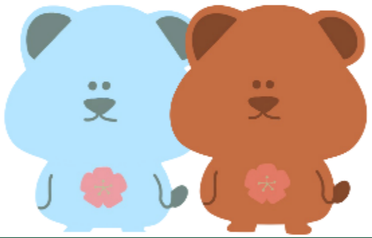
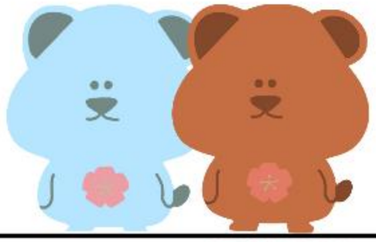
# # 산업 제어 시스템 보안 위협 탐지 AI 경진대회



EWHA EURON

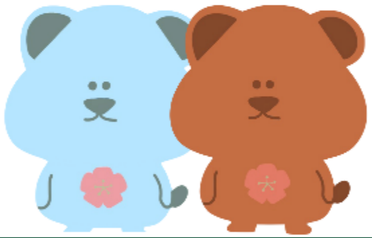
# 목차





EWHA EURON

# #01 대회 소개 & EDA



EWHA EURON

# #01. 대회 소개 & EDA

## HAI 보안데이터셋을 활용하여 AI 기반 제어시스템 보안위협 탐지

### train.csv

| | timestamp | C01 | C02 | C03 | C04 | C05 |
|---|---|---|---|---|---|---|
| 1 | 2021-07-11 10:00:00 | -2.2642 | 0 | 12.26196 | -0.00087000 | 12.01019 |
| 2 | 2021-07-11 10:00:01 | -2.4923 | 0 | 12.26196 | 0.00058000 | 12.56714 |
| 3 | 2021-07-11 10:00:02 | -2.8460 | 0 | 12.26196 | -0.00072000 | 14.48975 |
| 4 | 2021-07-11 10:00:03 | -2.1235 | 0 | 12.26196 | 0.00101000 | 15.93170 |
| 5 | 2021-07-11 10:00:04 | -2.9074 | 0 | 12.26196 | 0.00043000 | 16.10718 |

### validation.csv

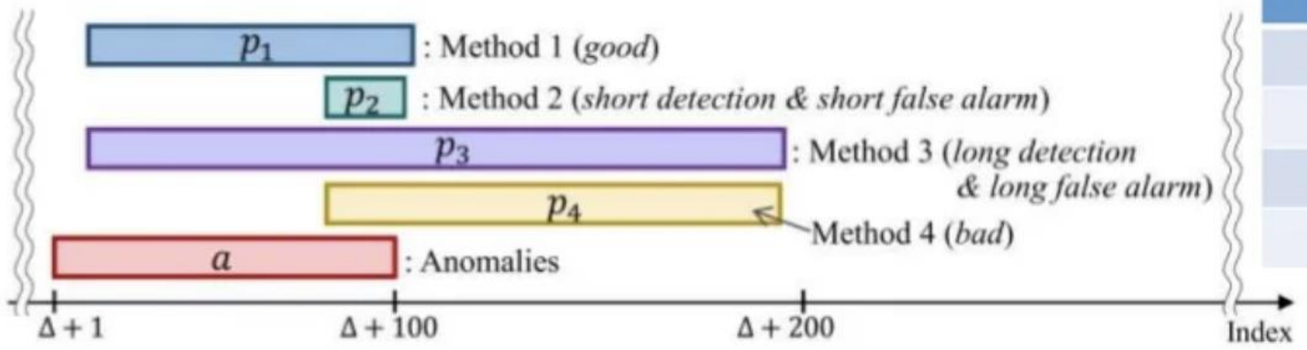
| | timestamp | | C83 | C84 | C85 | C86 | attack |
|---|---|---|---|---|---|---|---|
| 1 | 2021-07-10 00:00:01 | 1 | 1034.712769 | 12.0 | 50 | 161 | 0 |
| 2 | 2021-07-10 00:00:02 | 1 | 1034.712769 | 12.0 | 50 | 155 | 0 |
| 3 | 2021-07-10 00:00:03 | 1 | 1034.712769 | 12.0 | 50 | 149 | 0 |
| 4 | 2021-07-10 00:00:04 | 1 | 1034.712769 | 12.0 | 50 | 148 | 0 |
| 5 | 2021-07-10 00:00:05 | 1 | 1034.712769 | 12.0 | 50 | 148 | 0 |

