Dacon Exercise Classification
Private 1st Code

Analysis by Yeseo
https://github.com/newave986

리뷰하면서 인상적이었던 점

1. Agg 할 때 다양한 방법 사용하여 새로운 데이터 세트 만듦

- 논문 적극 활용
- 논문에 구현 코드 다 나와 있는 것도 한몫

2. Feature 만들기에서 다양한 acc/gy 관련 지식 활용한 흔적

- Roll/Pitch라거나, mag/mal, comtrapz 등 제작한 것

Load Data

train
train_acc
train_gy
train_time

train_label
train_y

test
submission

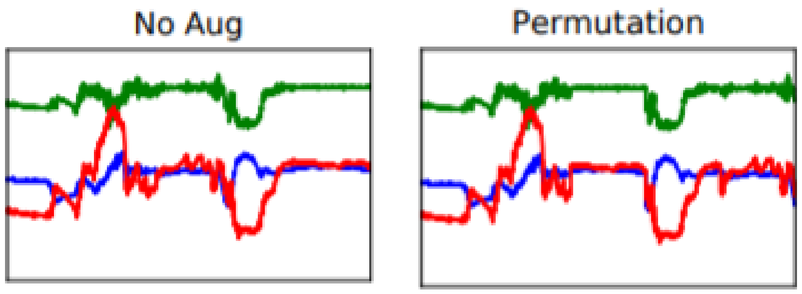
```
train_acc.head()
```

	acc_x	acc_y	acc_z
0	1.206087	-0.179371	-0.148447
1	1.287696	-0.198974	-0.182444
2	1.304609	-0.195114	-0.253382
3	1.293095	-0.230366	-0.215210
4	1.300887	-0.187757	-0.222523

```
train_gy.head()
```

	gy_x	gy_y	gy_z
0	-0.591608	-30.549010	-31.676112
1	0.303100	-39.139103	-24.927216
2	-3.617278	-44.122565	-25.019629
3	2.712986	-53.597843	-27.454013
4	4.286707	-57.906561	-27.961234

Data Aug
Permutation



"신호를 n segment로 나누어 순서를 random하게 바꾸어 주는 방법"

Permutation (Perm) is a simple way to randomly perturb the temporal location of within-window events. To perturb the location of the data in a single window, we first slice the data into *N* same-length segments, with N ranging from 1 to 5, and randomly permute the segments to create a new window. **Time-warping (TimeW)**

- Randomly perturb the temporal location of within-window events. : 창 안에 있는 시간 위치를 랜덤하게 섞음.

- **과정** Single window(단일 창)에 있는 데이터의 위치를 perturb하기 위해,

(1) 데이터를 N의 samelength segments로 나눔: 모두 똑같은 길이(N)를 가지는 부분으로 나눔.
(2) N의 범위는 1~5 N ranging from 1 to 5
(3) 그 조각들을 랜덤하게 섞어서 새로운 데이터 만듦. Randomly permute the segments to create a new window.

Permutation

- For permutation, a random integer N is determined by rounding a positive value sampled from a Gaussian distribution with 5.0 STD. For magnitude-warping and time-warping, random sinusoidal curves are generated using arbitrary amplitude, frequency, and phase values
- N - STD 5.0으로 가우스 분포에 의하여 샘플링된 값을 반올림한 것

```
def permutation(data, nPerm=4, mSL=10):
    data_new = np.zeros(data.shape)
    idx = np.random.permutation(nPerm)
    bWhile = True
    while bWhile == True:
        segs = np.zeros(nPerm+1, dtype=int)
        segs[1:-1] = np.sort(np.random.randint(mSL, data.shape[0]-mSL, nPerm-1))
        segs[-1] = data.shape[0]
        if np.min(segs[1:]-segs[0:-1]) > mSL:
            bWhile = False
    pp = 0
    for ii in range(nPerm):
        data_temp = data[segs[idx[ii]]:segs[idx[ii]+1],:]
        data_new[pp:pp+len(data_temp),:] = data_temp
        pp += len(data_temp)
    return(data_new)
```

