Fine-tune of Chemical Language Model

Faria Alam 7011180
Muhammad Sabeeh Rehman 7047213
Hafiza Hajrah Rehman 7063002

Abstract

In this experimental analysis, we fine-tune a chemical LM model Molformer. To enhance it's performance, we explore three experimental strategies. Our results demonstrate that the experiments have increased the model's predictive performance significantly.

Introduction

In our project we utilize a transformer-based neural network model from (Ross et al., 2022) trained on two public chemical datasets PubChem and ZINC in a self-supervised fashion. We use this pre-trained `Molformer model` for our downstream prediction task after fine-tuning it with the `Lipophilicity` dataset. Here, three different type of experiments were done: masked language modeling with the `Lipophilicity` dataset in unsupervised fashion, again fine-tuning the model with the combined dataset of `Lipophilicity` and external influential data samples measured by influence function, applying different data selection techniques and parameter-efficient fine-tuning strategies to improve model's performance.

Experimental Setup

Data For the first task the `Lipophilicity` dataset from Hugging face [1] has 4200 samples of sequences of chemical molecules called the `SMILES representation` or `SMILES strings` along with its lipophilicity values which is the target label. The 300 external samples that we use to choose impactful datapoints have a similar structure. Target values for both datasets were normalized to 0-1 using `MinMaxScaler()`

Model The pre-trained `Molformer model` is used as backbone model and a 2-layer neural network is used as regression head. Output of the `Molformer` model is a compressed mean pool representation of the `SMILES strings` of dimension 768 which is then fed into the regression head. An overview of the pipeline including our downstream regression task is illustrated in this image 1. In the regression head, we used `Relu` for non-linearity and dropout as a regularizer with 0.1 probability. This is used as the base model for performance comparisons.

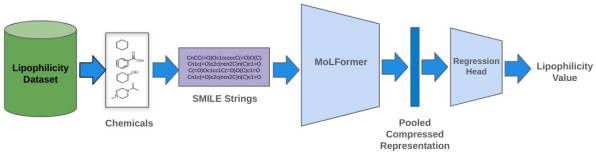

Figure 1: Molformer pipeline with downstream regression head

Metrics For this regression task, our primary metrics are mean squared error(MSE), root mean squared error(RMSE). In addition with these, we also use Spearman, Pearson correlation, R-squared to understand correlation and variability of target label.

Fine-tuning Chemical LM on Lipophilicity

In this first task, we fine-tune our pre-trained `Molformer model` with regression head using `Lipophilicity` dataset to predict the `Lipophilicity` value. Then the same `Molformer model` is fine-tuned in unsupervised manner using `masked language modeling(mlm)` and only `SMILES strings`. In both experiments, the `SMILES strings` are tokenized using `AutoTokenizer` and converted into a vector representation of dimensions 768. We measured all performance metrics on three seed values to ensure reproducibility. The effect of further `masked language modeling(mlm)` for all seeds can be seen here in Figure 2. The model for seed 100 has the lowest MSE and RMSE and used for

[1] https://huggingface.co/datasets/scikit-fingerprints/MoleculeNet_Lipophilicity

task2.

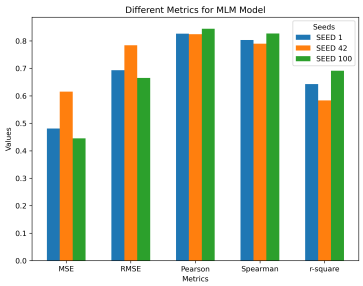

Figure 2: Different performance metrics of Molformer model after masked language modeling(mlm)

Influence Function-based Data Selection

In this experiment, we incorporate both the `MoleculeNet_Lipophilicity` dataset and an external dataset containing 300 samples for fine-tuning. However, unlike the straightforward inclusion approach used for `MoleculeNet_Lipophilicity`, the external samples are carefully selected based on an influence function–driven procedure.

Influence Function We employ influence function to determine the effect of each external sample on test set's loss function. Thus we are only selecting the most influential data points that has positive impact on model's performance. This approach follows the work of (Koh and Liang, 2020). The influence score is formulated as:

$$I(z) = -\nabla_\theta L(z_{test})^T H^{-1} \nabla_\theta L(z_{ext})$$

where $I(z)$ is the influence score of an external sample z_{ext}, H^{-1} is the approximated inverse Hessian matrix, $\nabla_\theta L(z)$ is the gradient of the loss function with respect to the model parameters. This is consisting of three steps-

(1) *Compute gradients of test samples-* To compute gradients we use the model obtained after `masked language modeling(mlm)` with regression head from task 1. Since after `masked language modeling(mlm)` this model has become more domain adapted compared to the pre trained `ibm/MoLFormer-XL-both-10pct` model.