# #01. 대회 소개 & EDA

평가식: eTaPR

- Enhanced Time-series Aware Precision (eTaP)
  - 예측 결과가 오탐 없이 이상 징후를 찾아내는가? (점수: 0~1)

- Enhanced Time-series Aware Recall (eTaR)
  - 얼마나 다양한 이상 징후를 찾아내는가? (점수: 0~1)

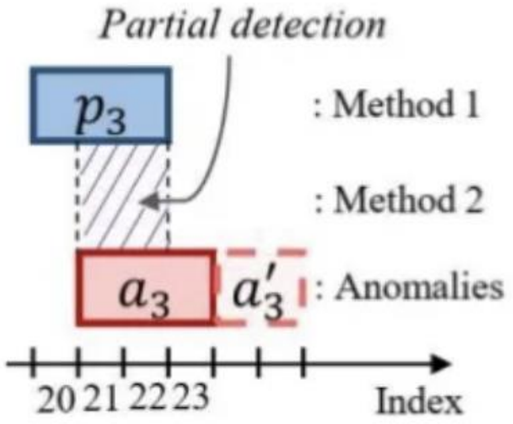



| 예측 결과 | eTaP | eTaR |
|---|---|---|
| $p_1$ | High | High |
| $p_2$ | High | Low |
| $p_3$ | Low | High |
| $p_4$ | Low | Low |

# #01. 대회 소개 & EDA

평가식: eTaPR

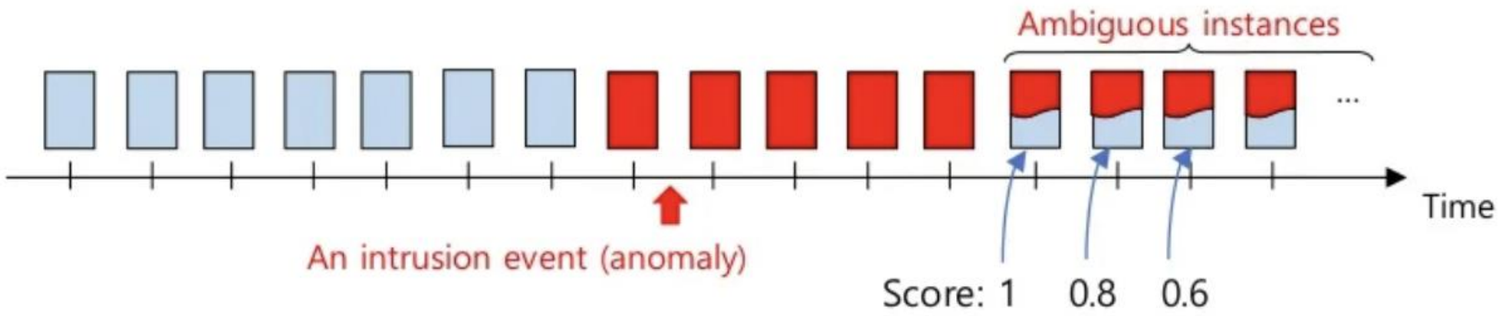
- 총 4가지 파라미터 $(\alpha, \delta, \pi, \rho)$ 설정 필요
- $\alpha$ (0.0 ~ 1.0)
  - Detection score와 portion score의 비율 조절
    - Partial detection에 대해 두 가지 점수로 계산
      - Detection score: 일부만 맞춰도 다 맞춘 것과 동일하게 간주 ($p_3$ 예: 1.00)
      - Portion score: 맞춘 비율만큼만 점수 부여 ($p_3$ 예: 0.66)
  - 높은 값일수록 detection score를 높게 설정



# #01. 대회 소개 & EDA

평가식: eTaPR

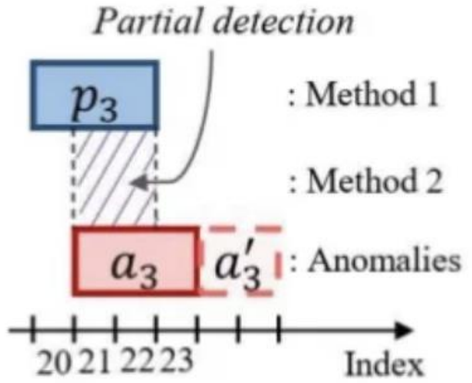
- $\delta$ (0 이상의 정수)
  - 공격을 수행하였을 때, 시스템에 그 영향이 얼마나 남아있는지 모름으로 인해 발생
  - 공격으로 라벨된 이후의 범위를 탐지하여도 어느 정도 점수를 인정
    - 인정할 범위를 결정하는 파라미터