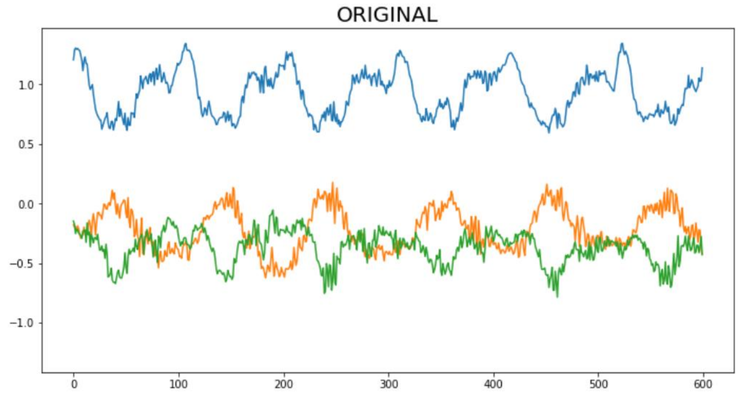
Hyperparameters :
nPerm = # of segments to permute 변경할 부분들의 갯수
minSegLength = allowable minimum length for each segment 각 부분들이 가질 수 있는 최소 길이

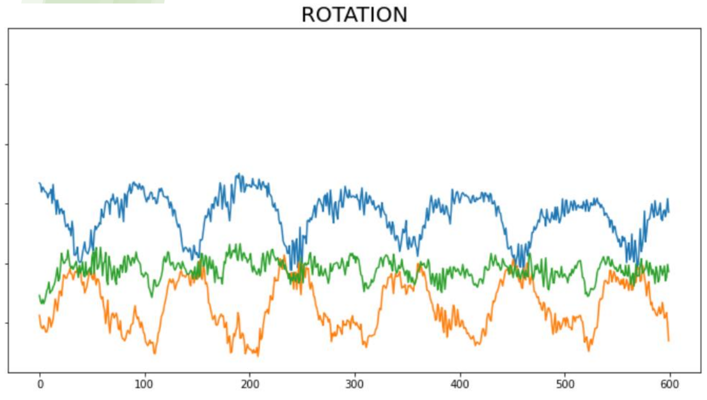
https://github.com/terryum/Data-Augmentation-For-Wearable-Sensor-Data

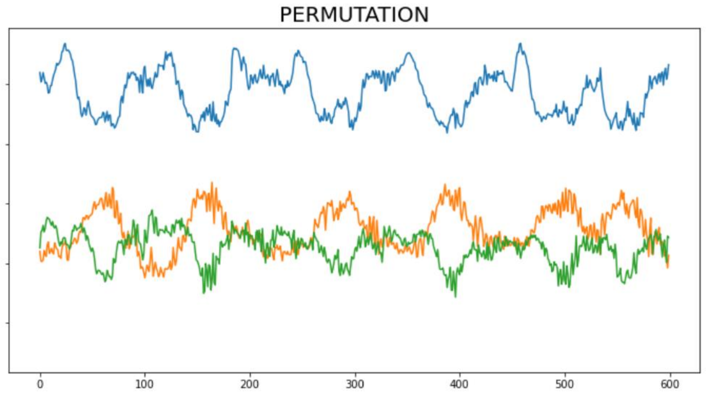
Data Aug **Selected**

- 짝수 epoch: rolling + permutation
- 홀수 epoch: rolling + rotation

id = 0일 때의 acc 값들을 각각 rotation, permutation 함수 이용하여 증강시킨 결과를 시각화:





