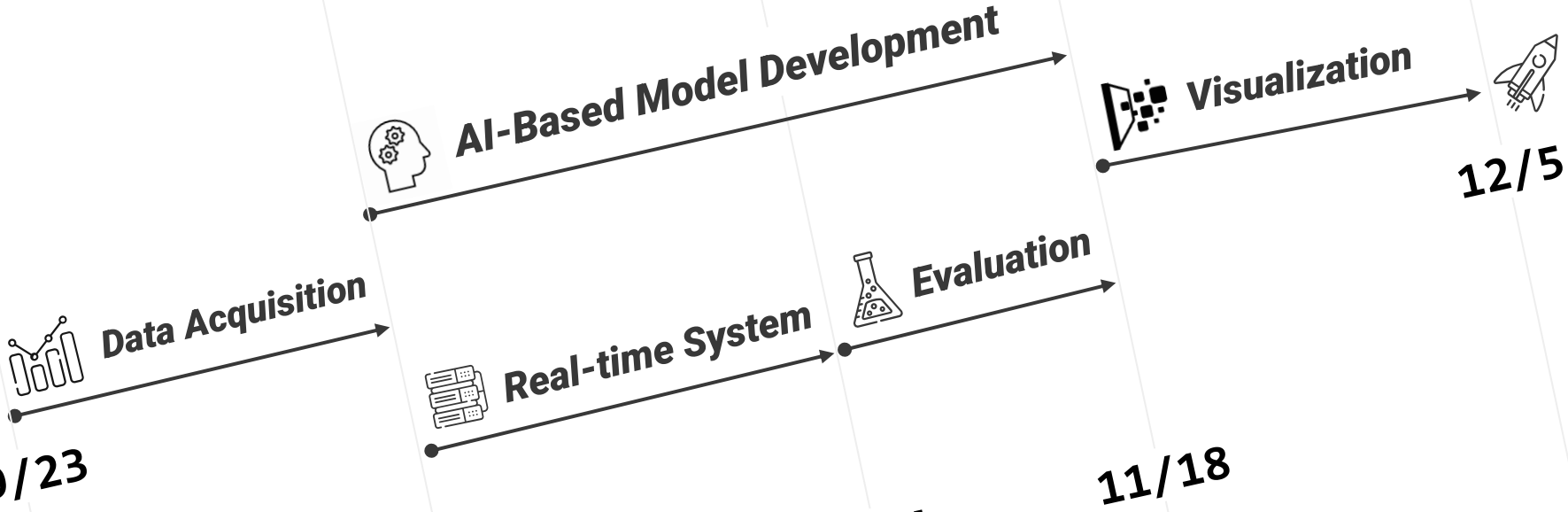
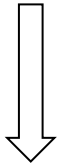
# Timeline


Data Acquisition
AI-Based Model Development
Real-time System
Evaluation
Visualization
9/23
10/4
11/4
11/18
12/5

# Visualization

- Two ways of Visualization

1. **Grad CAM**
2. **Attention of Transformer**



We can not use the both method 😢

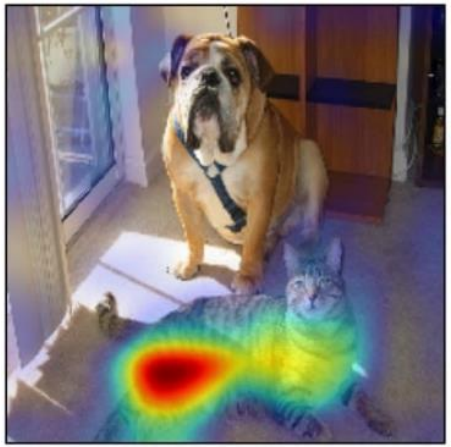
**Why?**

# Visualization

- Grad CAM

Original Paper : Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Grad-CAM was originally invented for 2D-Data which must be propagated through 2D-ConvNet.



