Model Training Report: Milestone 3

I. RNN vs. FFNN Evaluation

1. Introduction This report presents a comparative evaluation of two neural network architectures—LSTM (Long Short-Term Memory) and MLP (Multi-Layer Perceptron)—for the task of Native Language Identification (NLI). The objective is to determine which model better captures language-specific features based on English writing produced by Japanese learners, utilizing the NICT-JLE corpus and BERT-based embeddings.

2. Experimental Setup

1). Dataset: NICT-JLE (Japanese Learners of English Corpus)

2). Embedding: BERT word embeddings

3). Learning Rate: 0.001

4). Epochs: 20

5). Loss Function: CrossEntropy Loss

6). Optimizer: Adam

7). Software: TensorBoard for performance tracking

3. Model Architecture and Input Preprocessing

For RNN structure, Long-Short Term Memory has been selected as it is more suitable to process time sequential data in long form. (128 tokens)

While for FFNN, MLP has been selected: with linear layer, ReLU for non-linear and Dropout to reduce the rate of overfitting.

For the validation set performance, the comparisons are presented as follows:

Accuracy

Model	Peak Accuracy	Smoothed Accuracy	Training Steps	Training Time per Step
LSTM	95.1%	93.75%	6,399	40.22 sec
MLP	87.24%	81.25%	6,399	45.58 sec

For Observations on accuracy:

LSTM shows significantly higher accuracy (+7.86% absolute improvement) compared to MLP, demonstrating superior sequence modeling capabilities. (95.1% vs 93.75%)

MLP exhibits larger fluctuations in smoothed accuracy (81.25% vs. 87.24%), indicating training instability.

Loss

Model	Smoothed Loss	Final Loss	Training Steps	Training Time per Step
LSTM	0.3707	0.3855	6,399	40.22 sec
MLP	0.3185	0.4583	6,399	45.58 sec

LSTM maintains stable loss reduction (final loss: 0.3855), and it shows a strong generalization.

MLP shows loss inconsistency (final loss: 0.4583 vs. smoothed loss: 0.3185), suggesting challenges in optimization and possibly lead to overfitting.

Performance Findings:

The records shows a superiority of LSTM:

1. Accuracy: LSTM outperforms MLP by 7.86% (95.1% vs. 87.24%).
2. Loss: LSTM achieves 15.8% lower final loss (0.3855 vs. 0.4583).

3. Efficiency: LSTM trains faster (40.22 sec/step vs. 45.58 sec/step).

Validation Set Accuracy Comparison:

