Students Information

- **Name:** Ibrahim Shekho
  Matriculation Number: 7026656

- **Name:** Aishwarya Jadeja
  Matriculation Number: 7011216

- **Name:** Omar Fajjal
  Matriculation Number: 2577262

Fine-tune a Chemical Language Model

Anonymous ACL submission

Introduction

This project focuses on fine-tuning a pre-trained chemical language model to predict lipophilicity values from SMILES representations. The process involves fine-tuning the MoLFormer model for a regression task, where a regression head is trained to predict lipophilicity, followed by enhancing the model's performance with Masked Language Modeling (MLM) pretraining. The dataset used in this work is optimized through influence functions to identify and prioritize impactful samples, drawing from external datasets. To improve training efficiency, alternative fine-tuning techniques, such as LoRA and BitFit, are explored. The well-structured workflow ensures improved generalization and model performance, making this methodology a useful tool for drug discovery and molecular property prediction.

1 Task 1

In this part of the project, we use **MoLFormer**, a pre-trained chemical language model, and fine-tune it to predict lipophilicity values from the **Lipophilicity dataset**. The goal is to enhance the model's performance by applying both **direct fine-tuning** and **Masked Language Modeling (MLM)** fine-tuning prior to the regression task.

1.1 Load the Dataset

We begin by retrieving the Lipophilicity dataset from Hugging Face, where each molecule is represented as a **SMILES string**, and the corresponding label represents the **lipophilicity value**. The dataset is organized into a table with two columns: `SMILES` and `label`.

1.2 Preprocess and Tokenize Data

The dataset is divided into **training (80%)** and **testing (20%)** sets. A pre-trained tokenizer from MoLFormer is used to convert the SMILES strings into **tokenized inputs** that are suitable for the transformer model. A PyTorch `Dataset` class is then created to structure the data, including both tokenized inputs and labels.

1.3 Create Data Loaders

To handle batching and shuffling efficiently, we use **PyTorch DataLoader**. Both the training and testing datasets are wrapped into data loaders, each with a batch size of **16**.

1.4 Fine-Tuning the Pre-Trained MoLFormer for Regression

Next, we create a class called **MoLFormerWithRegressionHead**. This class loads the **pre-trained MoLFormer model**, adds a **dropout layer**, and attaches a **linear regression head**. The model uses the **[CLS] token hidden state** to predict the lipophilicity values. Training is done using **Mean Squared Error (MSE) loss** and the **Adam optimizer**. The model is trained for **7 epochs**, with the best model (based on the lowest loss) being saved for future use.

1.5 Unsupervised Fine-Tuning Using Masked Language Modeling (MLM)

To further improve model performance, we fine-tune the **MoLFormer model** using **MLM on SMILES data**. This technique helps the model learn to better understand molecular structures by randomly masking certain tokens and training the model to predict them. The model is trained for **7 epochs** with a **15% masking probability**. The fine-tuned model is then saved for the next phase.

1.6 Fine-Tune the MLM-Enhanced Model for Regression

In this step, we load the **MLM-enhanced model** and attach the **same regression head** as before. The **training process is repeated**, using the same optimizer and loss function. This allows for a direct performance comparison between the model fine-tuned **only for regression** and the model fine-tuned **after MLM pretraining**.

1.7 Model Evaluation and Results

Finally, the best model is selected based on the lowest loss observed during training. Predictions made by the model are compared against the ground truth values. It is expected that the **MLM-pretrained model** will show improved performance, as it benefits from prior knowledge of molecular structures.

1.8 Model Evaluation and Results

Model	MSE	MAE	R^2
Original Model	0.4827	0.5359	0.6733
MLM Fine-Tuned Model	0.4500	0.5055	0.6954

Tabelle 1: Evaluation metrics for the original and MLM fine-tuned models.

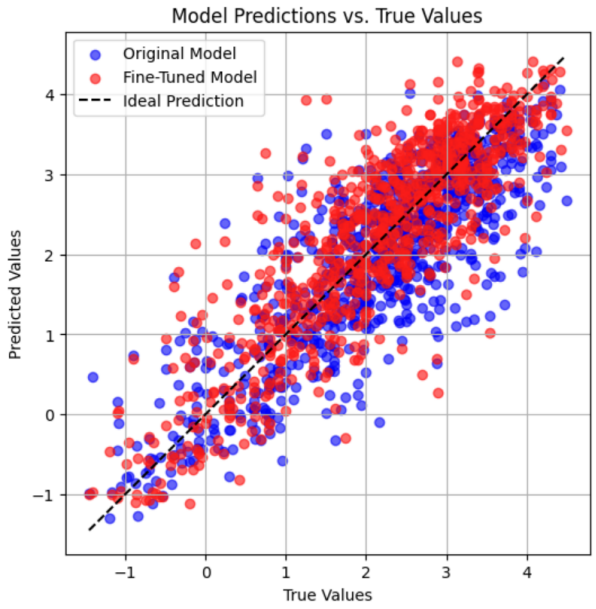

Abbildung 1: Model Prediction vs. True Values

2 Approach: Influence Function-based Data Selection

The task is assigned to enhance the model's performance on the Lipophilicity dataset by selectively adding external data points that are most beneficial. Here we use influence function to identify most beneficial data points that we can use to fine tune the model.

