# Task 2

## Overall Goal

The goal of this task is to fine-tune a chemical language model (MoLFormer-XL) for a regression task using the Lipophilicity dataset. Additionally, we analyze and leverage influence scores to identify and incorporate impactful external data samples, enhancing model performance.

The process consists of the following key steps:

- Loading a pre-trained MoLFormer model and adding a regression head.
- Training the model
- Computing influence scores using Hessian inverse-vector products (IHVP).
- Selecting external dataset samples based on influence scores to update the training set.
- Fine-tuning the model with and without the selected external samples.
- Evaluating and comparing model performance before and after influence-based fine-tuning.

## Additional Libraries Used and Justification

- torch: Handles deep learning model training, inference, and gradient computations.
- transformers: Provides the MoLFormer model and tokenizer.
- datasets: Loads the Lipophilicity dataset from Hugging Face.
- scikit-learn: Used for data splitting and evaluation metrics (MSE, RMSE, R2, MAE).
- tqdm: Displays progress bars for dataset loading and training.
- pandas: Handles dataset conversion and manipulation.

## Model and Dataset Handling

### Model Architecture

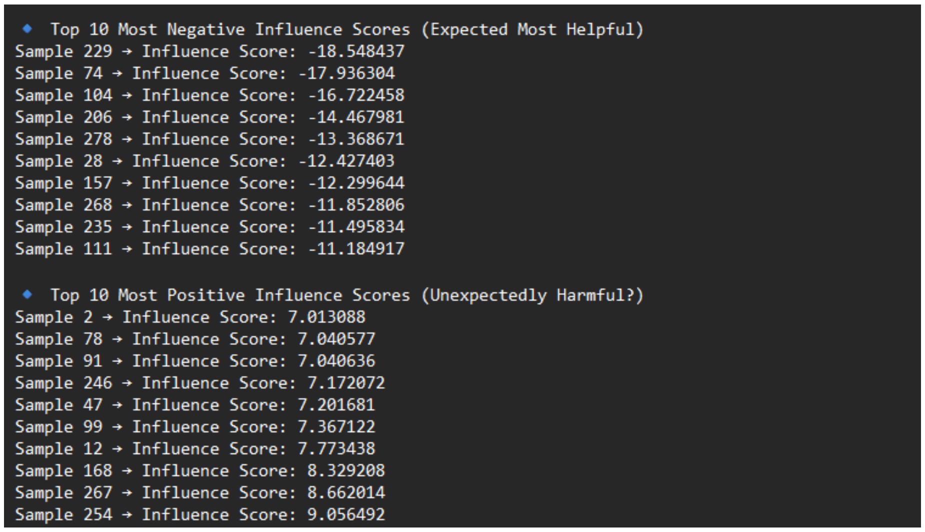
- The model is based on MoLFormer-XL, a chemical language model pre-trained on molecular representations.
- A regression head (fully connected layer) is added on top of the model to predict lipophilicity values.
- CLS token pooling is used to extract a global representation of the SMILES sequence.

- after adding regression head we did a training to update model with regression head.

### Datasets Used

- **Lipophilicity Dataset:** Loaded from `scikit-fingerprints/MoleculeNet_Lipophilicity`, split into 80% train and 20% test.
- **External Dataset:** Loaded from `External-Dataset_for_Task2.csv`, containing additional labeled SMILES sequences.
- **Updated Training Set:** Augmented with external samples selected based on influence scores.

## Influence Score Computation and Dataset Update



```
🔹 Top 10 Most Negative Influence Scores (Expected Most Helpful)
Sample 229 → Influence Score: -18.548437
Sample 74 → Influence Score: -17.936304
Sample 104 → Influence Score: -16.722458
Sample 206 → Influence Score: -14.467981
Sample 278 → Influence Score: -13.368671
Sample 28 → Influence Score: -12.427403
Sample 157 → Influence Score: -12.299644
Sample 268 → Influence Score: -11.852806
Sample 235 → Influence Score: -11.495834
Sample 111 → Influence Score: -11.184917

🔹 Top 10 Most Positive Influence Scores (Unexpectedly Harmful?)
Sample 2 → Influence Score: 7.013088
Sample 78 → Influence Score: 7.040577
Sample 91 → Influence Score: 7.040636
Sample 246 → Influence Score: 7.172072
Sample 47 → Influence Score: 7.201681
Sample 99 → Influence Score: 7.367122
Sample 12 → Influence Score: 7.773438
Sample 168 → Influence Score: 8.329208
Sample 267 → Influence Score: 8.662014
Sample 254 → Influence Score: 9.056492
```

Figure 1: Top 10 Most Negative and Positive Influence Scores

Figure 1 shows the top 10 most negative and most positive influence scores. Negative influence scores represent data points that are expected to be most beneficial for fine-tuning, while positive influence scores may introduce noise or harm performance.

