NNTI project presentation

Presented by:

Fahim Ahmed Shakil

Roba ElQadi

17.3.2025

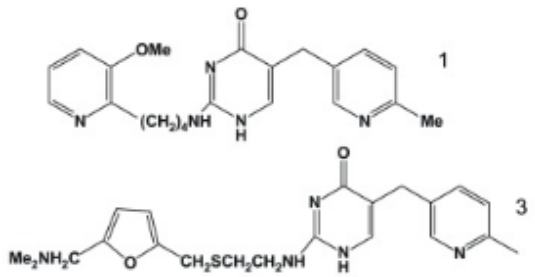
Saarbrücken, Germany

Table of contents

- Fine-tuning Pre-trained LM for a New Task (Task 1)
- Data Selection Using Second-Order Stochastic Optimization (Task 2)
- Comparing Fine-tuning and Data Selection Methods (Task 3)

Task 1

- Overview: aim to fine-tune a pretrained chemical language model (MoLFormer) on regression task to predict the Lipophilic value for chemical structures.

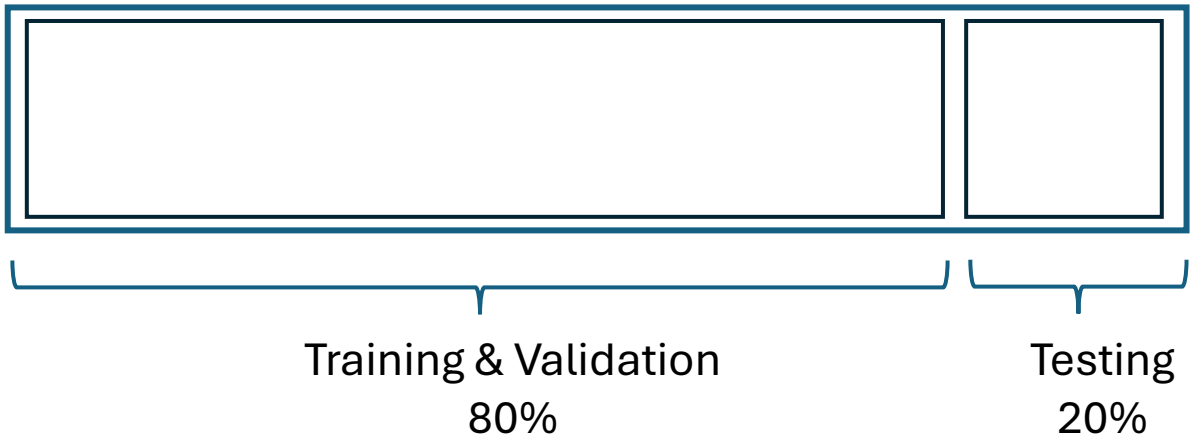


- Dataset: the MoleculeNet Lipophilicity dataset, contains SMILES strings and corresponding lipophilicity values.

Task 1

- Overview: This task aims to fine-tune a pretrained chemical language model MoLFormer) on regression task.

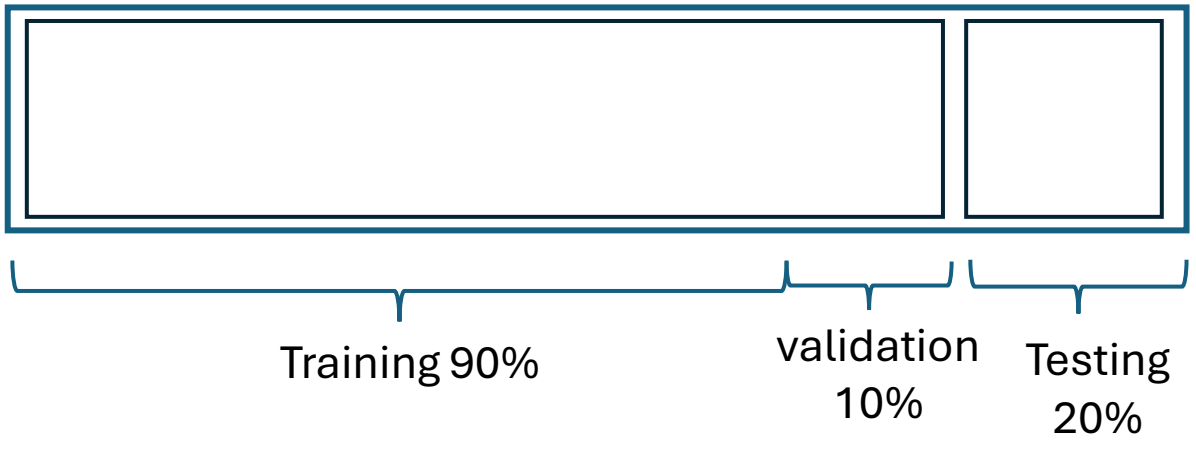
- Dataset: the MoleculeNet Lipophilicity dataset, contains SMILES strings and corresponding lipophilicity values.
- Data was split into three sets:
  - Training samples: 3024
  - Validation samples: 336
  - Test samples: 840



Task 1

- Overview: This task aims to fine-tune a pretrained chemical language model MoLFormer) on regression task.