# #01. 대회 소개 & EDA

평가식: eTaPR

- $\pi$ (0.0 ~ 1.0)
  - eTaP의 detection score에 필요
    - 각 prediction 중 anomaly와 어느 정도 겹쳐야 점수가 인정될지 설정하는 파라미터
    - $p_3$의 점수 예: 1점 ($\pi$가 0.66 이하) 0점 ($\pi$가 0.66 이상)
- $\rho$ (0.0 ~ 1.0)
  - eTaR의 detection score에 필요
    - 각 anomaly 중 어느 정도가 탐지 되어야 점수가 인정될지 설정하는 파라미터
    - $a_3$의 점수 예: 1점 ($\rho$가 0.66 이하) 0점 ($\rho$가 0.66 이상)

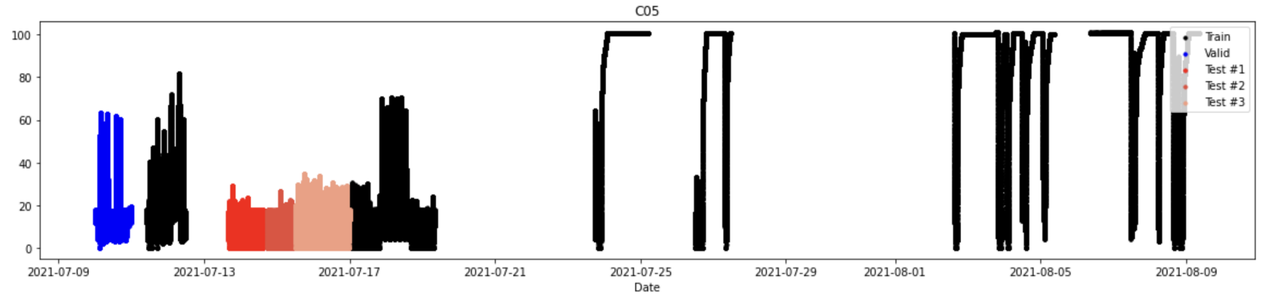


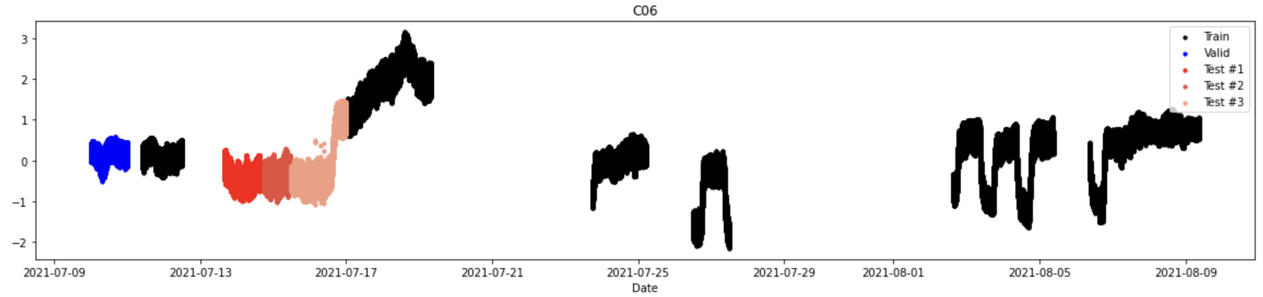
# #01. 대회 소개 & EDA

## EDA: 데이터 순서와 데이터 분포 확인

```
Valid #1 | START: 2021-07-10 00:00:01 | END: 2021-07-11 00:00:00 | DATA POINT: 86400
Train #1 | START: 2021-07-11 10:00:00 | END: 2021-07-12 12:00:00 | DATA POINT: 93601
Test  #1 | START: 2021-07-13 16:00:01 | END: 2021-07-14 15:00:00 | DATA POINT: 82800
Test  #2 | START: 2021-07-14 17:40:01 | END: 2021-07-15 11:00:00 | DATA POINT: 62400
Test  #3 | START: 2021-07-15 12:00:01 | END: 2021-07-17 00:00:00 | DATA POINT: 129600
Train #2 | START: 2021-07-17 00:00:01 | END: 2021-07-19 08:00:00 | DATA POINT: 201600
Train #3 | START: 2021-07-23 18:00:01 | END: 2021-07-25 05:00:00 | DATA POINT: 126000
Train #4 | START: 2021-07-26 12:00:00 | END: 2021-07-27 12:00:00 | DATA POINT: 86401
Train #5 | START: 2021-08-02 15:00:01 | END: 2021-08-05 09:00:00 | DATA POINT: 237600
Train #6 | START: 2021-08-06 09:00:01 | END: 2021-08-09 09:00:00 | DATA POINT: 259200
```

