Deep Learning in GBAR

Undergraduate Intern
Byeongyoon Park

File Conversion

- Slightly modified the code made by Hobin Lee.
- Raw dataset has 100,000 up, down, background signals each, with .root file format.
- Signals are made by Monte Carlo simulation.
- Using ‘uproot’, we can convert .root file into .hdf5 file in Python environment.
- A single converted signal is 112*88 = 9856 length vector.
- HDF(Hierarchical Data Format) is a file format designed to store and orginize large amounts of data.

SignalMCevrec_bg.root	2020-02-14 오...	ROOT 파일
SignalMCevrec_dw.root	2020-02-14 오...	ROOT 파일
SignalMCevrec_up.root	2020-02-14 오...	ROOT 파일
SignalMCwaveform_flatten.hdf5	2020-02-18 오...	HDF5 파일

Almost 12 hours has past to form a single dataset.
The capacity of .hdf5 is about 1.5GB.

Deep Learning - FCN

- Just a normal Deep Neural Network with fully connected neurons.
- Used desktop in home to process in GPU.
- General settings, and changed ‘# of neurons’, ‘epochs’, ‘batch size’.
- Sought for optimal settings with maximum accuracy.

```
270000/270000 [==============================] - 657s 2ms/step - loss: 0.9258 - acc: 0.5346
Epoch 2/10
270000/270000 [==============================] - 418s 2ms/step - loss: 0.8784 - acc: 0.5643
Epoch 3/10
270000/270000 [==============================] - 337s 1ms/step - loss: 0.8659 - acc: 0.5731
Epoch 4/10
270000/270000 [==============================] - 332s 1ms/step - loss: 0.8622 - acc: 0.5758
Epoch 5/10
270000/270000 [==============================] - 342s 1ms/step - loss: 0.8581 - acc: 0.5779
Epoch 6/10
270000/270000 [==============================] - 622s 2ms/step - loss: 0.8570 - acc: 0.5781
Epoch 7/10
270000/270000 [==============================] - 622s 2ms/step - loss: 0.8558 - acc: 0.5802
Epoch 8/10
270000/270000 [==============================] - 583s 2ms/step - loss: 0.8527 - acc: 0.5802
Epoch 9/10
270000/270000 [==============================] - 525s 2ms/step - loss: 0.8523 - acc: 0.5813
Epoch 10/10
270000/270000 [==============================] - 1050s 4ms/step - loss: 0.8496 - acc: 0.5826
30000/30000 [==============================] - 55s 2ms/step
test_acc :  0.6254333333333333

(base) C:₩Users₩User₩Google 드라이브₩DL₩UpDwBg10k>cd "Google 드라이브"₩DL₩UpDwBg10k
```

[0]

All layers have 512 neurons.

epochs = 10 , batch size = 128

```
270000/270000 [==============================] - 53s 196us/step - loss: 0.9539 - acc: 0.5064
Epoch 2/10
270000/270000 [==============================] - 51s 187us/step - loss: 0.8635 - acc: 0.6020
Epoch 3/10
270000/270000 [==============================] - 49s 180us/step - loss: 0.8167 - acc: 0.6437
Epoch 4/10
270000/270000 [==============================] - 50s 186us/step - loss: 0.8245 - acc: 0.6403
Epoch 5/10
270000/270000 [==============================] - 59s 219us/step - loss: 0.7971 - acc: 0.6541
Epoch 6/10
270000/270000 [==============================] - 49s 183us/step - loss: 0.7893 - acc: 0.6436
Epoch 7/10
270000/270000 [==============================] - 49s 182us/step - loss: 0.7986 - acc: 0.6406
Epoch 8/10
270000/270000 [==============================] - 49s 180us/step - loss: 0.7802 - acc: 0.6452
Epoch 9/10
270000/270000 [==============================] - 51s 188us/step - loss: 0.7673 - acc: 0.6509
Epoch 10/10
270000/270000 [==============================] - 50s 186us/step - loss: 0.7608 - acc: 0.6564
30000/30000 [==============================] - 22s 722us/step
test_acc :  0.7033333333333334
```

[1]