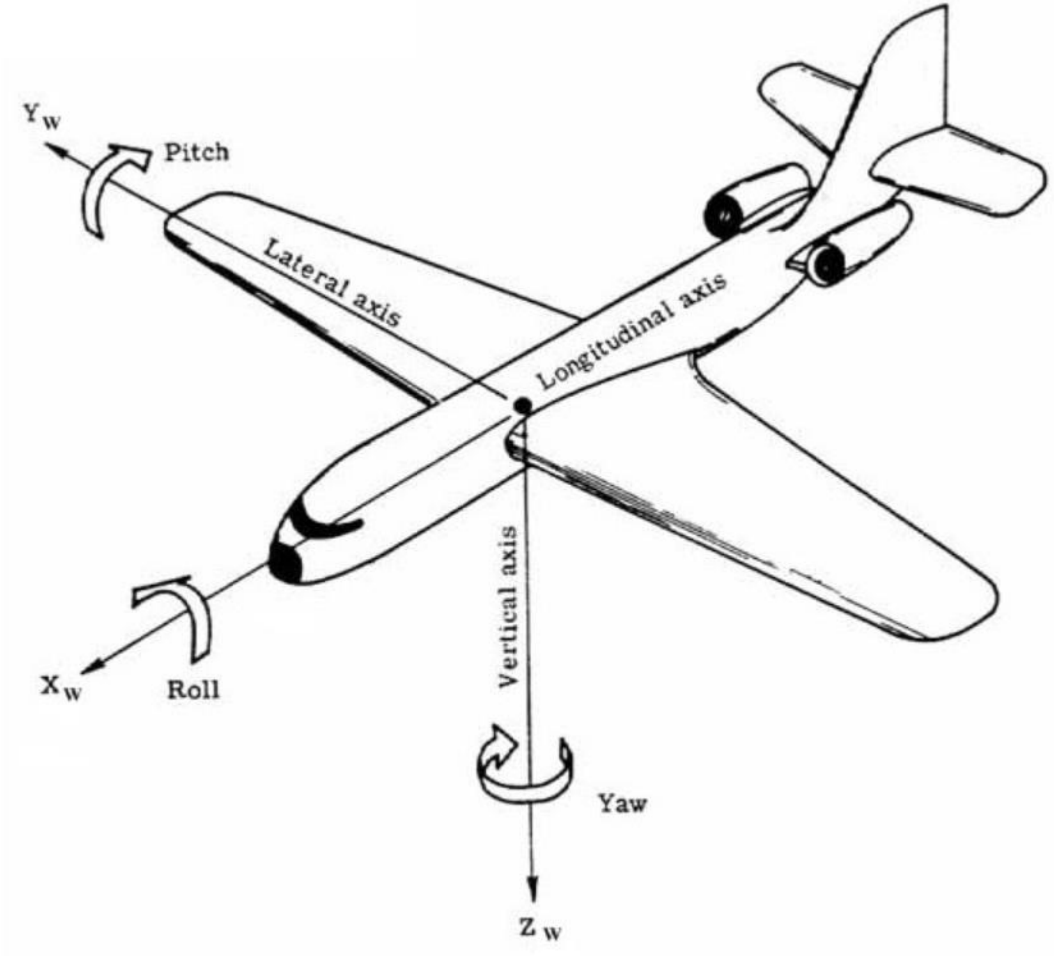


Feature

get_mag, get_mul, get_roll_pitch, setting, get_diff, get_cumtrapz

$$x^2 + y^2 + z^2$$

$$xyz$$



```python
def get_roll_pitch(data):
    roll = (data.iloc[:,1]/(data.iloc[:,0]**2 + data.iloc[:,2]**2).apply(lambda x : sqrt(x))).apply(lambda x : atan(x))*180/np.pi
    pitch = (data.iloc[:,0]/(data.iloc[:,1]**2 + data.iloc[:,2]**2).apply(lambda x : sqrt(x))).apply(lambda x : atan(x))*180/np.pi
    return pd.concat([roll, pitch], axis= 1)
```