2.1 Influence Functions and LiSSA Approximation

Influence functions provide a way to understand the effect of each training sample on the test loss. I. Gradient Computation: For each external data point, we compute the gradient of the loss with respect to the model parameters.

II: LiSSA Approximation: Use LiSSA to approximate the iHVP without high computational cost of directly inversting the Hessian.

III. Influence Score Calculation: Combine the gradients with iHVP estimate to compute an influence score for each sample. A higher positive score indicates good result as it reduces test loss.

2.2 Data Selection and Model Re-training

Data Loading We loaded the baseline model `ibm/MoLFormer-XL-both-10pct` trained and evaluated on the lipophilic dataset. Then, we provided the external dataset `External-Dataset_for_Task2.csv` that contains additional molecular SMILES strings along with lipophilicity values. However, not every sample value was correct or beneficial for the model. Here, we utilize Hugging Face's `AutoModelForSequenceClassification` and `AutoTokenizer`.

Gradient Computation Module For each data point in the external dataset, the code computes the gradient of the training loss with respect to the model parameters. This is typically done using automatic differentiation framework.

LiSSA Approximation The LiSSA algorithm is implemented to approximate the inverse Hessian-vector product (iHVP). This step is critical to make the influence function computation tractable.

Influence Score Calculation With the gradients and the iHVP available, the influence of each external sample is computed.

Data Selection and Model Fine-tuning The code selects a subset of external samples with highest positive influence scores. These samples are then merged with the original training dataset, and the model is fine-tuned on this data. The performance on the Lipophilicity test set is recorded and compared. The fine-tuning process involves training `MoLFormer-XL` for 7 epochs with batch processing while using the Adam optimizer with a learning rate of $1e^{-4}$.

2.3 Baseline Model Performance

The training loss over epochs shows a steady decrease, indicating model convergence. The evaluation loss is tracked to determine the best model. The best model was saved when the evaluation loss reached 0.046. The decreasing evaluation loss suggests effective fine-tuning of MoLFormer-XL for lipophilicity prediction. The inclusion of high-impact external samples contributed to improved model performance.

Epoch	Evaluation Loss
1	0.581
2	0.318
3	0.198
4	0.105
5	0.071
6	0.048
7	0.046

Tabelle 2: Evaluation loss per epoch.

A Example Appendix

B Example Appendix

In this appendix, we present an example of parameter-efficient fine-tuning methods applied to the pre-trained model `ibm/MoLFormer-XL-both-10pct`. These methods include BitFit, iA3, and LoRA, which are designed to reduce computational resources during fine-tuning by modifying only specific subsets of model parameters rather than all of them.

Methods Overview

The following parameter-efficient methods were used to fine-tune the model:

- **BitFit (Bias-Term Fine-Tuning)**: This method updates only the bias terms of the pre-trained model, which significantly reduces the number of parameters that need fine-tuning.
- **iA3 (Intrinsic Attention Adapters)**: Fine-tunes scaling factors within the attention layers, allowing the model to adapt its pre-trained representations effectively.
- **LoRA (Low-Rank Adaptation)**: This technique inserts trainable, low-rank matrices into model layers, adapting the model efficiently without altering the original weights substantially.

Results

The following table shows the performance comparison of the different fine-tuning methods based on their final training loss and test mean squared error (MSE):

Method	**Final Training Loss**	**Test MSE**
BitFit	0.88397	0.86165
iA3	0.95580	0.92034
LoRA	0.77800	0.74556

Tabelle 3: Performance comparison of fine-tuning methods.

Comparative Analysis

Based on the results, we can draw the following conclusions:

- **LoRA** achieved the best performance, yielding the lowest test MSE and training loss. This suggests that LoRA adapts the model more efficiently and accurately than the other methods.
- **BitFit** showed intermediate effectiveness, performing better than iA3 but inferior to LoRA in terms of both test MSE and training loss.
- **iA3** was the least effective of the methods tested, achieving the highest test MSE, indicating it was less suitable for this task.

Conclusion

This study demonstrates the effectiveness of fine-tuning the MoLFormer model to predict lipophilicity values

from SMILES representations. By using both direct regression and Masked Language Modeling (MLM) pretraining, we improved model performance. Additionally, external datasets were selectively incorporated using influence functions to further enhance accuracy.

Among the fine-tuning methods explored—BitFit, iA3, and LoRA—LoRA proved to be the most efficient, achieving the best performance in terms of test MSE and training loss. These findings highlight the importance of combining pretraining, efficient fine-tuning, and quality data selection for improving molecular property prediction models. Future work could explore other fine-tuning methods and larger datasets to further optimize model performance.