First layer has 1024, and the rest have 512 each.
epochs = 10 , batch size = 2048

```
270000/270000 [==============================] - 55s 203us/step - loss: 1.0134 - acc: 0.4513
Epoch 2/7
270000/270000 [==============================] - 52s 191us/step - loss: 0.8748 - acc: 0.5923
Epoch 3/7
270000/270000 [==============================] - 52s 191us/step - loss: 0.8302 - acc: 0.6248
Epoch 4/7
270000/270000 [==============================] - 52s 192us/step - loss: 0.8276 - acc: 0.6267
Epoch 5/7
270000/270000 [==============================] - 52s 192us/step - loss: 0.7946 - acc: 0.6427
Epoch 6/7
270000/270000 [==============================] - 75s 279us/step - loss: 0.8302 - acc: 0.6195
Epoch 7/7
270000/270000 [==============================] - 51s 191us/step - loss: 0.8111 - acc: 0.6380
30000/30000 [==============================] - 21s 712us/step
test_acc :  0.6890333333333334
```

[2]

All layers have 1024 neurons.

epochs = 7 , batch size = 2048

```
270000/270000 [==============================] - 66s 243us/step - loss: 0.9391 - acc: 0.5273
Epoch 2/5
270000/270000 [==============================] - 64s 238us/step - loss: 0.8693 - acc: 0.6023
Epoch 3/5
270000/270000 [==============================] - 64s 238us/step - loss: 0.8350 - acc: 0.6253
Epoch 4/5
270000/270000 [==============================] - 64s 235us/step - loss: 0.8039 - acc: 0.6401
Epoch 5/5
270000/270000 [==============================] - 64s 236us/step - loss: 0.7794 - acc: 0.6580
30000/30000 [==============================] - 21s 708us/step
test_acc :  0.6106
```

[3]

All layers have 1024 neurons.
epochs = 5 , batch size = 1024

```
270000/270000 [==============================] - 79s 292us/step - loss: 1.0173 - acc: 0.4657
Epoch 2/7
270000/270000 [==============================] - 74s 273us/step - loss: 0.8493 - acc: 0.6139
Epoch 3/7
270000/270000 [==============================] - 74s 274us/step - loss: 0.8194 - acc: 0.6419
Epoch 4/7
270000/270000 [==============================] - 74s 274us/step - loss: 0.8088 - acc: 0.6427
Epoch 5/7
270000/270000 [==============================] - 74s 275us/step - loss: 0.8271 - acc: 0.6377
Epoch 6/7
270000/270000 [==============================] - 74s 275us/step - loss: 0.7919 - acc: 0.6558
Epoch 7/7
270000/270000 [==============================] - 74s 274us/step - loss: 0.7955 - acc: 0.6534
30000/30000 [==============================] - 23s 754us/step
test_acc :  0.6570333333333334
```

[4]

All layers have 2048 neurons.
epochs = 5 , batch size = 2048

When batch = 4096, the process halted.

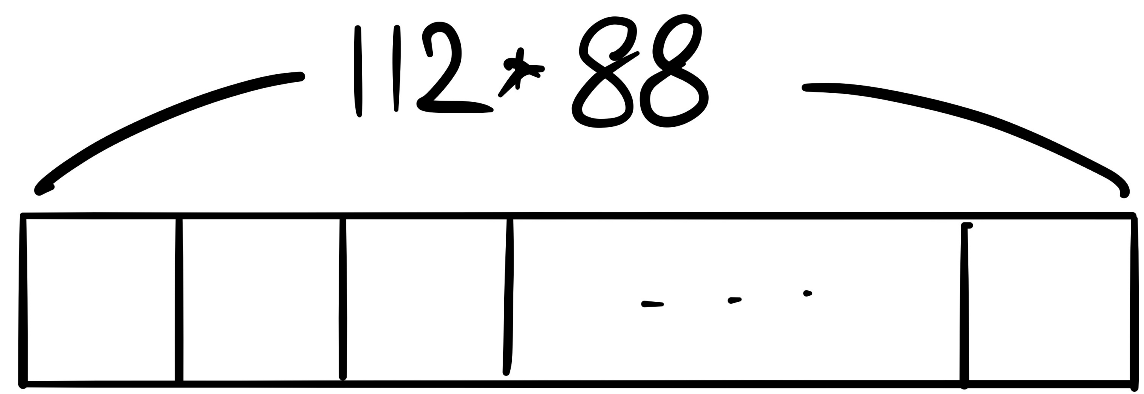
Limit

- No matter how I changed the ‘variables’, I couldn’t enhance the accuracy over 65%.
- Meanwhile, the required rejection rate of cosmic ray (muon) is at least 90%.
- I concluded that the limit of accuracy is unchangeable, at least we use only FCN, which results from the structure of input data and the method of DL.