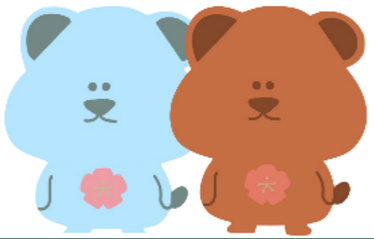
→ 테스트 데이터 전후로 훈련 데이터 분포




7월 20일 이후로 데이터 분포가 달라짐 ←

# #02 1등 솔루션: Conv1D + LSTM + AutoEncoder



EWHA EURON

# #01. 접근 논리 및 개요

1. **하나**의 베이스 모델 활용

2. 변수 간의 상관계수 분석 -> **4 그룹 단위** 분석

3. **데이터 전처리**

   (1) Normalization

   (2) Boundary Check

   (3) Threshold & Scaling

```python
def normalize(df):
    ndf = df.copy()
    for c in df.columns:
        if TAG_MIN[c] == TAG_MAX[c]:
            ndf[c] = df[c] - TAG_MIN[c]
        else:
            ndf[c] = (df[c] - TAG_MIN[c]) / (TAG_MAX[c] - TAG_MIN[c])
    return ndf
```

```python
def boundary_check(df):
    x = np.array(df, dtype=np.float32)
    print(x)
    return np.any(x > 1.0), np.any(x < 0), np.any(np.isnan(x))
```

```python
def flatten(X):
    flattened_X = np.empty((X.shape[0], X.shape[2]))  # sample x features array.
    for i in range(X.shape[0]):
        flattened_X[i] = X[i, (X.shape[1]-1), :]
    return(flattened_X)

def scale(X, scaler):
    for i in range(X.shape[0]):
        X[i, :, :] = scaler.transform(X[i, :, :])

    return X
```

EWHA EURON

# #02. 속성 간의 상관계수 분석

- pandas.DataFrame.corr() Function

```
for col in valid.columns:
  print("=======================================")
  print(a_val.corr()[col][a_val.corr()[col] > 0.7])
  print("=======================================")
```

`DataFrame.corr(method='pearson', min_periods=1)` [source]

Compute pairwise correlation of columns, excluding NA/null values.

**Parameters:** **method** : *{'pearson', 'kendall', 'spearman'} or callable*

Method of correlation:

- pearson : standard correlation coefficient
- kendall : Kendall Tau correlation coefficient
- spearman : Spearman rank correlation
- **callable: callable with input two 1d ndarrays**

and returning a float. Note that the returned matrix from corr will have 1 along the diagonals and will be symmetric regardless of the callable's behavior.

**min_periods** : *int, optional*

Minimum number of observations required per pair of columns to have a valid result. Currently only available for Pearson and Spearman correlation.

**Returns:** **DataFrame**

Correlation matrix.

## Correlation (상관계수) vs Covariance (공분산)

$$\rho = \frac{Cov(X,\ Y)}{\sqrt{Var(X)\ \ Var(Y)}}, \quad -1 \leq \rho \leq 1$$

$$E(X) = \mu,\ E(Y) = \nu$$

$$Cov(X,\ Y) = E((X - \mu)(Y - \nu))$$

$$\because E((X - \mu)(Y - \nu)) = E(XY - \mu Y - \nu X + \mu\nu)$$
$$= E(XY) - \mu E(X) - \nu E(Y) + \mu\nu$$
$$= E(XY) - \mu\nu$$