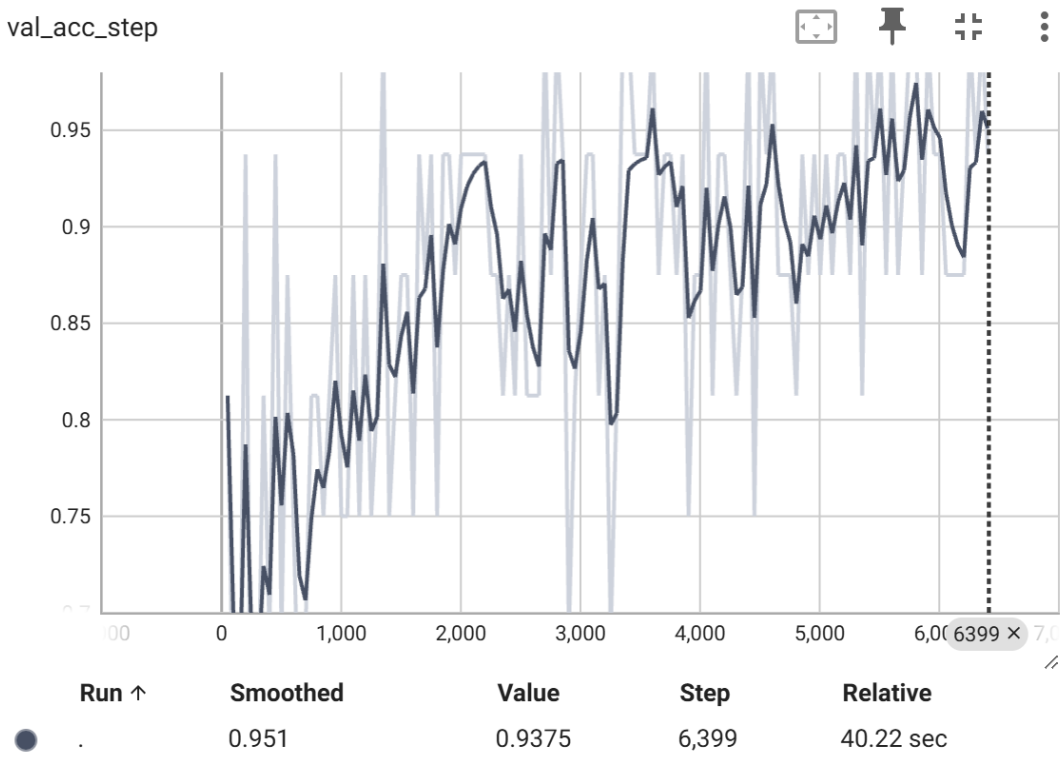

Figure 1. LSTM validation set accuracy record by step

LSTM validation accuracy (smoothed: 93.75%) shows steady improvement.

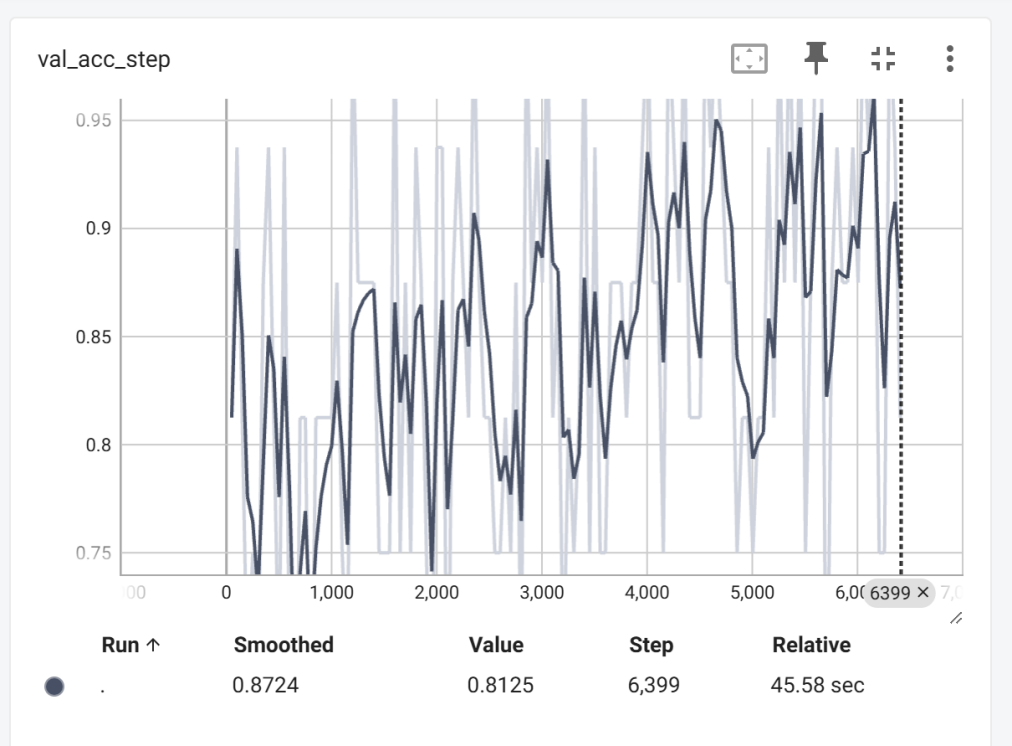

Figure 2. MLP validation set accuracy record by step

MLP validation accuracy (smoothed: 81.25%) exhibits significant fluctuations.

The performance of MLP shows a high fluctuation in validation accuracy—it shows a struggle in capture the relations in text data. Thus lead to high loss and lower accuracy.

Additionally, it shows a weakness in training when comparing with LSTM—LSTM's learning curve is more stable upward with less fluctuation.

