NNTI Project WS 2024/2025

Jatin Kumar
jaku00011@stud.uni-saarland.de

Alexandru Andrita
alan00004@stud.uni-saarland.de

Sambit Basu
saba00015@stud.uni-saarland.de

Abstract

This project focuses on fine-tuning and evaluating a chemical language model for predicting molecular properties using transformers architecture. The first task involves fine-tuning a pre-trained transformer-based model on a regression task with the Lipophilicity dataset, emphasizing dataset tokenization, model loading. The second task extends this by optimizing the fine tuned model's performance through data augmentation. The final task integrates these approaches, implementing advanced techniques to improve model performance. Through these steps, our work displays the effectiveness of transfer learning in chemical informatics and molecular property prediction.

1 Introduction

The purpose of this project is to understand the distribution of the Lipophilicity dataset, then construct a model with the scope of capturing and learning the distribution. In the second and third parts of the project, we will fine tune it using different techniques and analyze the performance.

1.1 Dataset exploration

The data set used for fine-tuning is the Lipophilicity dataset from the MoleculeNet benchmark. The dataset consists of a set of SMILES, each with a lipophilicity assigned. The value spans from negatives to positive values (fact that could be seen in the distribution of lipophilicity too), indicating that some molecules are more oil-soluble while others are water-soluble. Before continuing with the models, we started by splitting the data into train and test sets. Following this procedure, we constructed the data loaders for both the train and test datasets.

1.2 Model selection

When it comes to model selection, we used the MolFormer-XL model, specifically the version MoLFormer-XL-both-10pct. This model is particularly intended for feature extraction and fine-tuning for our task. Tokenization is an important part of the pre-trained model. Before applying the model, we need to convert the smiles into the token IDs. During tokenization, we need to apply padding to ensure all sequences have the same fixed length. Thus, we ensure all could be processed in batches of equal length tensors.

1.3 Evaluation and metrics

In order to evaluate the model performance, since it is a regression task, we used the R2 score and the Mean Squared Error. MSE indicates how far the model is from the ground truth. The R2 measures the proportion of variance in the target explained by the predictions of the model. The higher the score, the better the performance. As our R2 score is around 66%, we could argue that roughly 66% of the variability in lipophilicity is captured by the outputs produced by the model.

2 Influence Function-based Data Selection

2.1 Influence Function

We used the model that we got after fine-tuning from the first task, to find the influence scores. Koh & Liang's paper (2017) was used for calculating influence scores for all the external data points and for calculating we followed Agarwal et al., 2016 and use the LiSSA approximation techniques to find the inverse Hessian Vector product.

2.2 Fine Tuning

We selected top-k(in our case 128) positively scored samples and converted them into the training set without the augmentation part with the original dataset. The fine-tuned model was then evaluated on the original test set.

2.3 Results

The training loss decreased as the epochs increased, showing that the model was performing well on the selected training subset (loss decreasing each epoch). However, the final evaluation on the test split of the original dataset indicated minimal improvement in generalization potentially overfitting with a poor R2 score. This suggests that while the model fit the 128 chosen samples (probably memorizing them), it did not generally improve generalization performance, thus overfitting to the subset or maybe that the subset was not representative of the overall distribution.

2.4 Limitations

While calculating the inverse Hessian Product, the LiSSA approximation is computationally efficient but it is not accurate and may introduce errors while calculation. Additionally, influence-based data selection methods depend on the way the external dataset is representative of the original dataset. And also, selecting the top K data points among the dataset can be non-trivial and would affect model performance severely.

3 Data Selection methods

In this task we explore alternative methods for data selection and investigate several fine-tuning techniques to adapt a pre-trained model for a specific task.

3.1 Curriculum learning

The first data selection technique we try is the curriculum learning technique. This technique is inspired by the way humans learn: from easy to hard. When learning a new topic, the easier concepts are taught first, and slowly harder concepts are introduced. The lipophilicity dataset has SMILES strings and corresponding labels. SMILES (Simplified Molecular Input Line Entry System) strings are text-based representations of molecules. For this task we are taking the length of the SMILES string as the difficulty metric, that is, the bigger the molecule, the harder the problem. Curriculum learning was introduced by Bengio et al., (2009).

What we do is that for every epoch, we train a bit of the data, and then in the next epoch, introduce a bit more data. Here the spread of the data is important. We observe the following about the data: the shortest SMILES string in the dataset is of 11 atoms in length, and the longest one is 267 atoms. We split the difference in 20 bins and find the following:

Bin Range	Frequency	Cum. Frequency
(10.744, 23.8]	174	174
(23.8, 36.6]	897	1071
(36.6, 49.4]	1228	2299
(49.4, 62.2]	1319	3618
(62.2, 75.0]	452	4070
(75.0, 87.8]	102	4172
(87.8, 100.6]	16	4188
(100.6, 113.4]	3	4191
(113.4, 126.2]	2	4193
(126.2, 139.0]	1	4194
(139.0, 151.8]	2	4196
(151.8, 164.6]	1	4197
(177.4, 190.2]	1	4198
(241.4, 254.2]	1	4199
(254.2, 267.0]	1	4200

