data import ConcatDataset, Subset, DataLoader
from torch import optim
import torch.nn.functional as F

```

Pip install necessary packages in the virtual environment

```python
full_dataset = ConcatDataset([
    datasets.MNIST(root="/home/lain1385/scratch/project/data", transform=transform, train=True, download=True),
    datasets.MNIST(root="/home/lain1385/scratch/project/data", transform=transform, train=False, download=True)
])
```

Set download=True to automatically download MNIST Dataset

University of Victoria

CNN module

```python
class CNN(nn.Module):
    def __init__(self, input_dim, output_dim, feature_dim):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(input_dim[0], 5, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(5, 10, kernel_size=3, stride=1, padding=1)
        self.flatten = nn.Flatten()
        latent_shape = self._get_conv_shape(input_dim=input_dim)
        self.fc1 = nn.Linear(latent_shape[1], 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 10)
        self.dropout = nn.Dropout(0.5)
```

layer setting

(I reduced kernel channel)

conv1: convolution layer with5 output channels, a kernel size of 3x3, stride of 1, and padding of 1.

pool: A max pooling layer with a kernel size of 2x2 and a stride of 2.

flatten: A layer that flattens the input

fc1: A fully connected (linear) layer

For extracted feature: image pixel -> conv1 -> pool -> conv2 -> flatten

University of Victoria

CNN output

```
val_features, val_labels, train_features, train_labels = main()
np.save('val_features.npy',val_features)
np.save('val_labels.npy',val_labels)
np.save('train_features.npy',train_features)
np.save('train_labels.npy',train_labels)
```

Save output data

```
val_features: (2000, 490)

val_labels: (2000,)

train_features: (8000, 490)

train_labels: (8000,)
```

Check shape of features & labels after CNN

(# of samples set to 10000)

Original features dimension: 3136

Features dimension after modification: 490

University of Victoria

CNN output

```
Epoch [1/10], Loss: 1.55051, Accuracy: 61.03750%
Epoch [2/10], Loss: 0.61459, Accuracy: 81.88750%
Epoch [3/10], Loss: 0.47207, Accuracy: 85.96250%
Epoch [4/10], Loss: 0.41368, Accuracy: 87.21250%
Epoch [5/10], Loss: 0.36751, Accuracy: 88.97500%
Epoch [6/10], Loss: 0.33736, Accuracy: 89.72500%
Epoch [7/10], Loss: 0.32006, Accuracy: 90.06250%
Epoch [8/10], Loss: 0.29043, Accuracy: 91.11250%
Epoch [9/10], Loss: 0.27371, Accuracy: 91.55000%
Epoch [10/10], Loss: 0.25019, Accuracy: 92.01250%
Fold 1: Accuracy=0.94700, Precision=0.94638, Recall=0.94403, F1 Score=0.94482
Precision: 0.94638, Recall: 0.94403, F1 Score: 0.94482
Epoch [10/10], Test Loss: 0.39262, Test Accuracy: 94.70000%
Epoch [1/10], Loss: 1.47166, Accuracy: 62.33750%
Epoch [2/10], Loss: 0.55135, Accuracy: 82.93750%
```

The output during model training

K fold cross validation: k = 5 (the original is 10)

Epoch = 10

University of Victoria

RaybNN training

```
val_features = np.load('val_features.npy')
val_labels = np.load('val_labels.npy')
train_features = np.load('train_features.npy')
train_labels = np.load('train_labels.npy')

output_y = train_raybnn(train_features, train_labels, val_features, val_labels)
```

Load saved data and invoke function train_raybnn

```
def train_raybnn(x_train, y_train, x_test, y_test):
    accuracy_values = []
    precision_values=[]
    recall_values = []
    f1_values=[]
```

```
dir_path = "/home/lain1385/scratch/project/c/CNN/tmp"

max_input_size = 512
input_size = 490

max_output_size = 10
output_size = 10
```

modify input_size to match CNN output in the function train_raybnn

University of Victoria

RaybNN training ouput

```
******Network Information******
neuron_size: 1000
input_size: 490
output_size: 10
proc_num: 2
active_size: 978
space_dims: 3
step_num: 5000
batch_size: 1000
del_unused_neuron: true
time_step: 0.1
nratio: 0.5
neuron_std: 0.001
sphere_rad: 11.882076
neuron_rad: 0.1
con_rad: 9.190047
init_prob: 0.01
add_neuron_rate: 0
del_neuron_rate: 0
center_const: 0.005
spring_const: 0.01
repel_const: 0.01
***********
```

Parameters of RayBNN training

```
WValues.dims()[0] 80088
Start training
loss: 0.9709589, alpha0: 0.01, i: 0
loss: 0.94332623, alpha0: 0.01, i: 1
/scratch/lain1385/project/RayBNN_Python/magic/lib/pytho
  _warn_prf(average, modifier, msg_start, len(result))
Start training
loss: 0.9438329, alpha0: 0.01, i: 0
loss: 0.8922663, alpha0: 0.01, i: 1
loss: 0.87254024, alpha0: 0.01, i: 2
/scratch/lain1385/project/RayBNN_Python/magic/lib/pytho
  _warn_prf(average, modifier, msg_start, len(result))
Start training
loss: 0.8666107, alpha0: 0.01, i: 0
loss: 0.8581241, alpha0: 0.01, i: 1
loss: 0.7695129, alpha0: 0.01, i: 2
loss: 0.6814341, alpha0: 0.01, i: 3
Start training
loss: 0.68098223, alpha0: 0.01, i: 0
loss: 0.644085, alpha0: 0.01, i: 1
loss: 0.5558632, alpha0: 0.01, i: 2
loss: 0.47429863, alpha0: 0.01, i: 3
loss: 0.40849525, alpha0: 0.01, i: 4
Start training
loss: 0.41367176, alpha0: 0.01, i: 0
loss: 0.4257643, alpha0: 0.01, i: 1
loss: 0.3790139, alpha0: 0.01, i: 2
loss: 0.31628898, alpha0: 0.01, i: 3
loss: 0.3286937, alpha0: 0.01, i: 4
loss: 0.2764489, alpha0: 0.01, i: 5
Start training
```

Loss during RayBNN training

Still working on how to improve performance

University of Victoria

Thank you

University of Victoria