(2) *Estimate the inverse Hessian-Vector Product(iHVP)-* to compute the second-order gradient, inverse Hessian needs to be determined. However, computing the inverse of Hessian is computationally expensive `o(n^3)`. Therefore, instead of inverse Hessian, we approximate the inverse Hessian-Vector product `(iHVP)` by using the LiSSA approximation ((Agarwal et al., 2017)).

(3) *Compute the influence score of external samples-* In order to compute the influence score, the gradients of all external samples are determined. Subsequently, we perform a **dot product** between external sample's gradient and the estimated `(iHVP)`. Hence we obtain the influence score for every external samples.

Selecting the top influential data and its impact After computing influence scores for 300 external samples (both positive and negative), we select the top positive samples by sorting in descending order. These high-impact samples are expected to improve model performance by reducing MSE and RMSE. We combine them with the Lipophilicity training set and fine-tune the post-masked language modeling (MLM) model with a new regression head. During testing, we compare the performance of top positive influential samples against a few samples with negative scores. The results of this analysis are shown in Figure 3.

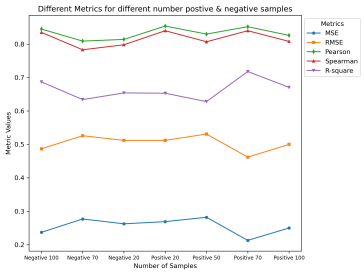

Figure 3: Comparison of highest scored positive samples with negative scored samples

The graph analyzes sample performance across a range of sample counts: [-100, -70, -20, 20, 50, 70, 100], negative sign indicating negative influence score. Some negatively scored samples (top 100 negative) outperform even the top 50 high-scored positive samples. Among high-scored positive samples, a non-monotonic pattern emerges across all performance metrics, indicating that model performance does not consistently improve with more samples until reaching the top 70. At this point, performance improves across all metrics. One possible reason is that LiSSA, which approximates the inverse Hessian-vector product (iHVP) via an iterative algorithm, may produce deviations from the true Hessian inverse if the number of iterations or hyperparameters is not well-tuned, leading to overestimated influence scores. Additionally, external data may contain slightly different SMILES strings compared to those the model learns during mask language modeling (MLM), resulting in domain

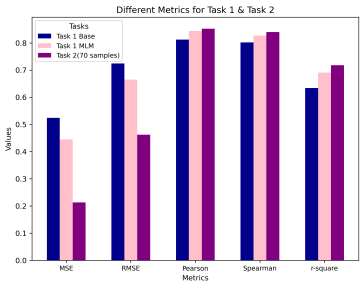

Figure 4: Comparison of base model, task1, task2 on performance metrics

mismatch and degraded performance. Conversely, the SMILES strings within the top 70 most influential samples may be more aligned with the test data, enhancing model performance. The model trained on these top 70 samples is used for performance comparisons between Task 1 and Task 2, as illustrated in Figure 4, with the initial fine-tuned Molformer model—prior to MLM—serving as the baseline.

Figure 4 shows that combination of 70 influential data with training data reveals synergic effect for all performance metrics.

Exploration of Data Selection and Fine-Tuning Methods

Fine-tuning a model requires careful selection of training data to ensure optimal performance. In this experiment, two data selection strategies were employed: **Diversity Sampling** and **Stratified Sampling**. The goal was to analyze how different data selection methods impact model performance and to compare the effectiveness of each strategy.

Data Selection Strategies

Stratified Sampling Stratified sampling ensures that the selected data maintains the proportional distribution of different classes or features in the dataset. This method helps in reducing bias and ensuring balanced representation. The process involves (1) Binning the labels into three categories (Low, Medium, High) using quantile-based discretization (2) Performing stratified sampling to select a representative subset of data while maintaining class distribution, (3) Merging the stratified subset with the original dataset.

Diversity Sampling Diversity sampling ensures that selected samples are highly varied, capturing different characteristics of the dataset. This method aims to improve generalization by exposing the model to diverse patterns. The process involves (1) Generating embeddings for SMILES data using a model and tokenizer, (2) Performing diversity-based selection using clustering techniques (e.g., k-means with 5 clusters), (3) Selecting 100 diverse samples based on the generated embeddings. (4) Merging the selected dataset with the original dataset.

Figure 5: Comparison of tuning strategies

Analysis and Discussion Figure 5 summarizes the results of both data selection methods. The results indicate that, The model trained with **Diversity Sampling** achieved slightly lower error metrics (MSE: 0.4815 vs. 0.4909, RMSE: 0.6939 vs. 0.7006, MAE: 0.5129 vs. 0.5304), suggesting improved generalization. The R^2 score for Diversity Sampling was marginally higher (0.6801 vs. 0.6726), indicating a better fit. Pearson correlation was also higher for Diversity Sampling (0.8476 vs. 0.8371), meaning a stronger linear relationship between predicted and actual values. Spearman correlation was higher for Diversity Sampling (0.8260 vs. 0.8094), indicating a stronger rank-based association between the predictions and actual valuesOverall, **Diversity Sampling** resulted in slightly better performance, likely due to increased variety in the training data, which helped the model generalize better.

