Mountain-NER. Report

I will break the report down into 3 sections.

I. The data.

The dataset used to train the model was Few-Nerd. It was processed in the following way:

1) All the tags except for the mountain tag (24) were discarded and set to 0.
2) Of the samples that do not contain mountains only a small fraction was used.

II. The model.

The final was a fine-tuned version of 'bert-base-cased'. It has 108M parameters and it was trained for 4 epochs with the following configuration: Adam optimizer with weight decay set to 0.01, huber loss with increased weight of the mountain class, learning rate remained constantly 2e-5. The model achieved almost perfect F_1-score on training data, and the F_1-score on the test set was around 0.87.

III. Possible improvements.

Firstly, I would like to outline what I tried that did not work:

1) Increase the fraction of samples without mountains to keep.
2) Use a different loss function, for example cross-entropy.
3) Use different model architecture.
4) Increase weight decay to regularize the model.

But I still have the following ideas:

1) Increment the dataset. A small improvement might be achieved by training on the whole Few-Nerd, but it would take very long and unlikely lead to a large improvement. So other data sources should be considered. Particularly, it could be useful to use LLMs, like ChatGPT. One could use the LLM to generate synthetic data, and then label it, or one could find real data in the web, and then label it with the LLM. In either case, the labeling process could be improved by firstly fine-tuning the LLM on a small subset of Few-Nerd, to show it how it is done. Of course, it would be really computationally expensive.
2) Try different model architectures. I tried only a few, but there are a lot more. For example, a larger version of BERT.
3) One could also try different regularization techniques. This looks a promising direction because the model rather easily fit to training data. For example, the dropout rate of the model could be tuned.