- Dataset: the MoleculeNet Lipophilicity dataset, contains SMILES strings and corresponding lipophilicity values.
- Data was split into three sets:
  - Training samples: 3024
  - Validation samples: 336
  - Test samples: 840



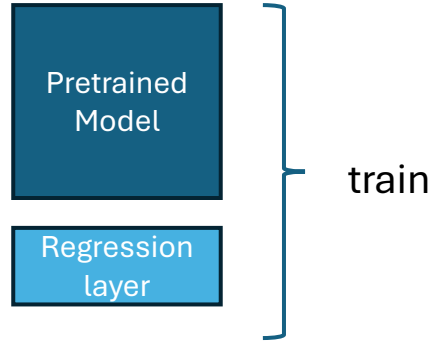
Task 1

- Maximum token length: 128 character
  - Decision based on dataset analysis.
  - We took 1000 string as a random sample.
  - Longest string was 127 char long.

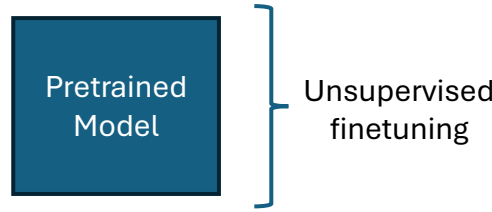
- Evaluation Metric: Root Mean squared error(RMSE)

Task 1: Method

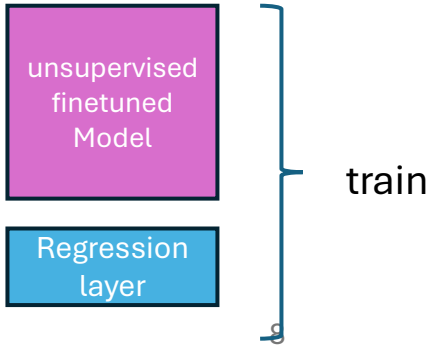
- Baseline Regression Fine-Tuning.
  - Load the pretrained model
  - Added regression head
  - Train the updated model on the training data



- Unsupervised finetuning using masked language modeling
  - Using combined training and validation datasets



- Regression Fine-Tuning.
  - Load the unsupervised fine-tuned model
  - Attach the regression head
  - Train the updated model on the training data



Task 1 : Training Parameters

- Epochs= 20
- Learning rate = 1e-7
- Batch size= 16

Task 1: Result

- Performance on the held-out test set.

Model	Test RMSE
Baseline Regression Model	1.1352
Finetuned Regression Model	1.1009

