1st place write up

Mars Spectrometry 2: Gas Chromatography

Introduction

In this challenge, we are asked to build a model to automatically analyze gas chromatography-mass spectrometry (GCMS) data to detect the presence of nine families of chemical compounds related to understanding Mars' potential for past habitability.

The 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.

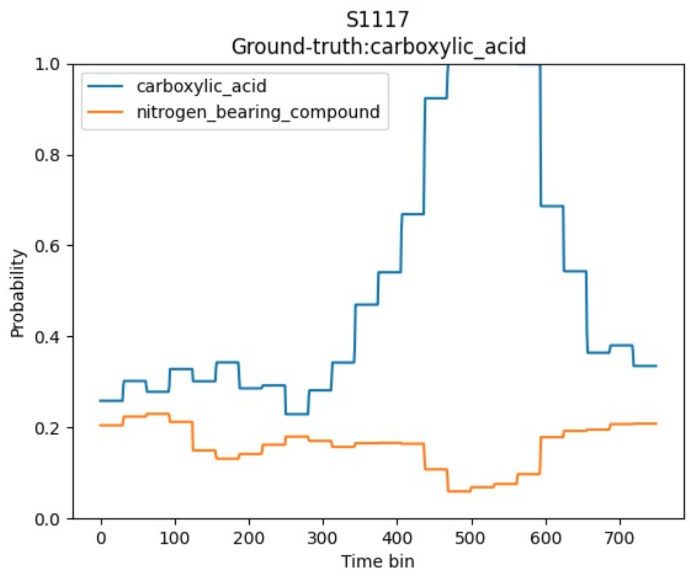

Figure 1. Detection result of sample S1117. We can see that this sample contains carboxylic_acid and its ions are emitted at around 480-530th time bin which is around 24-26th minutes during the GCMS process.

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.

Data processing

The standard deviation of the raw intensity is large, this makes the model harder to learn the features of the data, so I normalize the intensity value by *intensity = intensity**(1/scale_factor).* By doing so, the standard deviation is reduced significantly while the peak characteristics do not change much as shown in the figure 2.

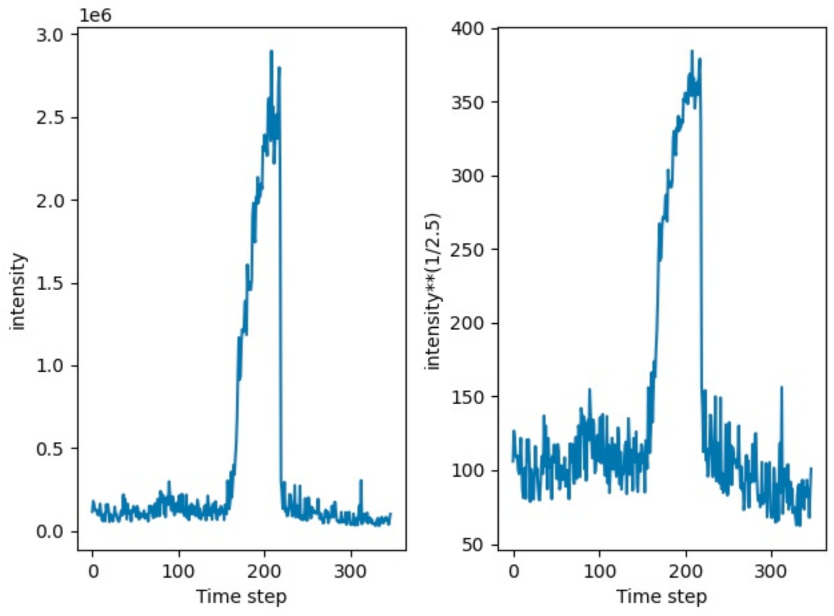

Figure 2. Normalizing the intensity of sample S0000 with mass=3, scale_factor=2.5. The raw intensity value is on the left; the normalized intensity value on the right.

The mass value is rounded to the nearest integer number and trimmed at value 500.

1D array representation: The time value is trimmed at value 50 and is divided into non-overlapping bin, then the max intensity of mass values within each bin are used. With bin_size = 0.05 minutes, each sample is divided into 999 time bin ([0:0.05], (0.05:0.10], ... , (49.05:50.00]) or 999 sequences. A sequence is a 1D array with 500 elements i.e 500 mass values. Then each sample is transformed into a 2D array with shape (number_of_sequence

* number_of_mass) or (S * M).

Image-like representation: By applying 3 different normalization scale factor on 1D representation and stack the results on top of each other, I get an array with shape (S * M * 3). I call this an image-like representation, since it has 3 channels like an RGB image.

Models

1D Event Detection network: The architecture includes a stem block (a fully connected layer and batch normalization layer), 3 Residual-like LSTM (R-LSTM) blocks and 3 CNN-Transformer (CNN-Tr) blocks are used to extract useful features. A Transformer and a LSTM are then used to aggregate features before passing it to the Event Detection head. The head includes Pooling, Conv1D and FC layers to output the presence of targets in each time bin (ED output). A Conv1D with softmax attention is applied on the ED output to yield a classification output for the whole sample.