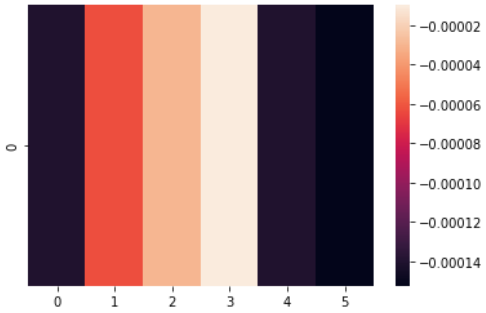
# Visualization

## ❑ Grad CAM

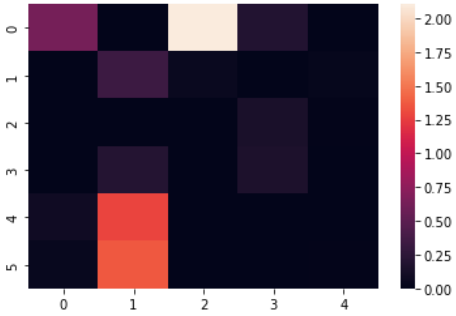
Our Data Shape : N X 6 X 8

**N : data number, 6 : time, 8 : features**

- We use 1D-CNN, so our grad-CAM result is also 1-dimensional.




- What we actually get (1D Heatmap)




- What we want to get (2D Heatmap)

# Visualization

## ❑ Grad CAM

Then, should we use 2D-ConvNet?

1. The model lose its accuracy for prediction because we treat each features with filter, not treating them as independents channels.

1. Since we use too small 2D dataset (6X8), we can not resolve the problem of low resolution.

➢ Low resolution problem arises because 2D-ConvNet continually shrinks its feature map size.

(e.g) Input data size: 224X224X3 **>>>** At the end of 2D-ConvNet14X14X1024)

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

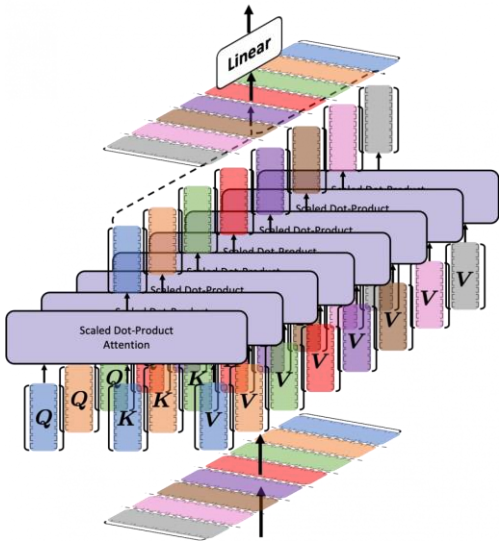
$$L_{\text{Grad-CAM}}^c = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

# Visualization

## ❑ Attention

Same problem with 2D-grad-CAM.

➢ Since attention aggregates feature dimension, we can not get 2-dimensional important weights.



# Visualization

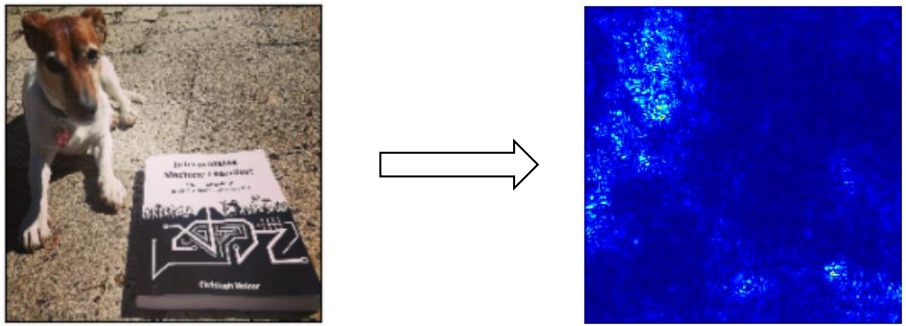
## ❑ Alternative visualization (Traditional Pixel Attribution)

We can get what we want more simpler way. Just get gradient saliency map !

➢ Many visualization methods have been designed for 2D-data, ConvNet. However, we use very simple and low dimensional time-series data.