Task 1: Training

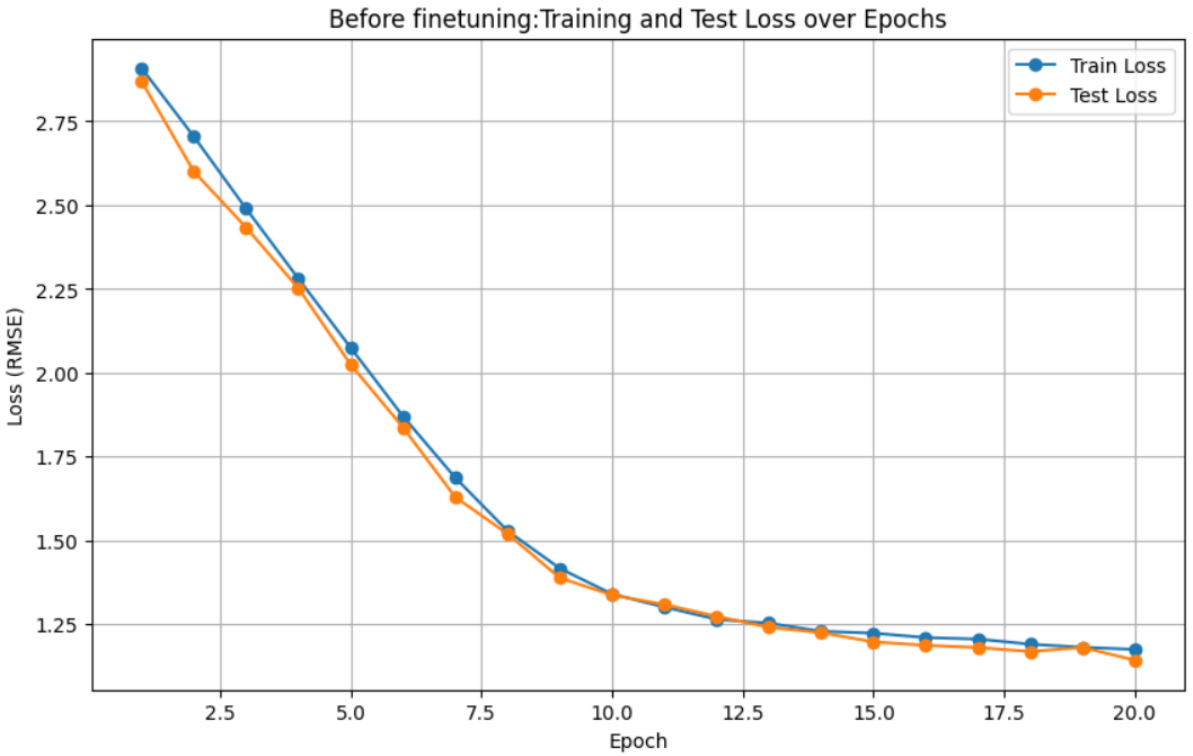


Figure 1

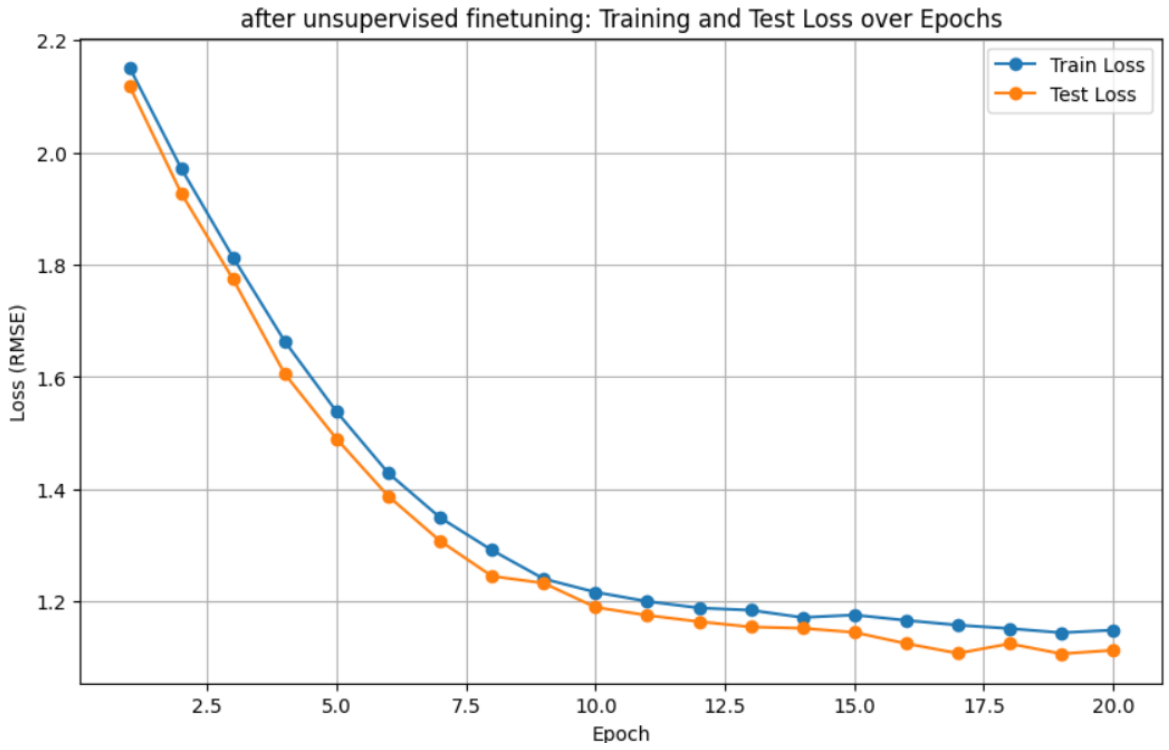


Figure 2

Task 2 - Overview

- **Objective:** Identify influential points from a secondary dataset
- **Technique:** Influence Function[1]
- **Dataset**: Secondary Lipophilicity dataset with 300 sample and similar structure as the dataset used for task-1