Roll & Pitch

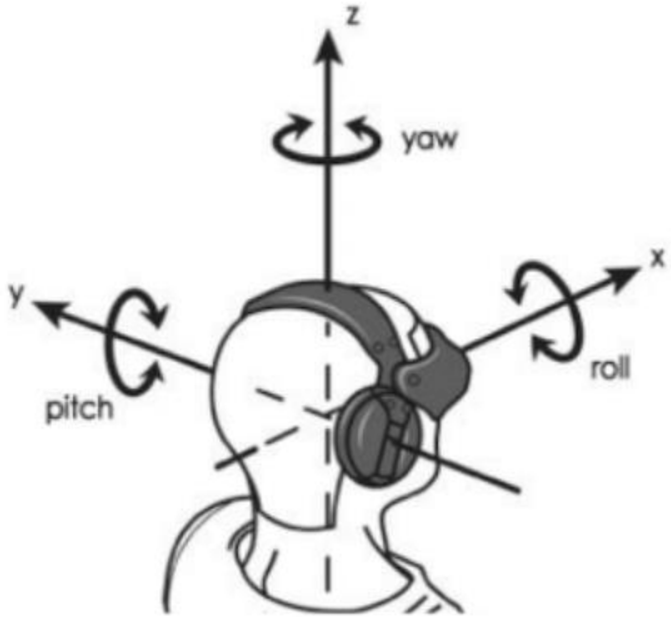
- Roll: x축(종축)을 중심으로 회전하는 변화각
- Pitch: y축(횡축)을 중심으로 회전하는 변화각
- Yaw: z축(수직축)을 중심으로 회전하는 변화각

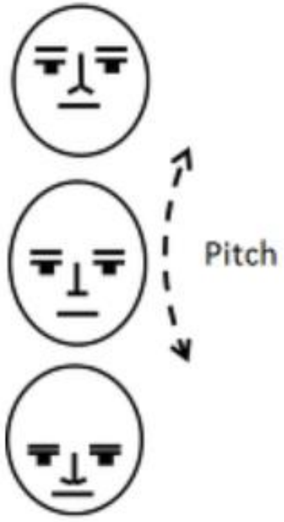
Pitch = atan2(Ax, sqrt(Ay²+Az²));

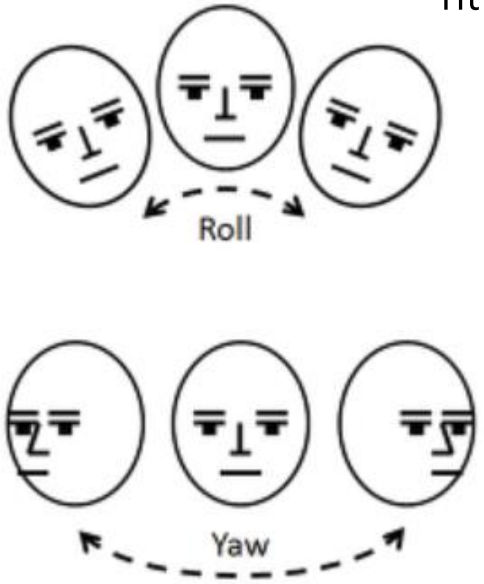
Roll = atan2(-Ay, -Az);

Yaw= atan2((-**Hy***cos(Roll) + Hz*sin(Roll)) , **Hx***cos(Pitch) + Hy*sin(Pitch)*sin(Roll) + Hz*sin(Pitch)*cos(Roll))

https://slideplayer.com/slide/9952882/







https://happy8earth.tistory.com/492

https://pandas.pydata.org/docs/reference/api/pandas.concat.html

```
def get_cumtrapz(acc):
    acc_x, acc_y, acc_z = [], [], []
    ds_x, ds_y, ds_z = [], [], []
    for i in range(int(acc.shape[0]/600)):
        acc_x.append(pd.DataFrame(cumtrapz(acc.iloc[600*i:600*(i+1), 0], train_time, initial=0)))
        acc_y.append(pd.DataFrame(cumtrapz(acc.iloc[600*i:600*(i+1), 1], train_time, initial=0)))
        acc_z.append(pd.DataFrame(cumtrapz(acc.iloc[600*i:600*(i+1), 2], train_time, initial=0)))
        ds_x.append(pd.DataFrame(cumtrapz(cumtrapz(acc.iloc[600*i:600*(i+1), 0], train_time, initial=0), train_time, initial=0)))
        ds_y.append(pd.DataFrame(cumtrapz(cumtrapz(acc.iloc[600*i:600*(i+1), 1], train_time, initial=0), train_time, initial=0)))
        ds_z.append(pd.DataFrame(cumtrapz(cumtrapz(acc.iloc[600*i:600*(i+1), 2], train_time, initial=0), train_time, initial=0)))
    return (pd.concat([pd.concat(acc_x), pd.concat(acc_y), pd.concat(acc_z)], axis = 1).reset_index(drop=True),
           pd.concat([pd.concat(ds_x), pd.concat(ds_y), pd.concat(ds_z)], axis= 1).reset_index(drop = True))
```