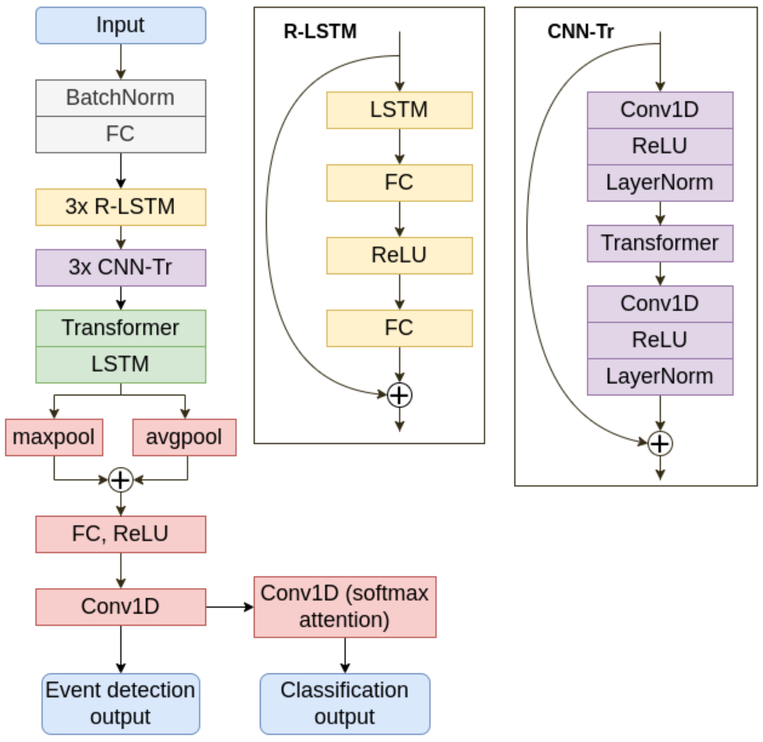

Figure 3. 1D based - Event Detection architecture.

2D Event Detection network: Similar to the idea of the 1D version, the 2D version uses a 2D CNN backbone to extract useful features before the Event Detection head. Since the input of this model is an image-like representation, I utilize the pretrained backbone from image classification architecture, specifically, I use Efficientnet backbone.

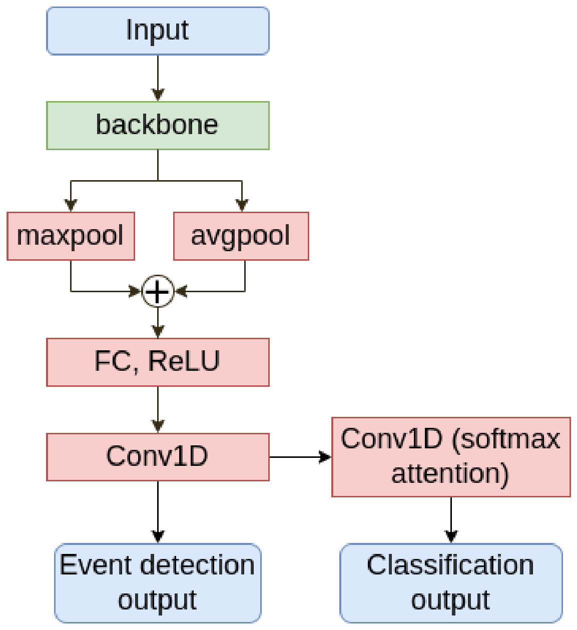

Figure 4. 2D based - Event Detection architecture.

Data augmentation

To make the model more robust and avoid overfitting, I randomly add a random background level to the sample, randomly apply mixup, and dropout during training.

I proposed a progressive augmentation method to help the model learn better. Inspired by the fact that we learn better if we start from easier lessons to harder lessons, I start training the model with very few and light augmentation, then keep adding more and heavier augmentation after epochs.

Experiments and Results

The models are trained with a varied set of parameters (sequence length, mass value, bin size, etc) to create more diversity. Then the final results are ensemble by averaging all output probability.

In table 1, models starting with 2D are 2D-based, b5, b6, b7 are the version of efficientnet architecture. 1D is 1D-based, _S, _M, _L are 3 variants of the architecture.

Table 1. Results of some models on the validation data

Model	**Bin size**	**Sequence length**	**Mass size**	**epoch**	**Agg Log loss↓**
2D_b5	0.05	800	500	150	0.1520
2D_b5	0.05	750	430	150	0.1509
2D_b5*	0.05	750	430	130	0.1455
2D_b6	0.05	750	400	130	0.1456
2D_b7	0.05	700	400	150	0.1476
1D_S	0.05	750	350	180	0.1509
1D_M	0.05	700	350	210	0.1435
1D_M	0.06	700	350	170	0.1465
1D_M*	0.06	700	350	190	0.1446
1D_L	0.06	700	350	190	0.1389

** model train with different folds split.*

The submission file is an ensemble of 10 models over 6 folds, each fold use two or three checkpoints at different epochs.