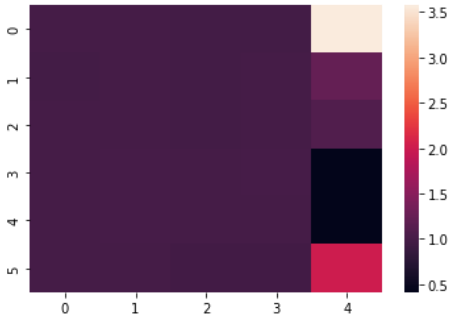
❑ Method

1. Feedforward data with pre-trained model
2. Backpropagate its gradient with maximum $Y^c$ score over each data.
3. Regard the gradients as important weights

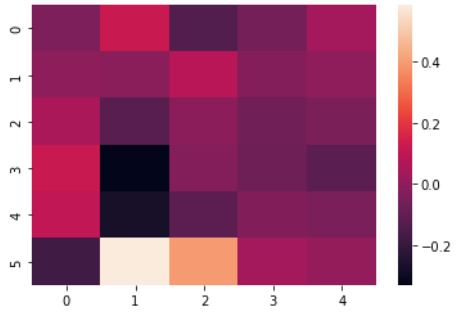


# Visualization

❏ Result




➢ Input data

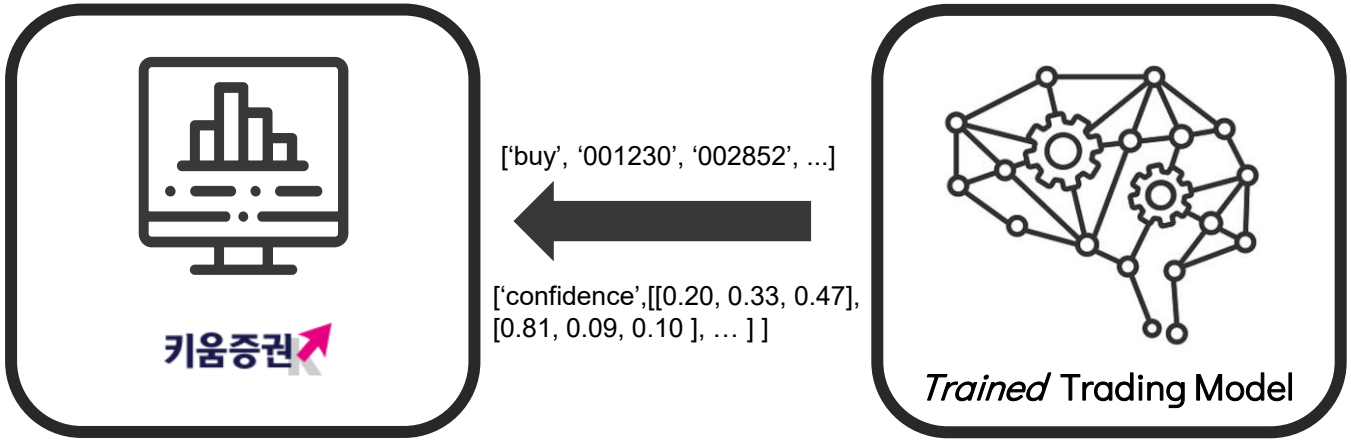



➢ Its gradient

Succeed to easily retrieve 2D-gradient for each data. 😊

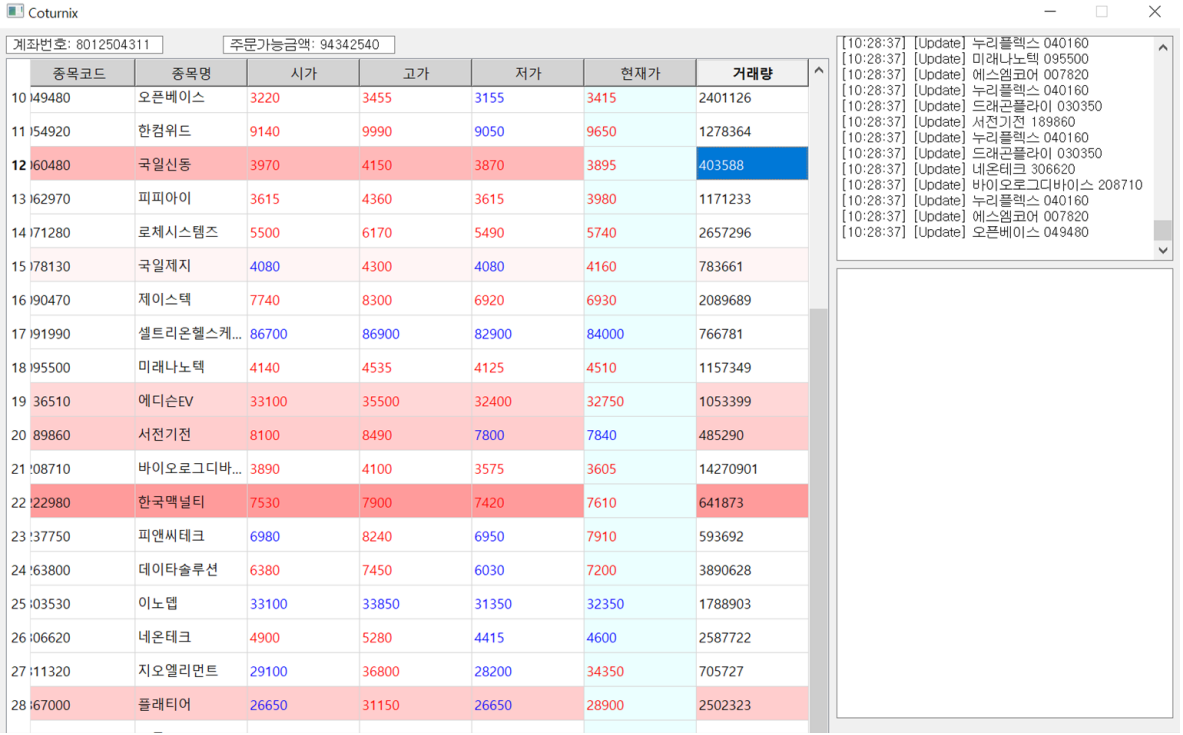
# Real-time System

- ❑ AI model 서버에서 client로 Decision Confidence 송신
