

## III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

I am a machine learning engineer, in my free time I participate in data science contests to learn new techniques and improve my data science skills.

2. What motivated you to compete in this challenge?

I want to gain new knowledge in this domain.

3. High level summary of your approach: what did you do and why?

My solution is an ensemble of variants of 2 network architectures, both architectures are designed to detect the presence of targets in each time step and in the whole sample. The first architecture is based on 1D CNN-Transformer where each sample is transformed into a sequence of 1D arrays before feeding into a 1D Event Detection network. The second architecture is based on 2D CNN where 3 different preprocessing methods are applied to the sample to generate a 3 channel image-like representation, then this image-like representation is fed into a 2D Event Detection network.

4. Do you have any useful charts, graphs, or visualizations from the process?

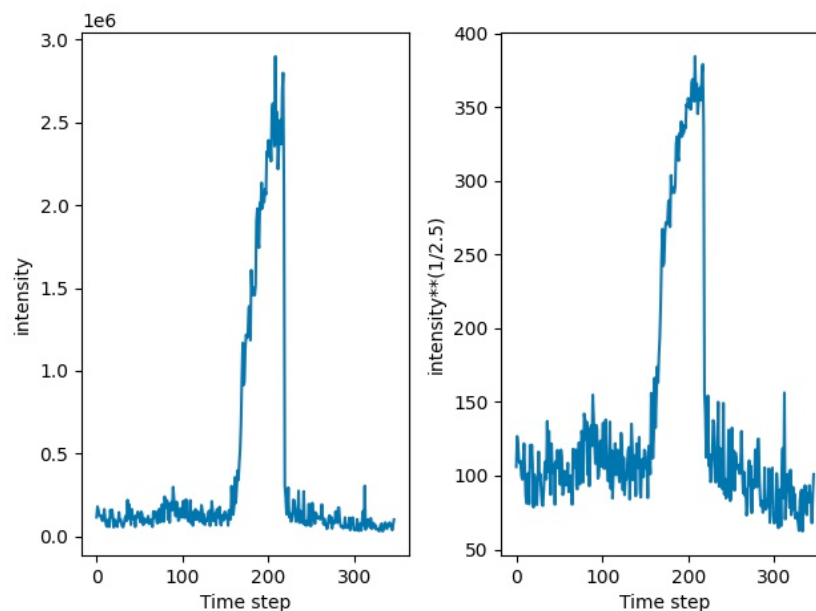


Figure 1. Normalizing the intensity of sample S0000 with mass=3, scale\_factor=2.5. The raw intensity value is on the left; the normalized intensity value on the right. The normalization helps transform the data from range [0; 3e6] to a lower range [50;400] while the peak characteristics do not change much.

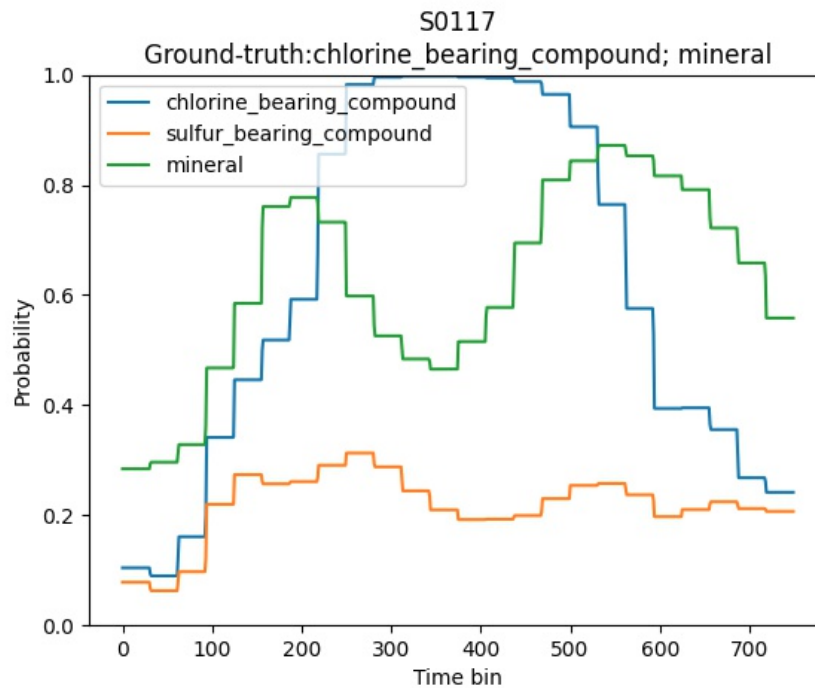


Figure 2. Event Detection of sample S0117. The proposed approach is not only capable of detecting the presence of the chemical compounds in the sample, but it is also capable of detecting the time when the ions are emitted during the GCMS process.

- Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

The following piece of code normalizes the intensity to a lower standard deviation while remaining the peak characteristics. It helps the model easier to capture unique features of each sample.

```
if cfg.normalize == 1:
    power_r = cfg.power_r
    if self.val == 0 and random.random() < cfg.power_prob:
        power_r += 0.01*random.randint(-20,20)

    if cfg.use_meta:
        features[:, :-2] = features[:, :-2]**(1/power_r)
    else:
        features = features**(1/power_r)
```

The following piece of code randomly adds a random background level to the sample. It helps the model be more robust and avoid overfitting.

```
if self.val == 0 and random.random() < cfg.aug*cfg.aug_prob:
# if 0:
    # noise = features*0.1*random.random()
    noise = random.randint(-100,100)
    if cfg.use_meta:
        features[:, :-2] = features[:, :-2] + noise
    else:
        features = features + noise
```

The following piece of code randomly applies mixup augmentation on each batch. It helps the model be more robust and avoid overfitting.

```
#mixup
if random.random() < cfg.mixup_prob:
    indices = torch.randperm(x.size(0))
    shuffled_data = x[indices]
    shuffled_targets = y[indices]

    # if random.random()<0.5:
    if 1:
        x = 0.5*x + 0.5*shuffled_data
    else:
        x = torch.max(x,shuffled_data)

    y = torch.max(y, shuffled_targets)
#~mixup
```

6. Please provide the machine specs and time you used to run your model.
  - CPU (model): Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz
  - GPU (model or N/A): 1xRTX3090
  - Memory (GB): 32Gb
  - OS: ubuntu
  - Train duration: 10 days
  - Inference duration: 3 hours
7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?  
Each model is trained in several rounds with a set of commands, a set of commands must be run sequentially. There are 10 models total with 10 set of commands, each model can be run separately on different GPUs for faster training.
8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?  
Only the published benchmark.
9. How did you evaluate performance of the model other than the provided metric, if at all?  
I check AUC and accuracy as well, but only rely on the provided metric.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

LGB, Logistic regression, image classification approach.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

In the last few days of the challenge, I realized that my code is not optimized yet, my training code should be run 2 times faster after optimization. The parameters are not carefully tuned yet. A hybrid approach that combines 1D and 2D representation should be considered as well.