### Hessian Inverse Approximation

- The Hessian inverse is computed using the LiSSA approximation method, which efficiently estimates the inverse Hessian-vector product (IHVP) using iterative updates.

- If we wanted to save this Hessian inverse as a matrix, the calculation would be too expensive to handle, and the GPU RAM would overload and stop working.
- The inverse Hessian vector is saved as `hessian_inverse_fast.pt`.

### Computing Inverse Hessian-Vector Product (IHVP)

- Once we have the Hessian inverse, we compute the IHVP, which captures how test gradients interact with the inverse Hessian.
- This is done by multiplying the test gradient vector by the precomputed Hessian inverse approximation.
- The computed IHVP is then used to determine the influence scores for external samples.
- To handle computation efficiently, element-wise multiplication was used to avoid excessive memory usage.
- Regularization was applied using the norm of IHVP to prevent excessively large values.

### Influence Score Computation

Influence scores measure the effect of individual training samples on model predictions:

$$\text{Influence Score} = -(H^{-1} \cdot \nabla L_{\text{test}}) \cdot \nabla L_{\text{external}} \tag{1}$$

The computed scores are stored in `external_influence_scores.pt`.

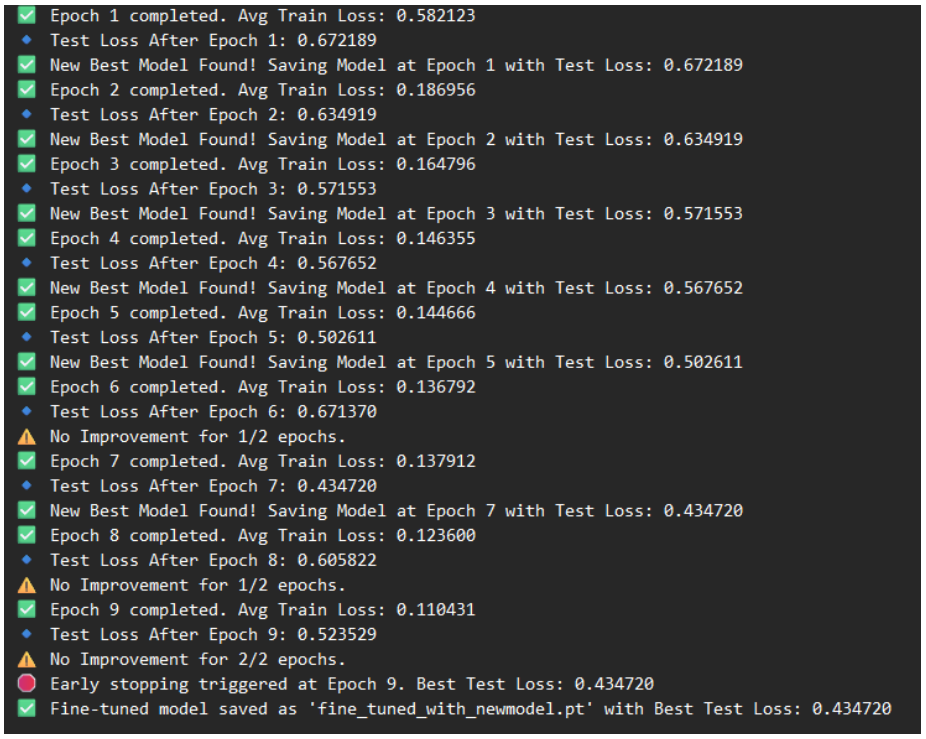
## Fine-Tuning and Evaluation

Table 1 presents the model performance metrics for different fine-tuning strategies. We observe that fine-tuning with 50 and 70 most negative influence samples improves performance, while using 100 most negative influence samples may lead to slight overfitting.

| Model | MSE ↓ | RMSE ↓ | MAE ↓ | $R^2$ ↑ |
|---|---|---|---|---|
| Baseline Model | 0.5457 | 0.7387 | 0.5756 | 0.6307 |
| 50 Most Negative | 0.4984 | 0.7062 | 0.5443 | 0.6624 |
| 70 Most Negative | 0.4934 | 0.7024 | 0.5420 | 0.6660 |
| 100 Most Negative | 0.5222 | 0.7227 | 0.5553 | 0.6465 |