Parameter Tuning - Model Structure

For Task 3b, we implement different algorithms to evaluate their performance on the `ibm/MoLFormer-XL-both-10pct` model. We use the `MoleculeNet_Lipophilicity` dataset along with the external dataset provided to us for training the model.

LoRA(Low-Rank Adaptation) freezes pre-trained weights and introduces two low-rank matrices, A and B, where $B \in R^{d \times r}$ and $A \in R^{r \times k}$. Instead of updating W_0, the model trains A and B, modifying the forward pass to $h = W_0 x + BAx$. In our implementation, only the self-attention layer is replaced with LoRA, reducing trainable parameters

to 885,505 (1.95% of the total) (Hu et al., 2021).

IA3 applies learned vectors to scale model activations element-wise while keeping all parameters frozen. In self-attention, it modifies the computation as:

$$softmax\left(\frac{Q(\mathbf{lk}\mathbf{K}^T)}{\sqrt{d_k}}(\mathbf{lv}\mathbf{V})\right)$$

where Lk and lv are learned vectors. Additionally, in the decoder, $fc1$ is scaled by another learned vector l_{ff}. This introduces 295,681 new trainable parameters (0.66% of the total) (Liu et al., 2022).

Observation

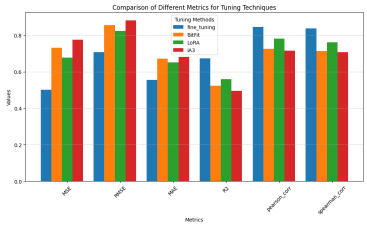

Figure 6: Comparison of tuning strategies

Fine Tuning For our baseline, we first fine-tune our model. In fine-tuning, we train all the available parameters, including the model's weights and biases. For our specific model, we have 44670721 parameters. As seen in Figure 6, fine-tuning achieves the highest R^2 score (0.72), demonstrating its ability to capture the most variance. It also exhibits the lowest RMSE, confirming its strong predictive accuracy. Furthermore, fine-tuning maintains the highest Spearman correlation ensuring that predictions preserve rank relationships effectively. However, it also highlights its drawback: Fine-tuning requires the most computational resources, making it less viable for efficiency-critical applications.

Bitfit(bias-term fine-tuning) is a parameter-efficient tuning method that trains only the model's biases. After implementing bit fit on our current model, we are left with 370177 parameters which is just 0.82% of the total number of parameters (Zaken et al., 2022). Despite its limited parameter updates, BitFit achieves an R^2 score of 0.55 (Figure 6), which, while lower than fine-tuning, still demonstrates competitive performance. However, it exhibits a higher RMSE, suggesting that its predictions differ more from true values. The Spearman correlation also shows a slight drop compared to fine-tuning, indicating reduced consistency in ranking relationships. BitFit, however, significantly reduces training time making it a viable option when computational efficiency is a priority.

LoRA emerges as the best alternative to fine-tuning, striking a balance between accuracy and efficiency. LoRA achieves an R^2 score of 0.65 (Figure 6), closely approaching fine-tuning. Its RMSE remains lower than that of BitFit and iA3, strengthening its predictive strength. Furthermore, we see that LoRA maintains a high Spearman correlation, preserving relative rankings within the data. While its training time is comparable to fine-tuning, the reduction in trainable parameters makes it more computationally efficient. This adaptability makes LoRA a strong contender for real-world applications requiring both performance and efficiency.

IA3 Among all techniques, iA3 exhibits the most significant increase in error rates. As seen in Figure 5, iA3 achieves the lowest R^2 score (0.50), meaning it explains the least variance in the data. Its RMSE is the highest among all techniques, confirming that its predictions are more prone to deviations. It also indicates that iA3 has the lowest Spearman correlation, suggesting that it struggles to maintain rank-based relationships. However, the key advantage of iA3 lies in its efficiency as it has the shortest training time, making it a suitable option for scenarios where computational resources are limited.

Conclusion

In this study, we perform fine-tuning and masked language modeling on chemical LM. We observe that combination of training set with influential samples selected based on influence score significantly improves performance. Though influence score should be determined with caution as it can be overestimated, adversely effecting model's performance. Diversity Sampling showed slight performance improvements over Stratified Sampling. We found that BitFit offers efficiency gains but struggles with generalization, while iA3 prioritizes training speed at the expense of predictive quality. Maintaining strong accuracy while reducing trainable parameters, LoRA might be the most practical alternative to full model fine-tuning. As future work, we should explore more powerful parameter and computation efficient fine-tuning techniques and explore hybrid data selection strategies to improve models overall efficiency and performance.

References

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

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.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. 2022. Large-scale chemical language representations capture molecular structure and properties.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language models. *arXiv preprint arXiv:2106.10199*.