University of New Mexico RoadRunners

Jupyter Notebook Path:

gpfs/trn022/proj-shared/wciscc2024/UofNM_RoadRunner/affinity_clip_2/eval_clip_roadrunners.ipynb

(Lowest loss value using same configuration and values as reference)

Machine learning is a quickly growing field that has wide-reaching applications. Many scientific disciplines have adopted machine learning models to improve computational abilities. Computational biology in particular has seen huge dividends from the use of machine learning. It is an incredibly valuable tool for drug discovery and speeding up the time of initial identification to clinical trials. The model we ran this week is based on the BERT masked language model used heavily in natural language processing and computational linguistics disciplines and was repurposed to predict and develop chemical compounds. Contrastive learning models can learn from positive examples and are a self-supervised type of model, which is what worked best for this purpose.

To achieve our solution for the first part of the challenge, we made sure to keep all values the same as the reference we were given, train the model in multiple sessions, and run each cell of the notebook individually to ensure we weren't running into any errors. We were able to achieve our lowest test loss value without modifying parameters after multiple attempts at training the model and running the notebook. None of us had any extensive experience with deep learning frameworks, so most of our team members explored pytorch tutorials and resources to get more familiar with how it operated. We ran into some issues accessing checkpoint directories a few times. Once a run had completed, we were not able to access the checkpoint generated directories inside the run directories, which was limiting for us in some cases. We were able to resolve the issue by cloning a fresh version of affinity clip, which means that someone made some slight change that we were not all aware of altered the checkpoint directory permissions somehow. Despite not having extensive experience with training neural networks, every member of our team was able to train the model and get valid results (cross trained with additional data sets mentioned in instructions), which we all consider a big win. 4/5 of our members have never participated in the winter classic, and many of us have no HPC experience, so being able to get a valid run of a deep learning model on ORNL systems was a huge win for all of us. We also thought that our jobs were failing for a while because they were killed after training for an hour. We worked to debug this issue, but started to get the feeling it may have been a wall time limit issue in the system, so we reached out. The wall time limit was set at an hour, but we were able to pick up training for the model where we left off just by restarting a job with the exact same parameters applied.

Based on our research, we constructed a few approaches for optimizing the model. Our first instinct was to work on parameter tuning. We started by adjusting the batch size, learning rate and number of workers, all to varying degrees of success. We were able to get wandb working on 3 of our runs and use those results to fine tune our parameters to greater effect. None of us have used a tool similar to wandb, and we all found it very useful to be able to track the progress of training the model in real time. The batch size is a huge factor in the loss values we were getting, but we had to be careful with how small we could make the batch size. Training with a tiny batch size can seem appealing because of how low your loss values will be, but it also is not beneficial for the actual learning progression for the model. There is a difficult balance to be struck between having a batch size that is too small and limits how much the model can actually learn and having a huge batch size that trains the model fully and effectively, but then wastes resources continuing to train a model that is already close to its peak performance and proficiency. Additional training datasets can be helpful for improving the performance of machine learning models in many cases, but we continue to try to balance how much training we should be attempting before it stops being beneficial. We attempted training with both the PDBbind and Binding affinity 100k datasets and tried to use both testing datasets provided for us to maximize the efficiency of the model. Additional training datasets can usually improve the generalization capabilities of machine learning models if the data sets are large enough and high quality. There is a point where improvement to the model can no longer be made with more training datasets, but since we only had 2 available to us, we did not think we were anywhere near the threshold of having to consider when too much training no longer improved the models results.

We also investigated using pretrained embeddings and distinct types of transfer learning that could be applied to this specific model. Evolutionary scale modeling was the main pretrained embedding that we focused on implementing. The GitHub repository (cited below) we explored had code and pre-trained weights for Transformer protein large language models, which was interesting to learn about. We were not able to get runs working with the ESM2 modeling code, but many team members are interested in continuing to attempt to implement this embedding for training after the competition this weekend.

Another idea we had for optimization was looking at how the model handles specific proteins and optimizing for each protein individually. We were not sure if this would be possible for this model specifically, but we were able to find a research paper that considered how protein specific optimizations could potentially improve the model more than increasing throughput and scaling up the model. We were not sure how we could try to implement this specifically, so most of our work for this idea was theoretical, but it could be interesting to experiment with if we had more time. The article we read and with that, formulated optimizations based on presented a working protein language model that focused on protein specific optimizations that succeeded in "learning protein evolutionary conservation-mutation trends and introducing functional diversity while retaining key structural-functional characteristics" with performance that surpassed many state-of-the-art models. The article included specific structure and function benchmarks, as well as a protein variant generation analysis on HighN and OneN data scales that the model did the best with. Although these benchmarks and data sets may not align exactly with the ones we were running with, it would be interesting to see how source code adjustments mirroring the protein-specific optimization model could impact performance.

Finally, we investigated using the equivalent of sentence transformers to finetune optimize performance. Because the model is based on a natural language processing framework, we wondered if there were any specific natural language processing tools that could optimize performance for the model when applied to protein mutation. We were able to find a research paper that focused on how natural language processing techniques, specifically sentence transformers, could optimize performance for machine learning models and improve the quality of the results. The paper used an NLP approach called sentence transformer finetuning, which trains a BiEncoder on a pair of sequences (BiEncoder was pretrained on ESM2 models). This approach was more of a limit for protein language models than it is for natural language processing models, since there are incredibly large datasets of example pairs for traditional language models and a significantly smaller dataset of example pairs for protein model training. The article using the sentence transformer approach saw that the model was able to generalize better (sometimes with incredibly significant margins) than models with other approaches that depend on frozen embeddings and perform better when evaluated for accuracy after the training stage. Trying to use this approach may call for a restructuring of much of the source code of the model, but it could be interesting to try to implement this type of model and do a direct comparison of the results using the data sets we were given.

Overall, I think many of us struggled with understanding what a deep learning model is doing at a low level at first, but by training the model, running the notebook and working through all the resources and papers provided to us, we all a much better, well rounded and in depth understanding of not only machine learning models, but their applications to the field of computational biology.

Sources

Blanchard, A. E., Gounley, J., Bhowmik, D., Chandra Shekar, M., Lyngaas, I., Gao, S., Yin, J., Tsaris, A., Wang, F., & Glaser, J. (2022). Language models for the prediction of SARS-CoV-2 inhibitors. The International Journal of High Performance Computing Applications, 36(5-6), 587–602. https://doi.org/10.1177/10943420221121804

Nambiar, A., Heflin, M., Liu, S., Maslov, S., Hopkins, M., & Ritz, A. (2020). Transforming the Language of Life. Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. https://doi.org/10.1145/3388440.3412467

Ahmed Elnaggar, Hazem Essam, Wafaa Salah Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, Burkhard Rost. Ankh ☥: Optimized Protein Language Model Unlocks General-Purpose Modelling, from bioRxiv 2023.01.16.524265; doi: https://doi.org/10.1101/2023.01.16.524265

Redl, I., Airoldi, F., Bottaro, S., Chung, A., Dutton, O., Fisicaro, C., Foerch, P., Henderson, L., Hoffmann, F., Invernizzi, M., Owens, B., Ruschetta, S., & Tamiola, K. (n.d.). Optimizing protein language models with Sentence Transformers. Retrieved February 24, 2024, from https://www.mlsb.io/papers_2023/Optimizing_protein_language_models_with_Sentence_Transformers.pdf

Github Repositories

https://github.com/facebookresearch/esm?tab=readme-ov-file#main-models

https://github.com/PeptoneLtd/contrastive-finetuning-plms