$$Cov(X,\ Y) = E(XY) - \mu\nu$$

# #03. 구현

1. TF Layer (1) **Bidirectional**

2. TF Layer (2) **RepeatVector**

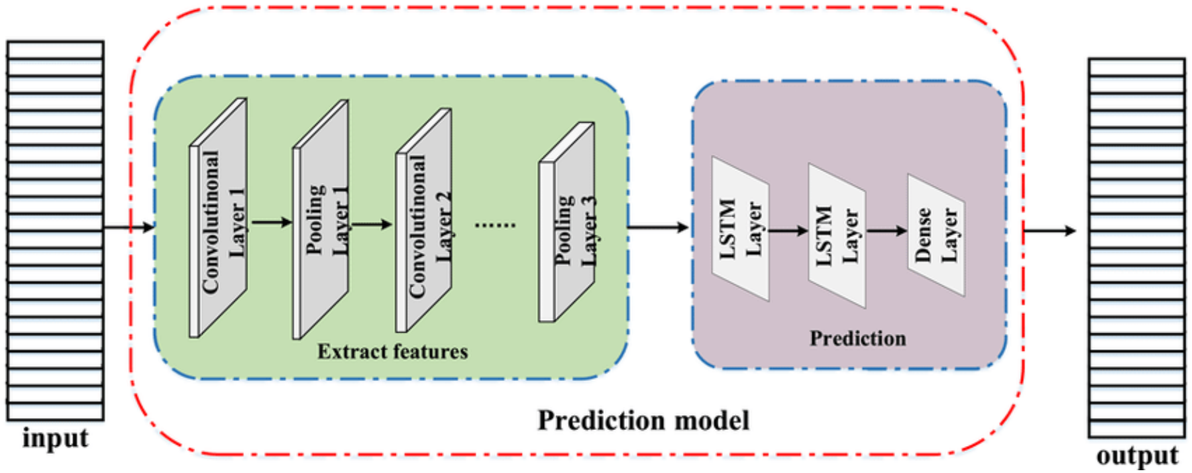
3. TF Layer (3) **TimeDistributed**

4. **Adam** Optimizer / **MSE** Loss

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d (Conv1D) | (None, 1, 128) | 393344 |
| dense (Dense) | (None, 1, 128) | 16512 |
| bidirectional (Bidirectional | (None, 128) | 98816 |
| repeat_vector (RepeatVector) | (None, 1, 128) | 0 |
| bidirectional_1 (Bidirection | (None, 1, 128) | 98816 |
| dense_1 (Dense) | (None, 1, 128) | 16512 |
| conv1d_1 (Conv1D) | (None, 1, 128) | 786560 |
| time_distributed (TimeDistri | (None, 1, 64) | 8256 |

# #04. Conv1D vs LSTM

1. Conv1D : 입력 시간의 평활화로 인한 이동 평균값을 입력 feature로 사용할 필요 X
2. LSTM : 여러 입력 변수로 모델링 가능




- 비지도 학습 기반 (Auto-Encoder)

  (1) 초반의 time-series를 통해서 이후 예측

  (2) RMSE(Ground Truth, Predicted) = Anomaly Score

- 4개의 군집 중 하나라도 “anomal time series”라고 하면 이상치로 간주

```
for i in range(len(submission)):
    if group3_3['attack'][i] == 1 or group4_2['attack'][i] ==1 or group5_1['attack'][i] == 1 or group6_1['attack'][i] == 1:
        pred_y_test[i] = 1
```

# #05. Moving Average & Threshold Set

1. **단순 이동 평균** 사용

   : 방향성을 갖고 움직이면서 구해지는 평균 값

   - Window size (위에서는 30) 이전 데이터는 평균에 영향 X

```
#이동평균
mean_window = error_df['Reconstruction_error'].rolling(30, center=True).mean()
window_error = mean_window.fillna(0)
```

2. **Threshold**를 군집별로 정하기

   - auto_threshold에서의 값들을 다 적용
   - Accuracy가 높은 threshold를 군집마다 적용

```
def check_graph(xs, att, piece=2, THRESHOLD=None):
    l = xs.shape[0]
    chunk = l // piece
    fig, axs = plt.subplots(piece, figsize=(20, 4 * piece))
    pred_y_test = []
    for i in range(piece):
        L = i * chunk
        R = min(L + chunk, l)
        xticks = range(L, R)
        axs[i].plot(xticks, xs[L:R])

        #Threshold 설정
        pred_tem = [1 if e > THRESHOLD[i] else 0 for e in list(xs[L:R])]
        pred_y_test.extend(pred_tem)

        #SUBPLOT
        if len(xs[L:R]) > 0:
            peak = max(xs[L:R])
            axs[i].title.set_text(i)
            axs[i].plot(xticks, att[L:R] * peak * 0.3)
            axs[i].set_ylim(0,0.00005)
        if THRESHOLD!=None:
            axs[i].axhline(y=THRESHOLD[i], color='r')
    plt.show()

    pred_y_test
    pred_y_test = np.array(pred_y_test)
    return pred_y_test
```

```
auto_threshold = [
0.0000045,0.0000045,0.0000045,0.0000045,0.0000045,
0.0000045,0.0000045,0.0000045,0.0000045,0.0000045,
0.0000045,0.0000045,0.0000045,0.0000045,0.0000045,
0.0000045,0.0000045,0.0000045,0.0000045,0.0000045]
```