Cumtrapz

Composite Trapezoidal Rule

- Assuming n+1 data points are evenly spaced, there will be n intervals over which to integrate.
- The total integral can be calculated by integrating each subinterval and then adding them together:

$$I=\int_{x_0}^{x_n} f_n(x)\,dx=\int_{x_0}^{x_1} f_n(x)\,dx+\int_{x_1}^{x_2} f_n(x)\,dx+\cdots+\int_{x_{n-1}}^{x_n} f_n(x)\,dx$$

$$I=(x_1-x_0)\frac{f(x_0)+f(x_1)}{2}+(x_2-x_1)\frac{f(x_1)+f(x_2)}{2}+\cdots+(x_n-x_{n-1})\frac{f(x_{n-1})+f(x_n)}{2}$$

$$I=\frac{h}{2}\left[f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)\right]$$

Cumulative Trapezoidal Numerical Integration
“누적 사다리꼴 수치 적분"

왜 넣었을까?

1. ds에는 Cumtrapz 2번 사용 2. gy 사용하지 않고 acc만 사용

가속도 적분 → 속도(acc_x.append)
속도 적분 → 위치 (ds_x.append)

https://slidetodoc.com/chapter-17-objectives-recognizing-that-newtoncotes-integration-formulas/

https://kr.mathworks.com/help/matlab/ref/cumtrapz.html

https://blog.marketmuse.com/gated-recurrent-unit-gru-definition/
https://en.wikipedia.org/wiki/Gated_recurrent_unit