Table 1: Comparison of Model Performance with Influence-Based Fine-Tuning

Figure 2 illustrates the early stopping process, where the best test loss was achieved at epoch 7, preventing overfitting and ensuring optimal model performance.



```
✅ Epoch 1 completed. Avg Train Loss: 0.582123
🔹 Test Loss After Epoch 1: 0.672189
✅ New Best Model Found! Saving Model at Epoch 1 with Test Loss: 0.672189
✅ Epoch 2 completed. Avg Train Loss: 0.186956
🔹 Test Loss After Epoch 2: 0.634919
✅ New Best Model Found! Saving Model at Epoch 2 with Test Loss: 0.634919
✅ Epoch 3 completed. Avg Train Loss: 0.164796
🔹 Test Loss After Epoch 3: 0.571553
✅ New Best Model Found! Saving Model at Epoch 3 with Test Loss: 0.571553
✅ Epoch 4 completed. Avg Train Loss: 0.146355
🔹 Test Loss After Epoch 4: 0.567652
✅ New Best Model Found! Saving Model at Epoch 4 with Test Loss: 0.567652
✅ Epoch 5 completed. Avg Train Loss: 0.144666
🔹 Test Loss After Epoch 5: 0.502611
✅ New Best Model Found! Saving Model at Epoch 5 with Test Loss: 0.502611
✅ Epoch 6 completed. Avg Train Loss: 0.136792
🔹 Test Loss After Epoch 6: 0.671370
⚠ No Improvement for 1/2 epochs.
✅ Epoch 7 completed. Avg Train Loss: 0.137912
🔹 Test Loss After Epoch 7: 0.434720
✅ New Best Model Found! Saving Model at Epoch 7 with Test Loss: 0.434720
✅ Epoch 8 completed. Avg Train Loss: 0.123600
🔹 Test Loss After Epoch 8: 0.605822
⚠ No Improvement for 1/2 epochs.
✅ Epoch 9 completed. Avg Train Loss: 0.110431
🔹 Test Loss After Epoch 9: 0.523529
⚠ No Improvement for 2/2 epochs.
🔴 Early stopping triggered at Epoch 9. Best Test Loss: 0.434720
✅ Fine-tuned model saved as 'fine_tuned_with_newmodel.pt' with Best Test Loss: 0.434720
```

Figure 2: Early Stopping Mechanism During Fine-Tuning

### Fine-Tuning Setup

- at first Models are fine-tuned for 5 epochs with a learning rate of $2e-5$.
- The AdamW optimizer is used with a weight decay of 0.01.
- Training is performed on CUDA if available, otherwise CPU.
- Baseline: Lipophilicity dataset only.
- fine tuned with same model: Lipophilicity dataset + influence-selected external samples training on the same model that is trained with training data
- fine tuned with fresh model: Lipophilicity dataset + influence-selected external samples training on a fresh model that regression head is just applied and we didn't train it yet.

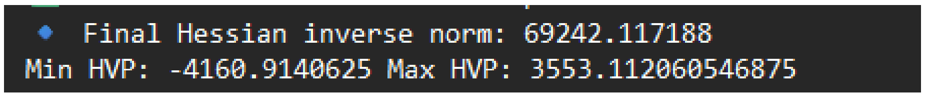
### Evaluation Metrics

- Mean Squared Error (MSE)

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R² Score (Coefficient of Determination)

## Conclusions

We observed that adding external data points with the most negative influence scores significantly improved performance. However, excessive addition (e.g., 100 most negative samples) led to overfitting, as shown in Table 1. The Hessian inverse and IHVP calculations, as visualized in Figures 3 and 4, provide insight into the influence of training samples on test loss.

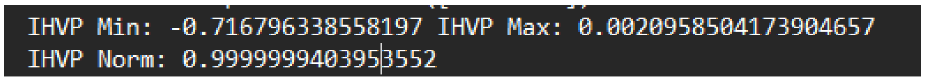


```
 ◆  Final Hessian inverse norm: 69242.117188
Min HVP: -4160.9140625 Max HVP: 3553.112060546875
```

Figure 3: hessian inverse norm and max and min

These findings highlight the importance of careful influence-based data selection and the role of early stopping in optimizing performance.

- Influence-based sample selection can improve test performance if selected carefully
- it is prone to overfit fast if we just add most negative influence scores for example, it is better to add from both ends to keep generalization if we are using same model to fine tune.
- if we are using a fresh model to fine tune, it is better to use most negative ones.
- Computing the Hessian inverse is computationally expensive but manageable with LiSSA approximation.
- The number of negative influence samples must be optimized for best generalization.
- using early stopping prevents model from overfitting , and helps us with fine tuning on both trained model and not trained model comparing the evaluations of a trained model without fine tunin.



```
IHVP Min: -0.716796338558197 IHVP Max: 0.0020958504173904657
IHVP Norm: 0.9999999403953552
```

Figure 4: ihvp min and max and norm

## future works

Next steps include experimenting with different sample selection strategies, exploring various loss functions for Hessian computation, and implementing a dynamic re-weighting approach instead of hard selection.

## References

[1] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, vol. 70, pp. 1885–1894, 2017. [Online]. Available: `https://proceedings.mlr.press/v70/koh17a/koh17a.pdf`

[2] A. Agarwal, L. Bottou, X. Munos, and P. S. Pereyra, "Second-order stochastic optimization for machine learning in linear time," *Journal of Machine Learning Research*, vol. 18, no. 116, pp. 1–40, 2017. [Online]. Available: `https://www.jmlr.org/papers/volume18/16-491/16-491.pdf`