[1]**Koh and Liang, 2017**

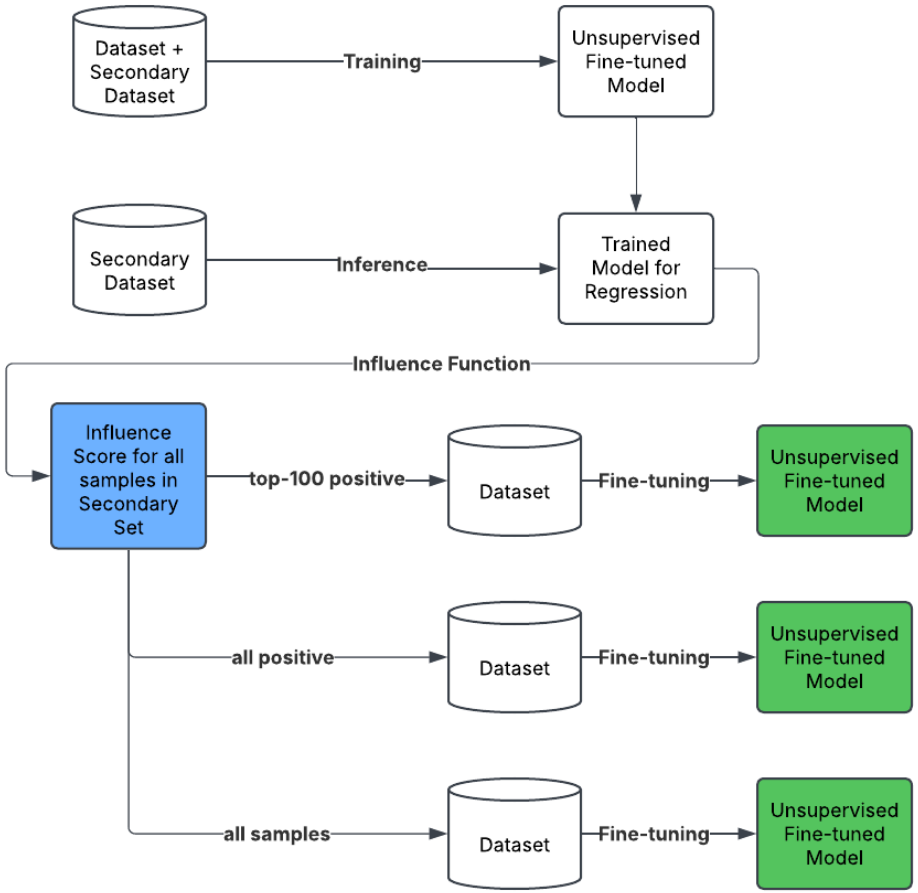
Task 2 – Influence Function

- Measure how much each training sample contributes to the prediction for a test sample
- Estimates the effect of removing or upweighting a sample efficiently without retraining the model multiple times.
- Selects the most influential samples using a heuristic (e.g., top-K).

Task 2 – Experimental Setup

- Initially computed influence for individual test samples (up to 12 points).
- To save time, used the loss gradient over the full test set instead of per-sample gradients.

Task 2 – Experimental Setup & Results



Data Selection	Test RMSE
Train + Secondary	1.0780
Train + Secondary (all positive)	1.1178
Train + Secondary (top 100 positive)	0.9426

Task 2 – Discussions

- Selecting highly influential samples improves fine-tuning.
- Not all positively influential samples improve performance.
- Using the loss gradient from the full test set works similarly to computing the loss gradient for each test sample separately.

Task 3: Overview

Objective: trying out different data selection methods and fine-tuning techniques and compare them

Data selection methods: Influence Function, TS-Dshapley[2]