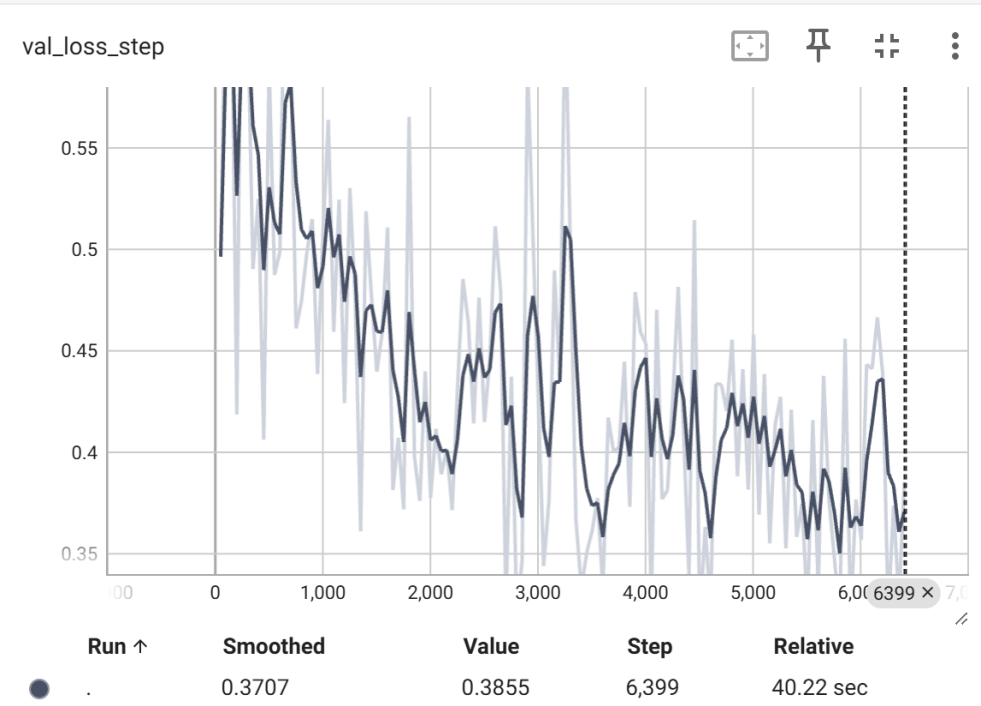

Figure 3. LSTM validation set loss record by step

LSTM loss declines smoothly, confirming training stability.

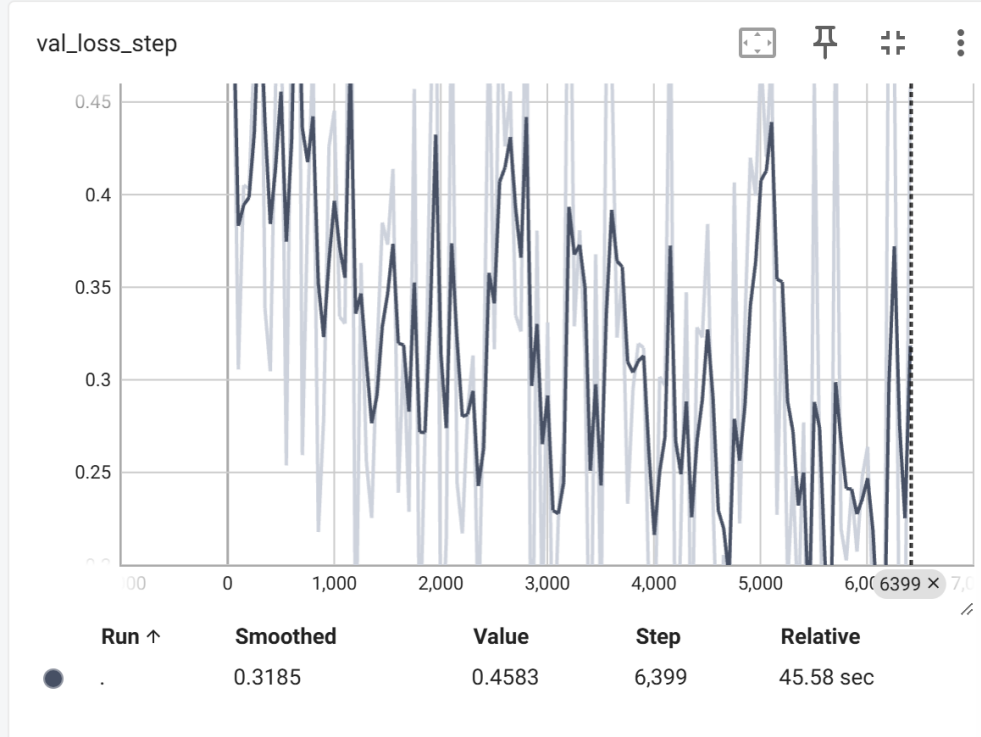

Figure 4. MLP validation set loss record by step

MLP loss goes up after initial improvement, indicating learning struggle.