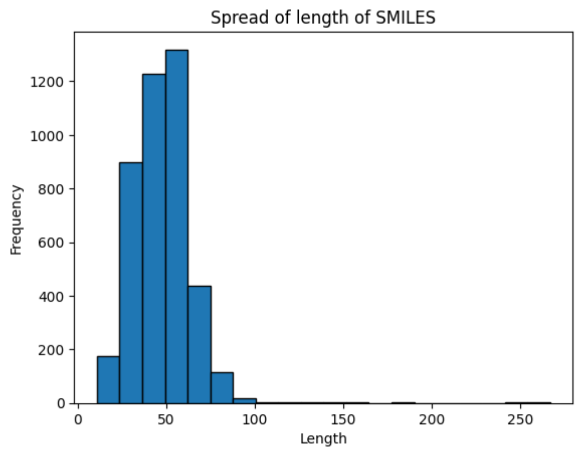

Figure 1: SMILEs spread

We feed the easier data points first into the model, and then along with the increase in epochs, we feed harder strings. We define a threshold, strings of length below which will be fed in that specific epoch. We make use of the following formula to calculate the threshold:

$$\text{Threshold} = d_{\text{min}} + (d_{\text{max}} - d_{\text{min}}) \times \frac{e_{\text{current}}}{e_{\text{total}}} \quad (1)$$

In the equation from above, we use the following notations:

- d_{min} is the minimum difficulty; it represents the initial threshold at the start of training;
- d_{max} is the maximum difficulty; it represents the final threshold at the end of training;
- e_{current} represents the epoch number in the current training iteration

- e_{total} represents the total number of epochs planned

We have performed data selection on the pre-trained model ibm/MoLFormer-XL-both-10pct, onto which we have added a regression head for this task. We curate the data points and then train this model. We created a custom dataloader that would be called before every epoch to feed the difficulty sorted data points. During a base run, with the Adam optimiser with learning rate 0.0001, and with 5 epochs, the final training loss became 0.0939. The test MSE became 0.3752 and the R^2 upto 0.746.

3.2 Active learning via uncertainty sampling

In Active learning, we are utilising uncertainty sampling, where the model itself identifies which data points it is most uncertain about, and thus these data points would be the most beneficial to learn from. We use dropout as the main uncertainty feature. To select the indices, we switch the model into the training mode, which activates the dropout layers in the model architecture, this introduces randomness. This technique is known as Monte-Carlo Dropout, first introduced by Gal and Ghahramani (2016) and extended by Gal et al. (2017). For each batch in the external DataLoader, we perform several forward passes with dropout enabled. We then calculate the standard deviation of these predictions. The higher the standard deviation, the more the uncertainty. The number of uncertain samples is a hyperparametre. During a base run, with the Adam optimiser with learning rate 0.0001, and with 5 epochs, the final training loss became 0.3992. The test MSE became 1.3472 and the R^2 upto 0.0881.

4 Fine tuning methods

4.1 BitFit

The simple basic idea behind **Bi**as-**t**erms **Fi**ne-**t**uning is to just update the bias terms while freezing the rest of the weights. It was first introduced by Ben-Zaken et al., (2022). The main benefit of this fine tuning methodology is that it reduces the number of trainable parametres to less than 0.1% of the full fine-tuning parametres. During a base run, with the Adam optimiser with learning rate 0.0001, and with 5 epochs, the final training loss became 1.0687. The test MSE became 1.2802 and the R^2 upto 0.1334.

4.2 LoRA

Low **R**ank **A**daptation reduces the number of trainable parametres by utilising Low Rank decomposition of matrices. A matrix M (size $d \times k$) can be represented by $M = B \cdot A$, B $(d \times r)$ and A $(r \times k)$, where $r << d$ and $r << k$. When the weight matrices are represented by these low rank matrices, then only these low rank matrices are updated, thus greatly reducing the number of trainable parametres. Hu et al., (2021) introduced the idea. During a base run, with the Adam optimiser with learning rate 0.0001, and with 5 epochs, the final training loss became 1.0499. The test MSE became 1.239 and the R^2 upto 0.1613.

4.3 $(IA)^3$

$(IA)^3$, or Infused Adapter by Inhibiting and Amplifying Inner Activations, introduces lightweight adapter modules into the attention mechanism of the transformers. $(IA)^3$ relies on learning a simple, per-dimension scaling vector. This reduces the number of trainable paremetres to the number of hidden dimensions in the architecture. It was first introduced by Liu et al., (2022). During a base run, with the Adam optimiser with learning rate 0.0001, and with 5 epochs, the final training loss became 0.9788. The test MSE became 1.0415 and the R^2 upto 0.295.

References

Naman Agarwal, Brian Bullins, and Elad Hazan. 2017. Second-order stochastic optimization for machine learning in linear time.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Pang Wei Koh and Percy Liang. 2020. Understanding black-box predictions via influence functions.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.