Fine-tuning techniques: Parameter efficient fine-tuning strategies

[2]Schoch et al., 2023

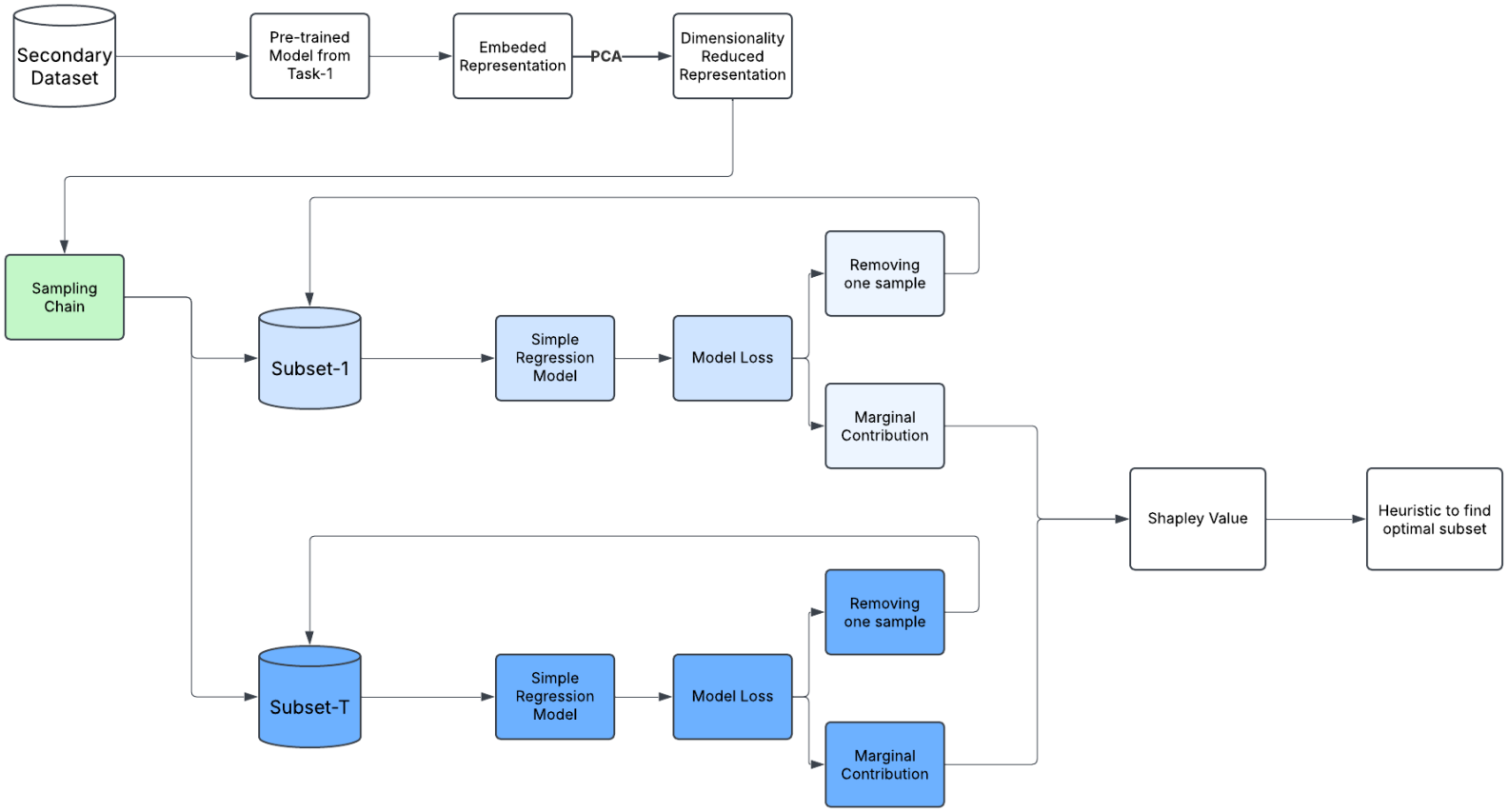
Task 3: TS-DShapley

- **Reduces computational cost** of Shapley-based data valuation by avoiding full retraining.
- Uses a **simpler proxy model** trained on representations from the target language model to approximate data values.
- **Monte Carlo sampling** is applied on **small subsets** instead of the full dataset, reducing complexity.
- Computes **marginal contributions** by iteratively removing data points from sampled subsets and aggregating their effects.

Task 3: TS-DShapley

- **Parallel computation** is possible since each Monte Carlo sampling chain is independent.
- Identifies the most useful samples by **iteratively removing low-contribution points** until an optimal subset is found for fine-tuning.

Task 3: Experimental Setup



Task 3: Comparison of Selected Sample by Data selection Methods

- Selected number of samples from the secondary dataset
  - TS-Dshapley = 159
  - Influence Function = 161 (all positive)
  - Influence Function = 100 (top-100)