Result:

LSTM out performs MLP in NLI tasks, reaching 95.1% validation accuracy and stable training curve. On the other side of the spectrum, MLP shows a weaker performance. Therefore, when dealing with time sequential text data, LSTM is strongly recommended.

II. CNN hand written letters evaluation

Introduction:

The CNN structure model training EMNIST dataset for hand written letter classification.

Tensor Board has been used to monitor the training and validation process.

2. Experimental Setup

1). Dataset: EMNIST letters

2). Vectorizer: transform. totensors

3). Learning Rate: 0.001

4). Epochs: 10

5). Loss Function: CrossEntropy Loss

6). Optimizer: Adam

7). Software: TensorBoard for performance tracking

8). Batch_size= 32

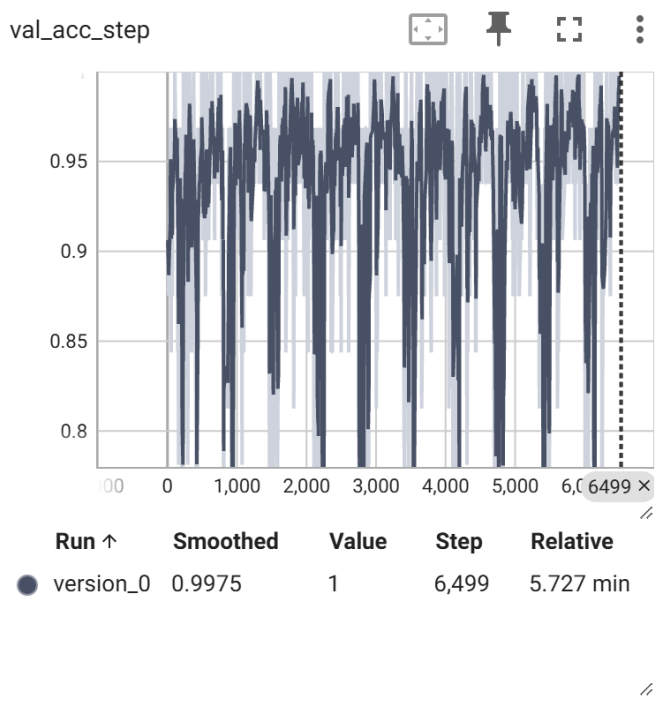

Figure 5. validation accuracy on step for CNN

The highest accuracy for step is 0.9975, which nears perfect—it shows a robust prediction performance in certain steps.

Figure 6. validation set accuracy for epoch for CNN

The highest accuracy is 0.9403, which is a bit lower than the step, but still in a high range.

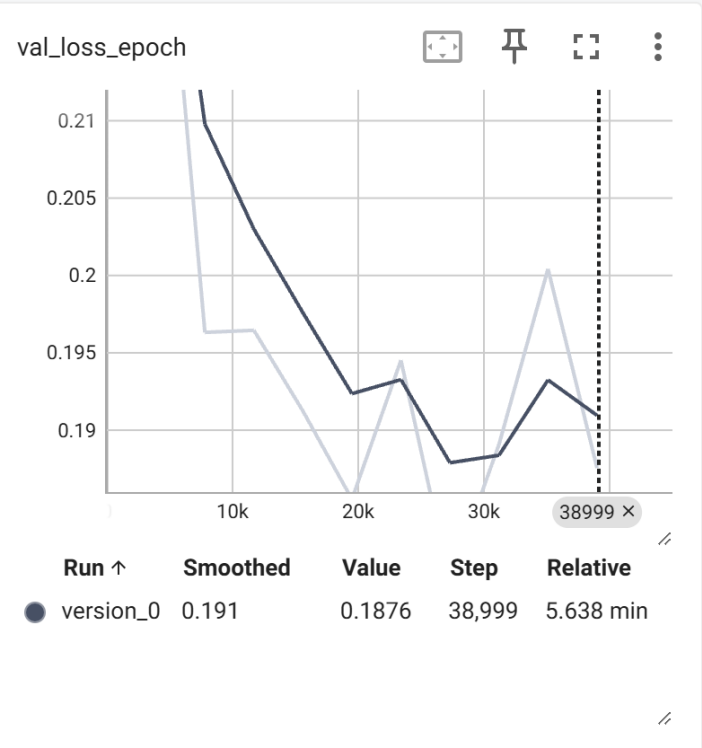

Figure 7. Validation loss for epoch for CNN

The loss for epoch shows the lowest is 0.1876, and there are fluctuations from the 26k steps—the loss bounces back to higher level and then goes down.