Convolutional Neural Network

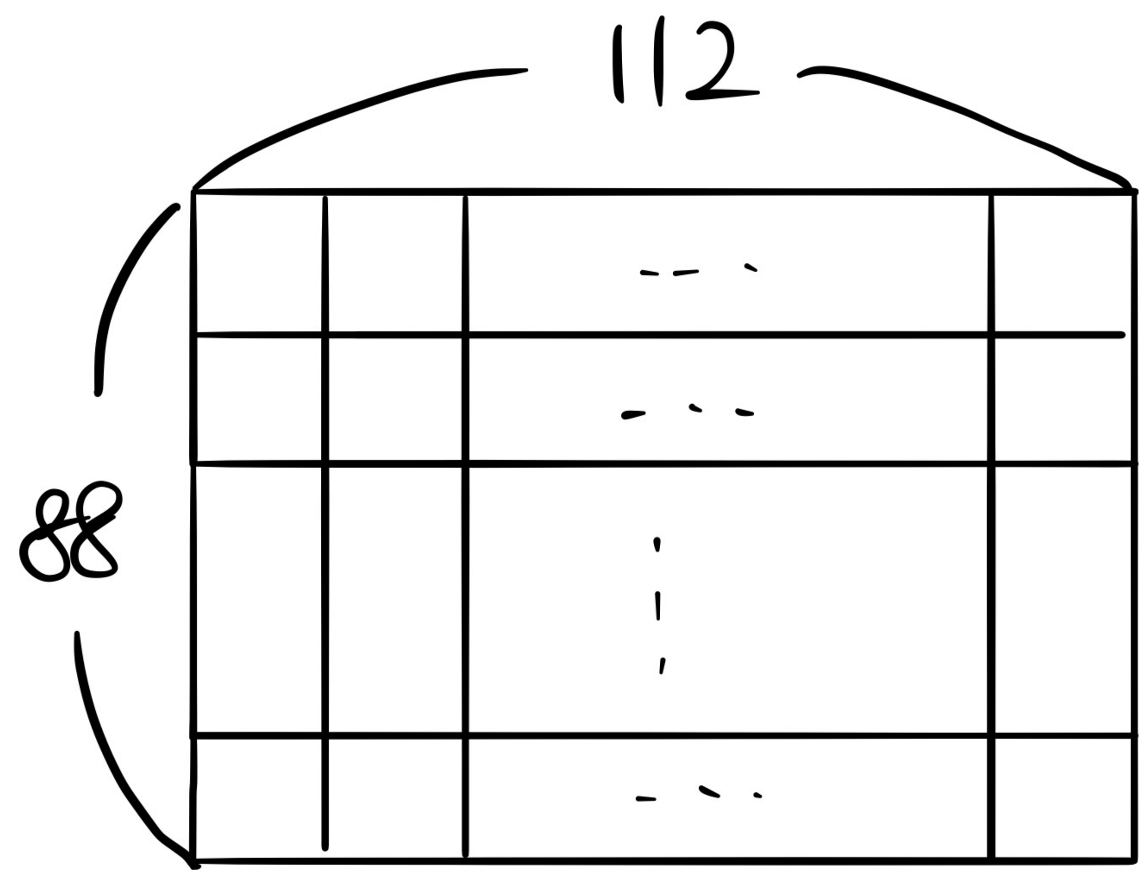
Current Structure

- Single waveform in a PMT has 112 numbers.
- There are 88 PMT signals for single waveform.
- So a waveform is 9856-length vector.