# #03 4등 솔루션: MLP Mixer + SAM

EWHA EURON

# #03-1 MLP Mixer + SAM

## Model Selection

- Bidirectional RNN
  GRU, LSTM 사용. Valid에서 매우 낮은 점수, 테스트에서 상대적으로 높은 점수를 보임

- CNN
  1D CNN을 이용해 만든 VGG style 모델은 전혀 학습을 하지 못함. RNN+CNN도 마찬가지

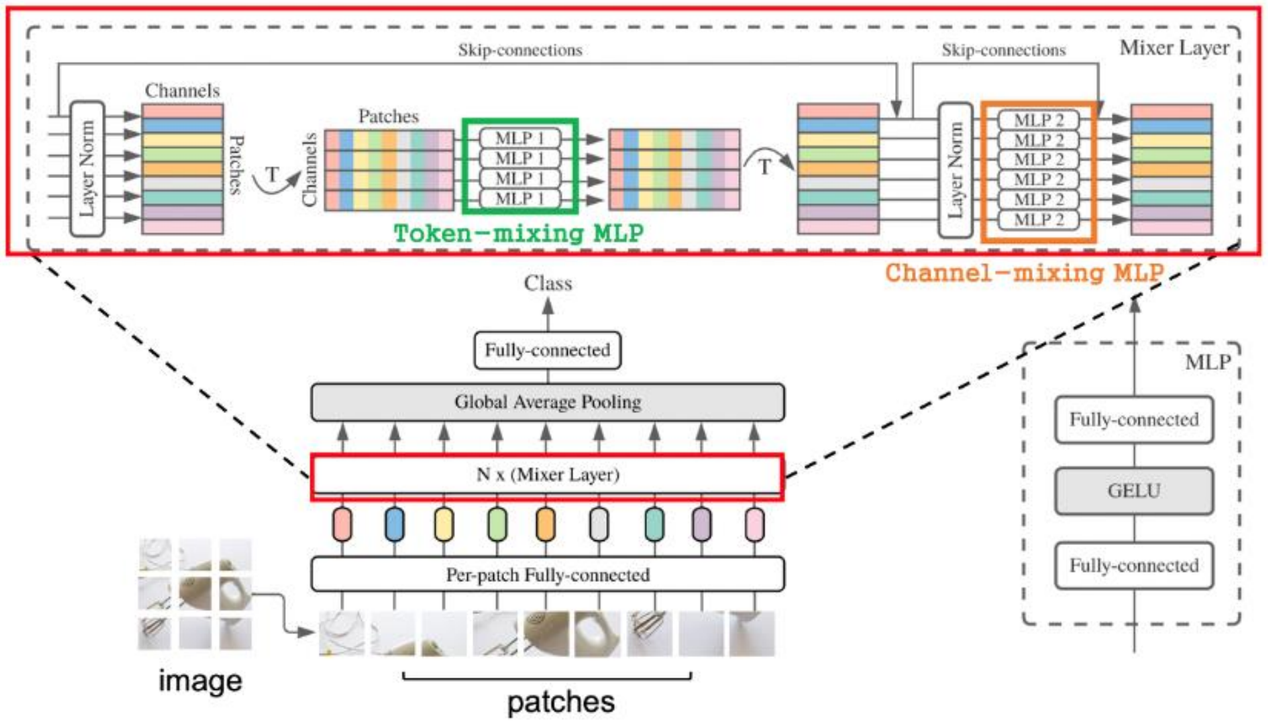
- Transformer
  인코더만 만들어서 사용 (예측하고자 하는 Timestep이 한 개)
  점수가 눈에 띄게 상승 (전체 시퀀스를 한 번에 볼 수 있기 때문)
  입력 시퀀스의 길이가 길어 시간이 오래 걸림.