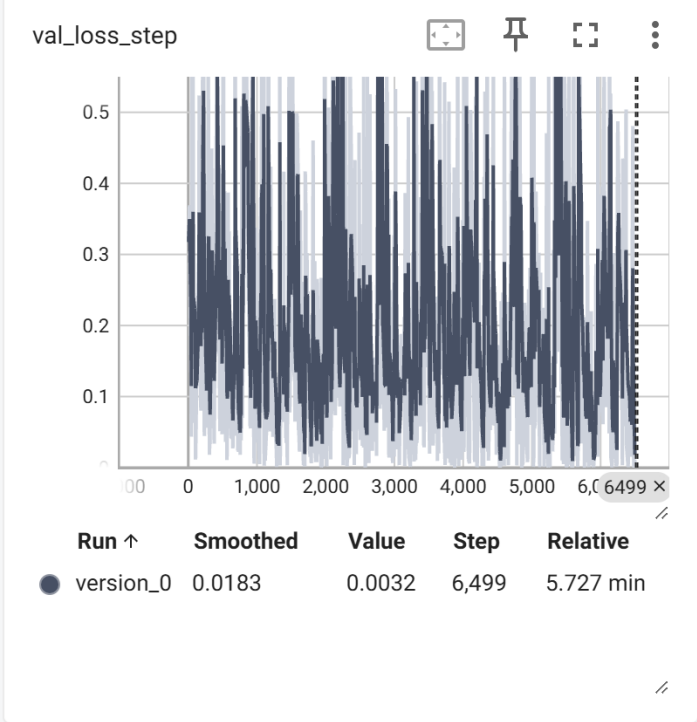

Figure 8. Validation loss for step for CNN

The lowest loss for step is 0.0032—it shows there are certain steps that could make predictions nearly perfect.

Each epoch takes about 5.6 minutes to accomplish, and it is a long time.

Conclusion:

The CNN model performs outstanding in validation set (94% of accuracy) but there are more methods could be applied (e.g. early stopping) to enhance the model.

III. Optimizer Evaluation

This part mainly covers evaluation of three different optimizer on LSTM NLI classifier.

Hyperparameters setup are same as part I except for the optimizers.

The optimizers are:

1. Adam

2. RMSprop
3. SGD

As Adam is the default optimizer for the model training in the part 1, the following figures will only cover performance on RMSprop and SGD. At last, there will be a figure listing the three optimizers performance.