  - ➔ [0, 1, 2]의 출력 (하향 예측, 횡보 예측, 상향 예측)을 가지며 결과가 2일 경우 해당 종목 코드를 기록
  - ➔ 각 class에 대한 confidence를 기록
- ❑ 기록된 종목코드를 'buy' tag와 같이 GUI client로 송신
- ❑ 기록된 confidence를 'confidence' tag와 같이 GUI client로 송신



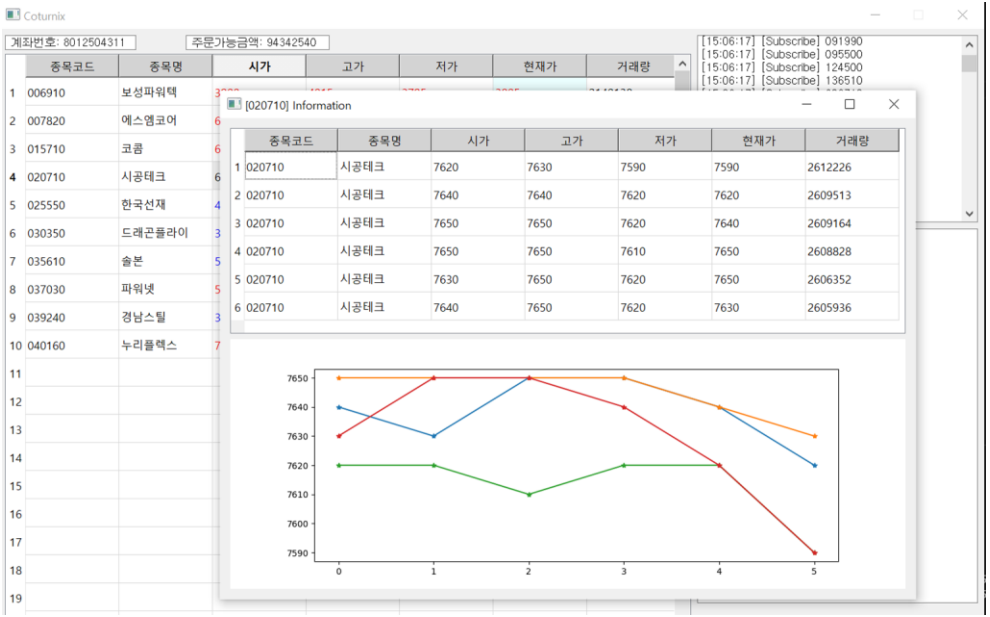
# Real-time System

❑ Client는 confidence 수신 후 class 2 [buy]의 confidence에 따라 배경색 변경



# Real-time System

- ❑ 종목 더블 클릭시 상세 정보 및 그래프 출력
- ❑ (예정사항) 이후 input에 대한 heatmap 수신시 Thread 를 통한 실시간 갱신



# Real-time System

- Demo Video