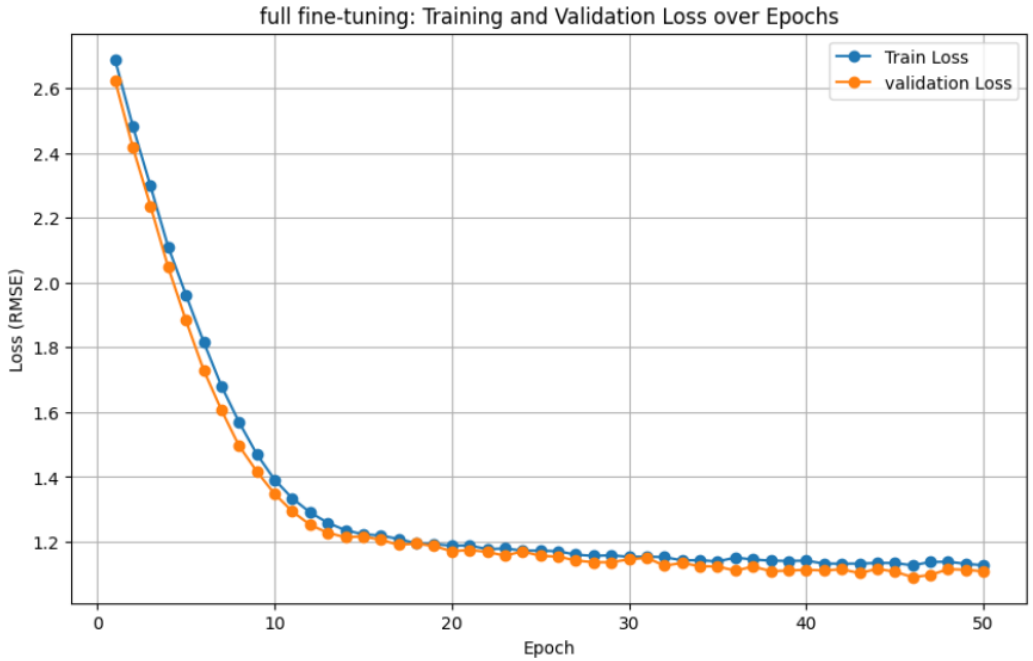
Task 3: Fine-tuning strategies

- Full Finetuning
- BitFit[3]
- LoRA[4]
- IA3[5]

