Problem statement:
In previous experiments, we have obtained the captions of traffic scenario images, which are supposed to reflect whether the image is dangerous or safe. A naive idea is to send the caption to a per-trained model such as llama2, and then generate the response by applying appropriate instruction and few shots. However, it turns out that the pre-trained model may not be very effective in processing the caption description in traffic domain, which thus restrict the performance on our specific task.

Fine-tuning strategy:
To improve the performance on determining whether the scenario is safe or not with limit numbers of datasets, we fine-tuned a Bert model and qwen1.5-1.8b-chat LLM model using PEFT to predict the result, which is compared with baseline models. To some degree, if fined-tuned models could achieve better performance on validation datasets, the caption about images are valuable to be adopted.

BERT
Fine-tune a pre-trained 'bert-base-uncased' model with Lora, which has 110 million parameters in total. Lora is a parameter-efficient fine-tuning method, which tracks changes to weights instead of updating weights directly, instead, it will decompose the large matrix of weight changes in two row-rank matrices so that the number of trainable parameters will be reduced dramatically.

Basic Process:
1. Concatenate caption with corresponding label and build the map between the label and prediction token (0: 'dangerous'; 1:'safe')

2. Split into training set and validation set (9:1). Finally, training set has 890 samples, and validation set has 99 samples

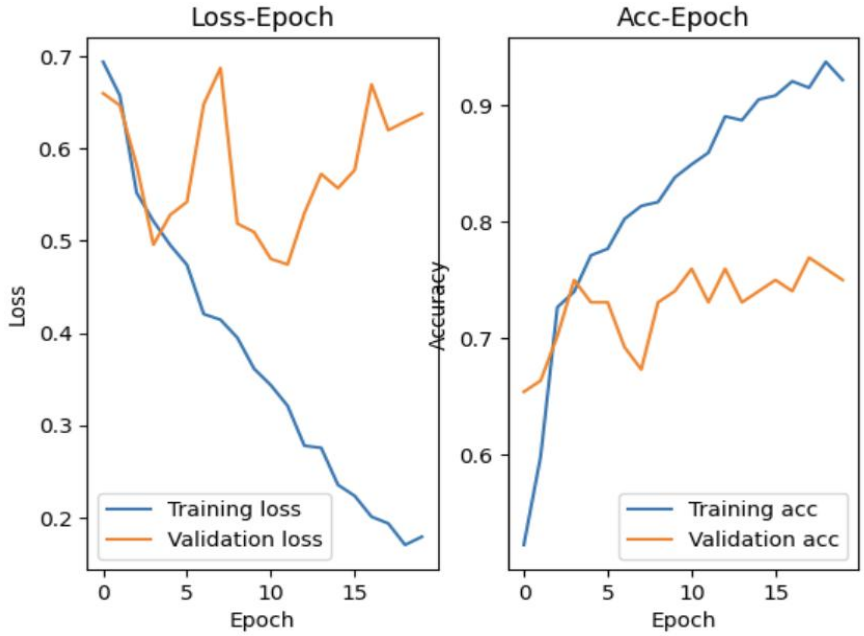
3. 4-bit quantization, Lora configuration: rank=2, alpha=4, dropout rate=0.2, lora is conducted on all linear layers, and the number of trainable parameters is 75266, which is 0.07% of all model parameters.

4. Hyper-parameters:
Optimizer: AdamW
Learning rate: 2e-4
Loss function: cross entropy loss
The number of epoch: 20
Batch size: 8

5. Results:
Baseline model: we predict the [cls] token that has been added to the model.

```
Validation loss: 0.7792, Validation accuracy: 0.3173
```

Fine-tuned model: validation accuracy is close to 75%, which is much higher than the baseline result. However, as the size of our datasets is quite small, it is easy to over-fit the model during the training process.



Qwen1.5-1.8b-chat:
In this experiment, we fine-tune the Qwen1.5-1.8b-chat with 1.8 billion parameters, which are much more than bert. Aftr that we compare the performance with these two models and figure out that fine-tuning on more parameters does not indicate better performance.

1. Dataset pre-processing
We pre-process samples to make them follow the same template. The template is a dictionary which consists of three key-value pairs. Keys include “instruction”, “input” and “output”. In our task, we set the instruction as **“You are traffic police, please evaluate whether the traffic scenario described between the </start> and <end/> is safe. Your answer could only be chosen from the word “dangerous” and “safe”** ”. The value of “input” is the caption concatenated with special tokens </start> and <end/> . The output is the word label.
Here is an example:

```
{
    "instrution":"回答以下用户问题，仅输出答案。",
    "input":"1+1等于几?",
    "output":"2"
}
```

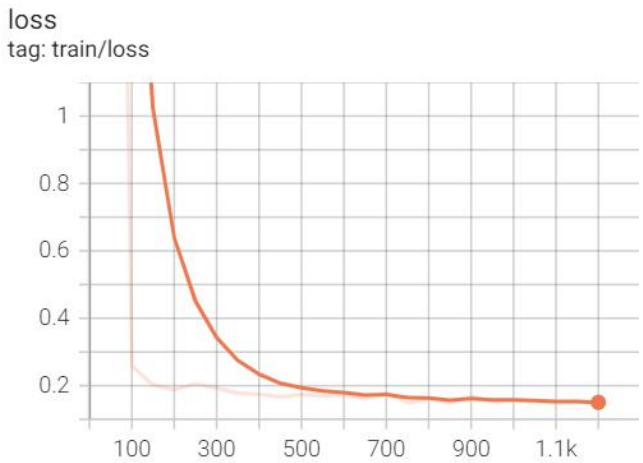
2. Build the prompt template
Instruction and caption are concatenated as the input prompt using the template: **"<s>Human: " + instruction + caption + "\n" + "</s><s>Assistant:"**. The **output** is expected to be the model response.

3. Tokenizer is ultilized to encode prompts and response into tokens with respective indexes in the word library (prompt_ids, target_ids). The maximum sequence length is set to 256, which means parts of the sequence after the 256th token will be truncated.
4. Since Qwen series model is decoder-only, prompt_ids and target_ids should be combined. Therefore final input_ids could be written as:
input_ids = prompt_ids + target_ids + [config.eos_token_id]

5. Split into training set and validation set (9:1). Finally, training set has 890 samples, and validation set has 99 samples

6. Lora config: rank=2; alpha=4, dropout rate=0.2, lora is just conducted on 'o_proj', 'gate_proj', 'up_proj' and 'down_proj' modules. The number of trainable parameters is 1284096, 0.1% of total model parameters.

7. Hyper-parameters:
Optimizer: AdamW
Learning rate: 2e-5
Loss function: cross entropy loss
The number of epoch: 10
Batch size: 8



Even though loss is close to 0, the prediction accuracy only achieves 65%, even worse than the performance of fine-tuned bert.