Model

gru layer
+ polling layer
+ dense layer

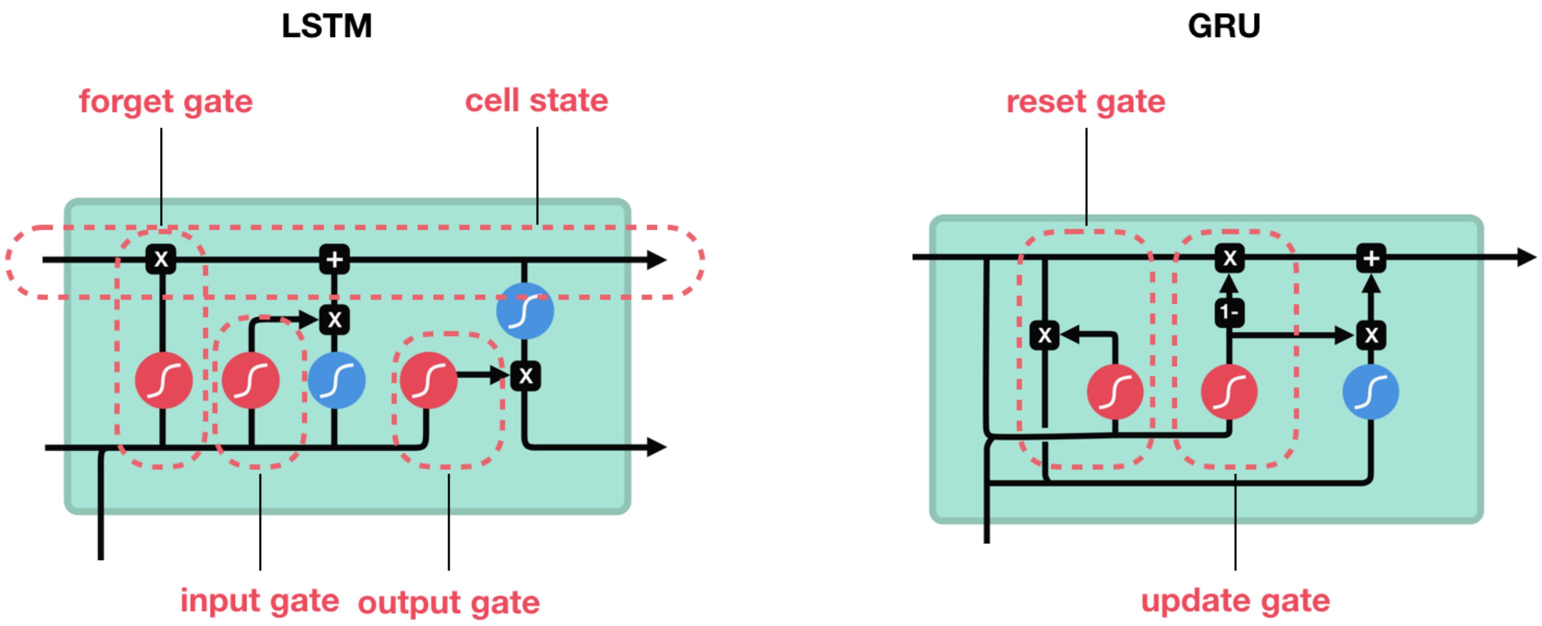
GRU Layer

RNN의 한 종류
- LSTM보다 빠름
- 메모리 효율적으로 사용
- Vanishing Gradient 문제 해결

Dense Layer

- Softmax 이용

GRU Layer



업데이트 게이트 / 리셋 게이트 두 가지 게이트 존재

- 리셋 게이트: 새로운 입력을 이전 메모리(hidden state 값)와 어떻게 합칠지 정해 줌
- 업데이트 게이트: 이전 메모리(hidden state)를 얼마나 기억할지를 정해 줌

If 리셋 게이트 == 1, 업데이트 게이트 == 0: **기본 RNN 구조**

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
https://blog.naver.com/lynhyul/222241255979

Pooling Layer

AveragePooling1D
MaxPool1d
GolbalAveragePooling1D

- Average Pooling: Batch의 Average 값이 선택됨
- Max Pooling: Batch의 Max 값이 선택됨
- Global Average Pooling: Batch의 Average Pooling 후 이를 1x1xN matrix(neural) 형태로 만듦

https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

4 models

- First Model: GRU + AveragePooling1D + GRU + AveragePooling1D + Dense Softmax

- Second Model: GRU + MaxPool1d + AveragePooling1D + Concatenate + GRU + GlobalAveragePooling1D + Dense Softmax

- Third Model: GRU + MaxPool1D + AveragePooling1D + Concatenate + GRU + GlobalAveragePooling1D + Dense Softmax

- Fourth Model: GRU + AveragePooling1D + GRU + GlobalAveragePooling1D + Dense Softmax

Seed 다르게 하여 각각 한 모델 당 두 개의 결괏값이 나오도록 만든 후,
총 8 개의 값을 평균 하여 최종 결과 제작

Tensorflow to Pytorch

https://pytorch.org/docs/stable/index.html
https://pytorch.org/docs/0.3.0/nn.html#torch.nn.functional.adaptive_avg_pool2d

	Tensorflow	Pytorch
import	import tensorflow as tf from tensorflow import keras from tensorflow.keras import layers as L from tensorflow.compat.v1 import ConfigProto from tensorflow.compat.v1 import InteractiveSession	Import torch.nn
GRU	L.GRU(256, return_sequences = True, dropout = 0.2) (inputs)	torch.nn.GRU
AveragePooling1D	L.AveragePooling1D() (gru1)	torch.nn.AvgPool1d (*kernel_size*, *stride=None*, *padding=0*, *ceil_mode=False*, *count_include_pad=True*)
GlobalAveragePooling1D	L.GlobalAveragePooling1D() (gru2)	torch.nn.AvgPool2d
MaxPool1D	L.MaxPool1D() (gru1)	torch.nn.MaxPool1d (*kernel_size*, *stride=None*, *padding=0*, *dilation=1*, *return_indices=False*, *ceil_mode=False*)
Dense Activation Softmax	L.Dense(61, activation = "softmax") (GAP)	torch.nn.Softmax(*dim=None*)