[3]Ben-Zaken et al., 2022
[4]Hu et al., 2021
[5]Liu et al., 2022

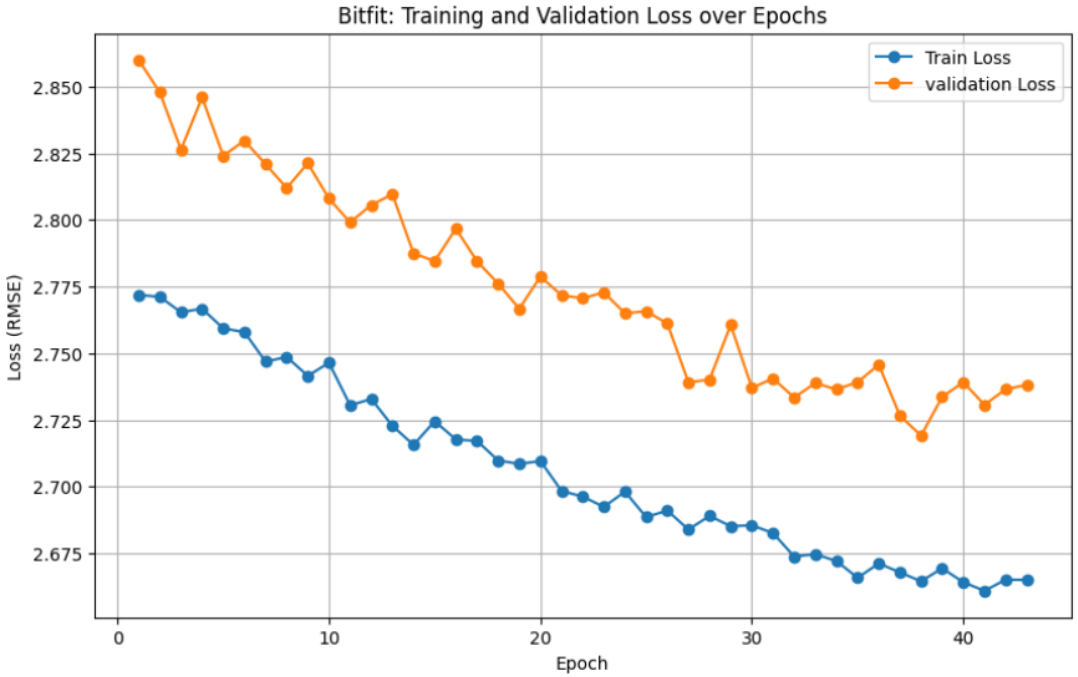
Fine-tuning strategies: Full Finetuning

- For baseline comparison.
- All model's parameters were updated



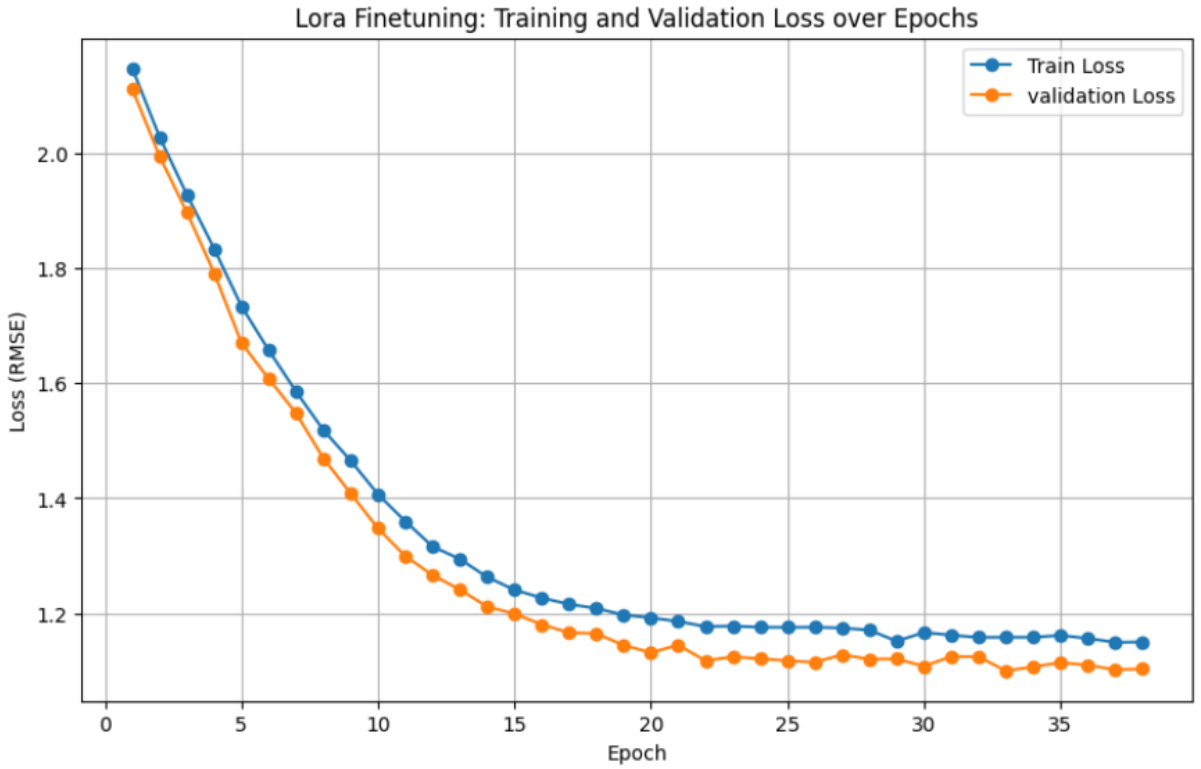
Fine-tuning strategies: BitFit

- Parameter-efficient approach for model adaptation.
- Updates only the model's bias parameters and task-specific layers