- MLP-Mixer
  Transformer와 같이 한 번에 전체 시퀀스를 다룰 수 있으며, 더 가볍고 빠름

# #03-1 MLP Mixer + SAM

MLP Mixer

- DNN의 가장 기본적인 알고리즘인 MLP만을 이용한 이미지 인식.
- 패치의 수가 증가해도 계산량이 선형적으로만 증가.
  → ViT와 비슷한 성능에도 훨씬 빠른 속도



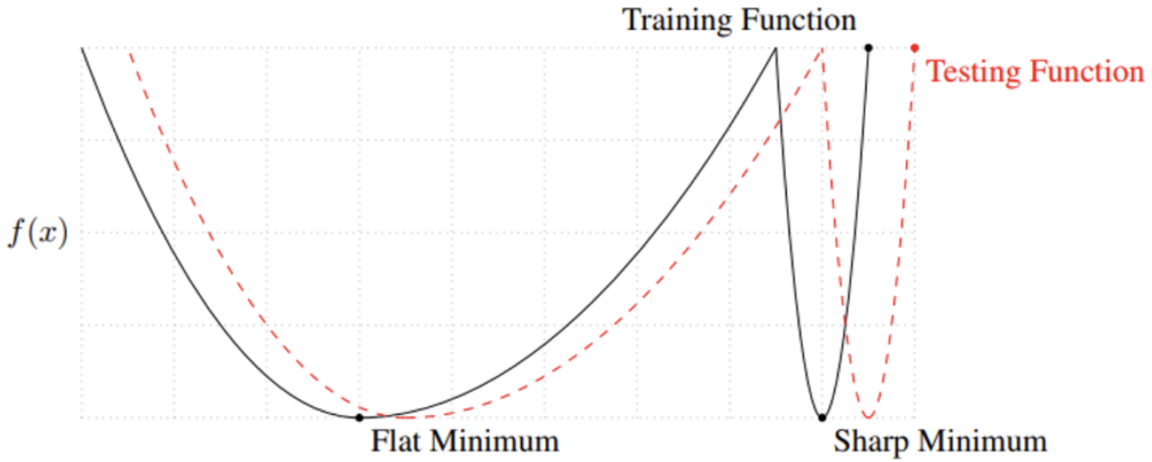

1. 입력받은 이미지를 S개의 Image Patch로 나눔.
2. Image Patch를 크기가 C인 token으로 만듦
3. token을 Mixer Layer에 넣음
   a) Token-mixing MLP
   b) Channel-mixing MLP
4. Global Average Pooling
5. FC

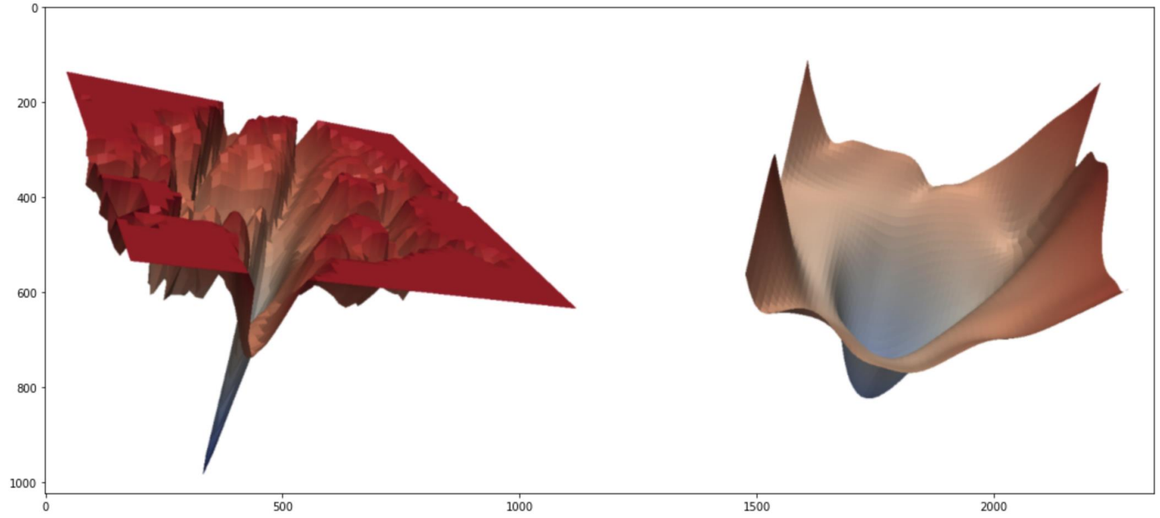
# #03-1 MLP Mixer + SAM

SAM: Sharpness-Aware Minimization Optimizer

- training loss 값을 단순히 줄이는 것만이 아니라 generalization 성능과 관련이 있다고 여겨지는 loss의 sharpness도 고려하여 최적화