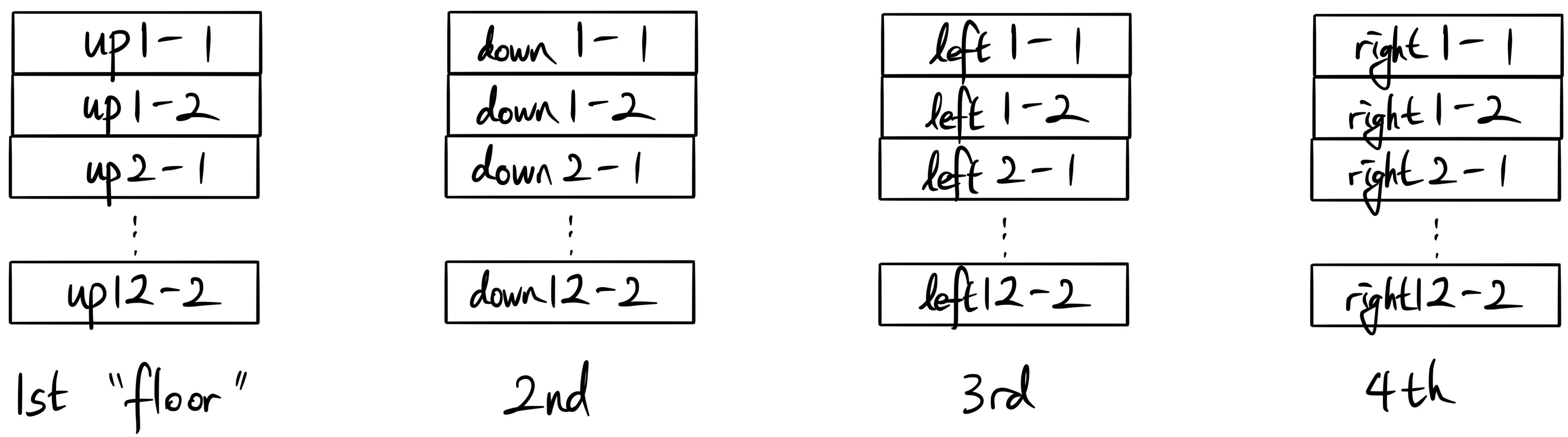


CNN #1

- Convert the flattened waveform into 112*88 matrix.



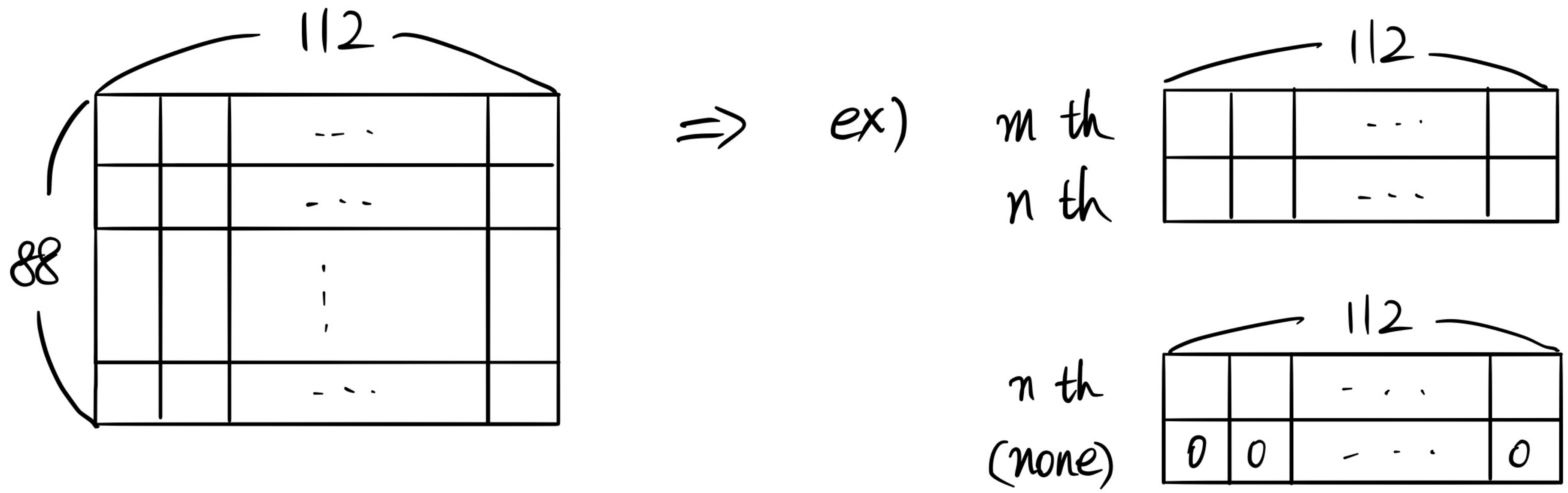
CNN #2



- Convert into 112*24*4 3D tensor.
- Since the number of side PMT is 20, last 4 rows of 3rd, 4th 'floor' are all zeros.

"Give the geometry of TOF to the dataset!"

Other Method?



- Assumption
  : Maybe only 1 or 2 signals would be caught in a single trigger.
  →Delete the unnecessary zero vectors to make the data structure simple.

- Reject the case of ‘more than’ 2 signals in a single trigger.