

## Task 1. Natural Language Processing. Named entity recognition

This task was solved using Hugging Face's models and datasets. More specifically it uses BERT models (dslim/bert-base-NER and dslim/bert-large-NER) for fine-tuning on on the NER dataset with mountain names (Gepe55o/mountain-ner-dataset).

After 3 epochs of training, I got these results:

	train	val	test
eval_loss	0.017100	0.052447	0.054503
eval_model_preparation_time	0.002000	0.002000	0.002000
eval_precision	0.887781	0.776042	0.769197
eval_recall	0.911282	0.831512	0.830727
eval_f1	0.899378	0.802820	0.798779
eval_accuracy	0.994210	0.983945	0.982885
eval_runtime	207.303800	130.359800	127.922300
eval_samples_per_second	144.715000	76.711000	78.172000
eval_steps_per_second	18.089000	9.589000	9.772000

For the base model, and these results:

	train	val	test
eval_loss	0.010216	0.050208	0.052177
eval_model_preparation_time	0.004000	0.004000	0.004000
eval_precision	0.938637	0.817246	0.813042
eval_recall	0.932396	0.846650	0.844976
eval_f1	0.935506	0.831688	0.828702
eval_accuracy	0.996643	0.985701	0.985128
eval_runtime	470.077700	339.036900	346.532100
eval_samples_per_second	63.819000	29.495000	28.857000
eval_steps_per_second	7.977000	3.687000	3.607000

For the large model.

Results are better on the large model, but it also works about 2 times slower.

***What could be done to improve the results:***

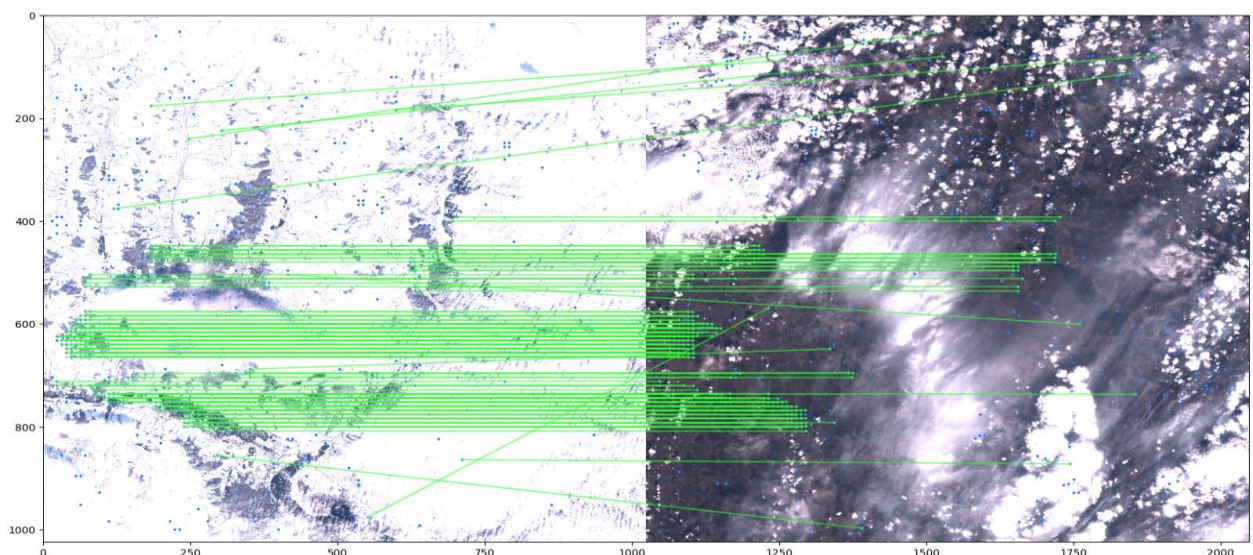
- Trying other models\libraries.

- Trying different hyperparameters (at least the number of epochs and learning rate).
- As far as I know, Hugging Face's Trainer trains all the weights of the model, but considering that we use different classification head, it should be better to firstly train only it with higher learning rate and then unfreeze all the weights and train further with lower learning rate.
- Trying other datasets or adding data to the used one. Perhaps adding\reducing the number of mountain tokens in the dataset could lead to an improvement. Also it is worth checking whether the dataset contains all possible mountain names.

## **Task 2. Computer vision. Sentinel-2 image matching**

This task was solved using SIFT from opencv, DISK+SuperGlue and LoFTR from kornia and images from the following dataset: <https://www.kaggle.com/datasets/isaienkov/deforestation-in-ukraine>. I could rate models\algorithms from the worst to the best like this: SIFT -> DISK+SuperGlue -> LoFTR.

LoFTR uses more computer resources than the other two, but still quite fast and gives good results even with difficult images. For example:



Despite finding some incorrect points, it still found good matches. The other two algorithms found zero matches here.

***What could be done to improve the results:***

- Trying different scaling methods and augmentations (different parameters of clahe, brightness or contrast changes, etc.).
- Changing parameters of algorithms or image sizes.
- Trying another models\algorithms.
- Training or fine-tuning deep learning models (but from what I've seen, it's challenging because there are no user-friendly interfaces for this, and you need to prepare a dataset for it).