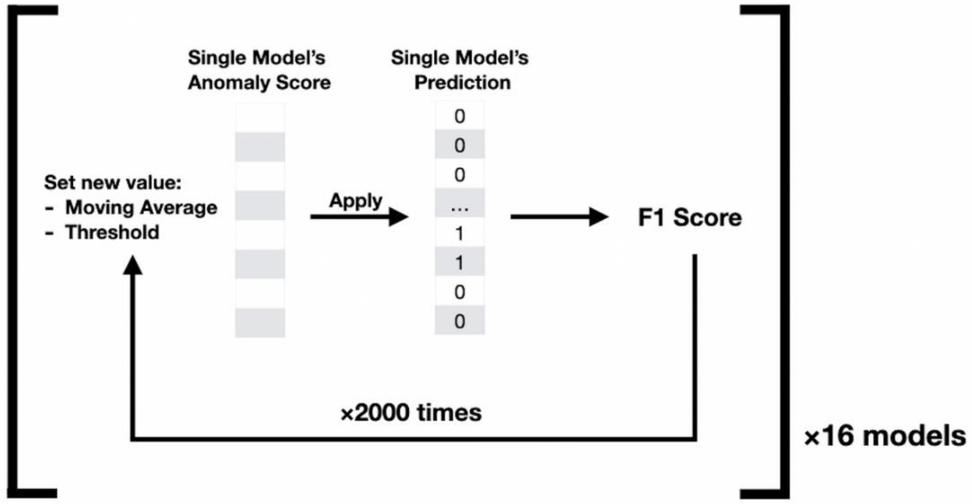
- 아래의 이미지와 같이 모델의 학습과정에서 local minima 들이 조금 더 스무스해짐



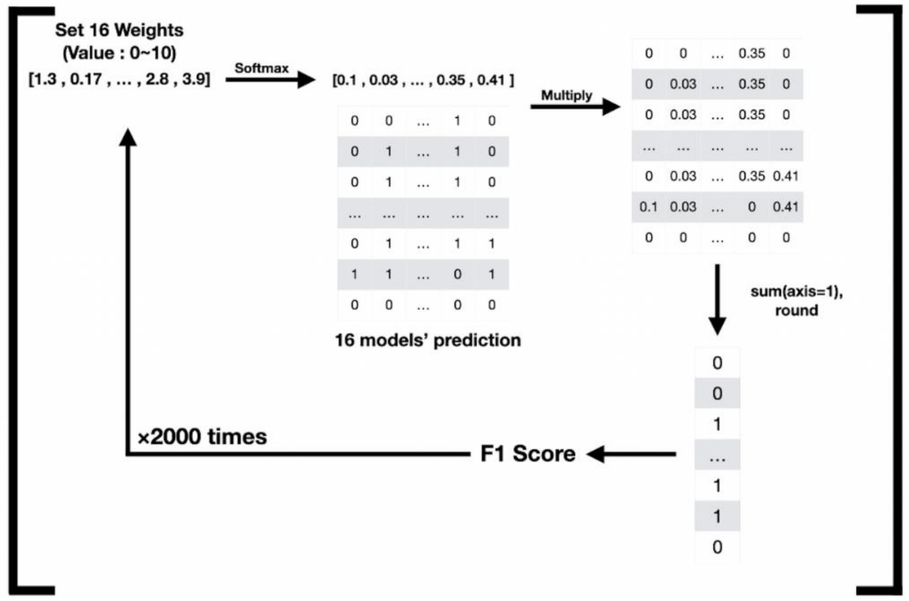
# #03-2 Threshold

## <Bayesian Optimization 1>



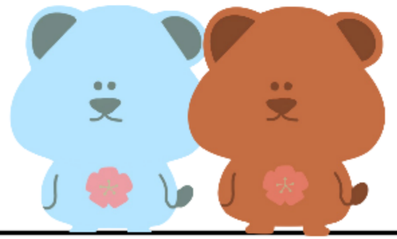

* Returns : 16 moving average value, 16 thresholds
* Apply to test set.

## <Bayesian Optimization 2>




* Returns : Softmax(Weight vector with 16 values.)
* Apply to test set.

# THANK YOU



EWHA EURON