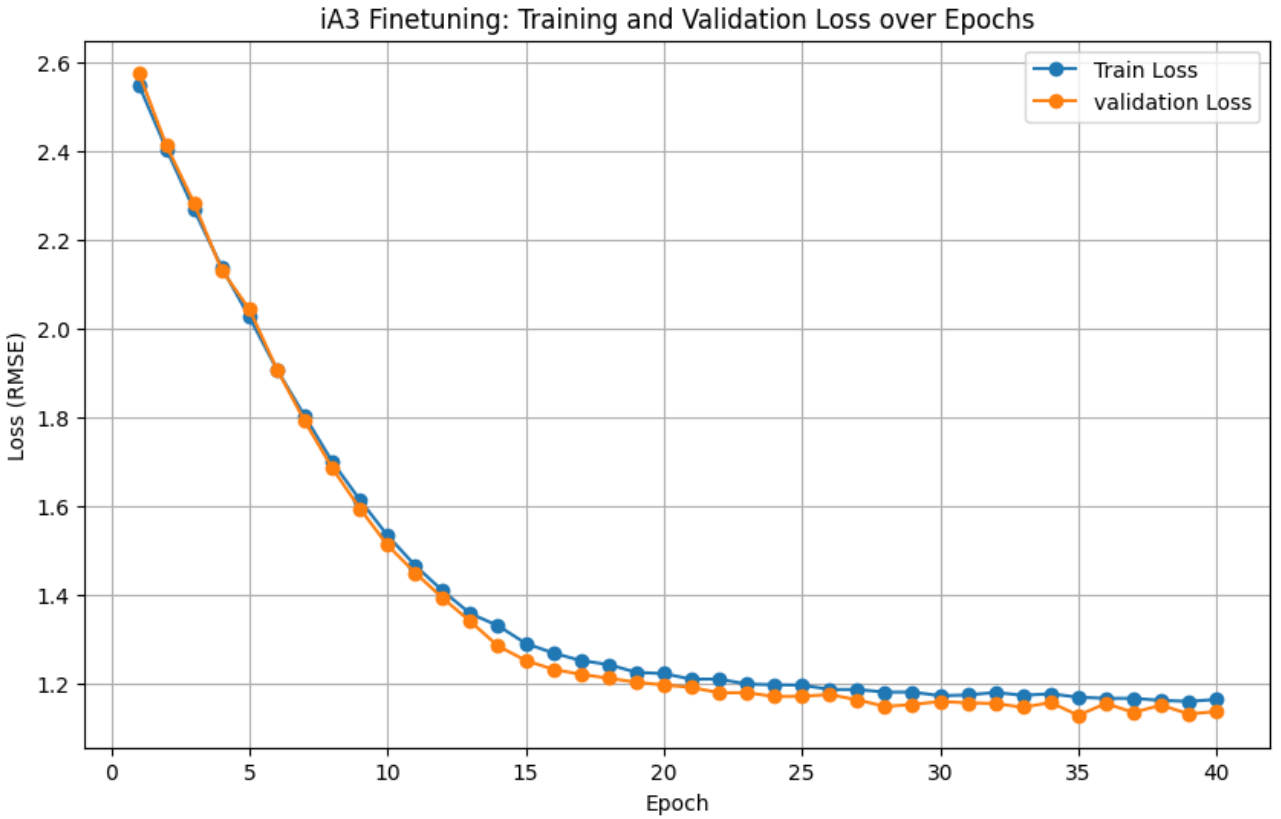
Fine-tuning strategies: LoRA

- Freeze the original model weights.
- Introduce small trainable low-rank matrices within the self-attention layers to approximate the weight changes of a layer.



Fine-tuning strategies: IA3

- Freeze all model weights.
- Introduce trainable scaling vectors that modify intermediate activations.
- Rescale keys, values, and feedforward activations in attention layers.



Hyperparameters and Finetuning

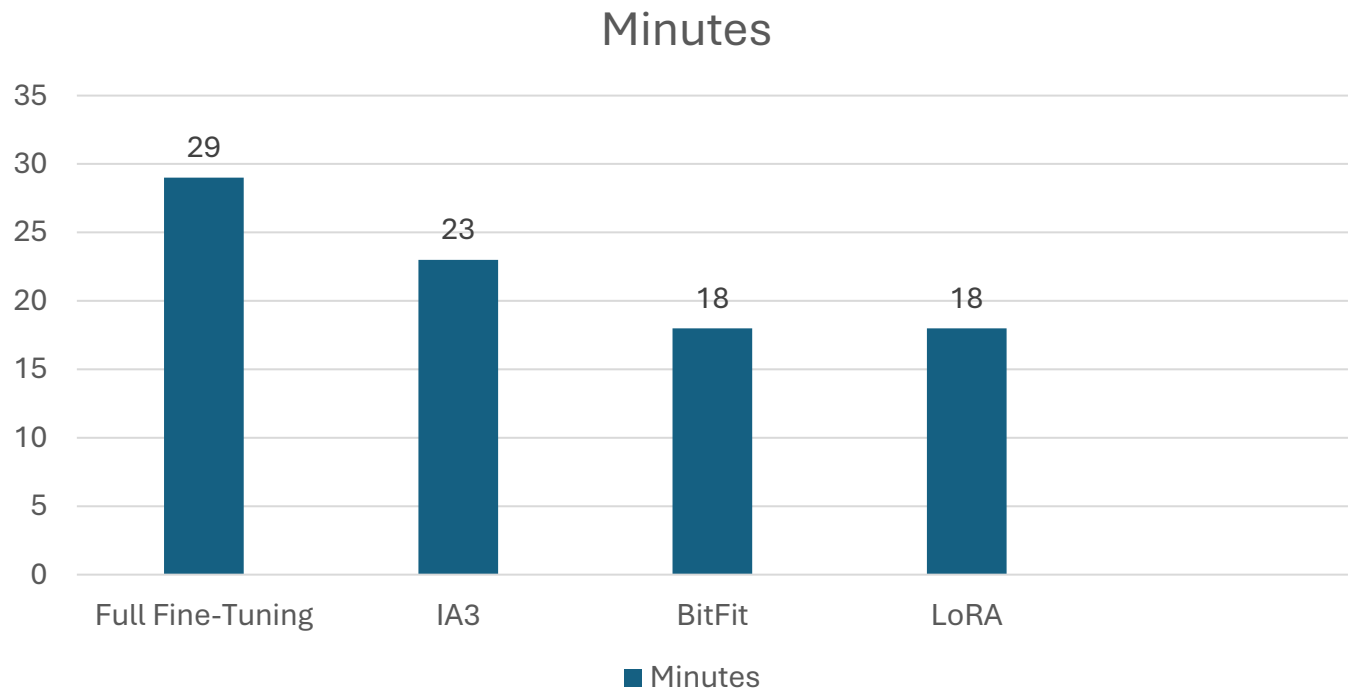
- LoRA Rank: 4
- LoRA Alpha: 8
- LoRA Dropout: 0.1
- Learning Rate: 1 × 10−7
- Batch size: 16
- Epochs: 50
- Early Stopping: Patience of 5 epochs with a minimum delta of 0.001

Task 3: Results Loss

Data Selection Strategy	Full Fine-Tuning	BitFit	LoRA	IA3
No Selection	1.0780	2.4351	1.0972	1.1514
Influence Method	0.9426	2.6366	1.1175	1.1469
TS-DShapley	1.2223	2.3313	1.471	1.1494

Task 3: Results: Training Computation time

- Computation time calculated using TS-Dshapley selection strategy.



Thank You!