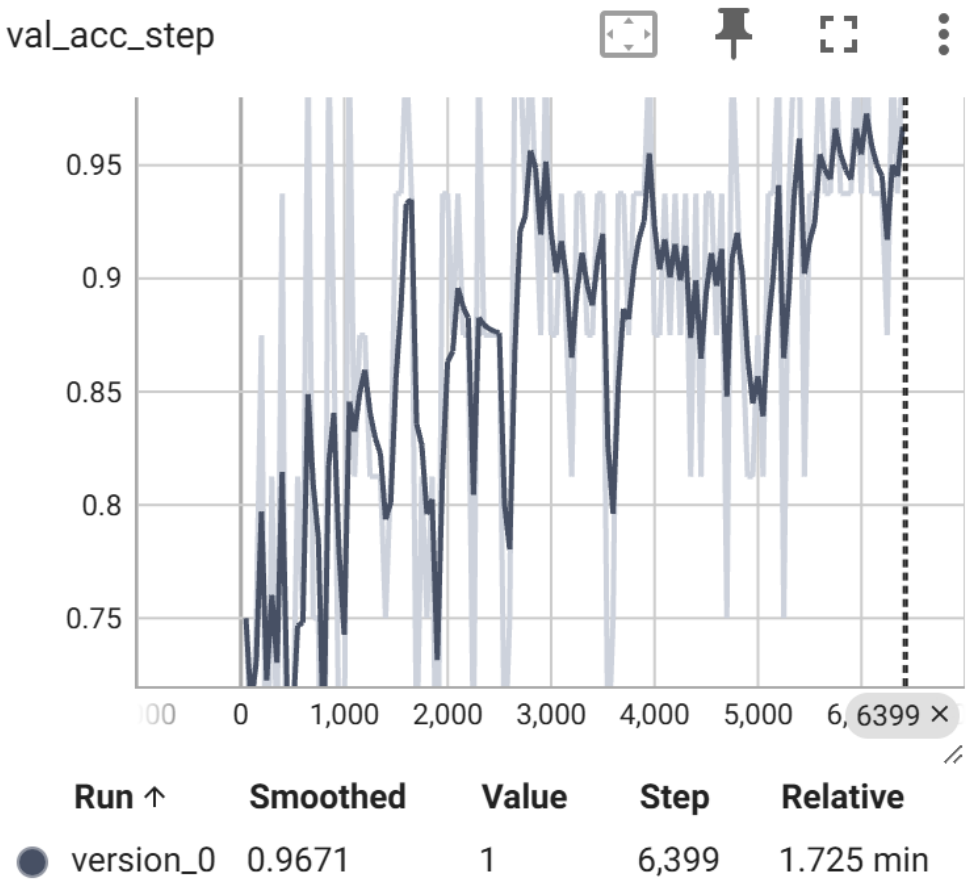

Figure 9. Validation set accuracy for LSTM using RMSprop optimizer

The final accuracy is 0.9671, during the validation process, the accuracy increased from 0.75-0.95+, highlighting that the model is being trained effectively.

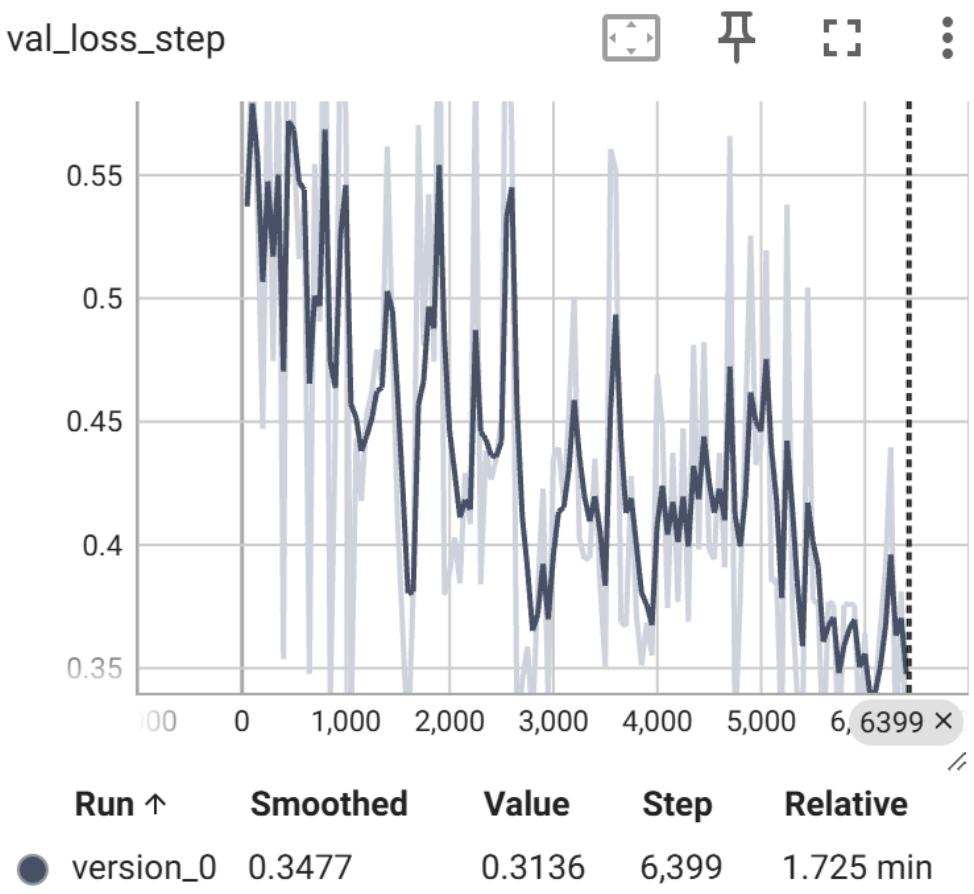

Figure 10. Validation set loss for LSTM using RMSprop optimizer

In terms of validation set loss, the final loss is 0.3477, while initially the loss is 0.55, but it drops to 0.35 and lower, performing a decent convergence during the training process.

Conclusion: RMSprop as a optimizer performs outstanding in LSTM-based Native Language Identification(NLI) task and it could also help with the model have nearly the State Of the Art (SOTA) performance with little overfitting.

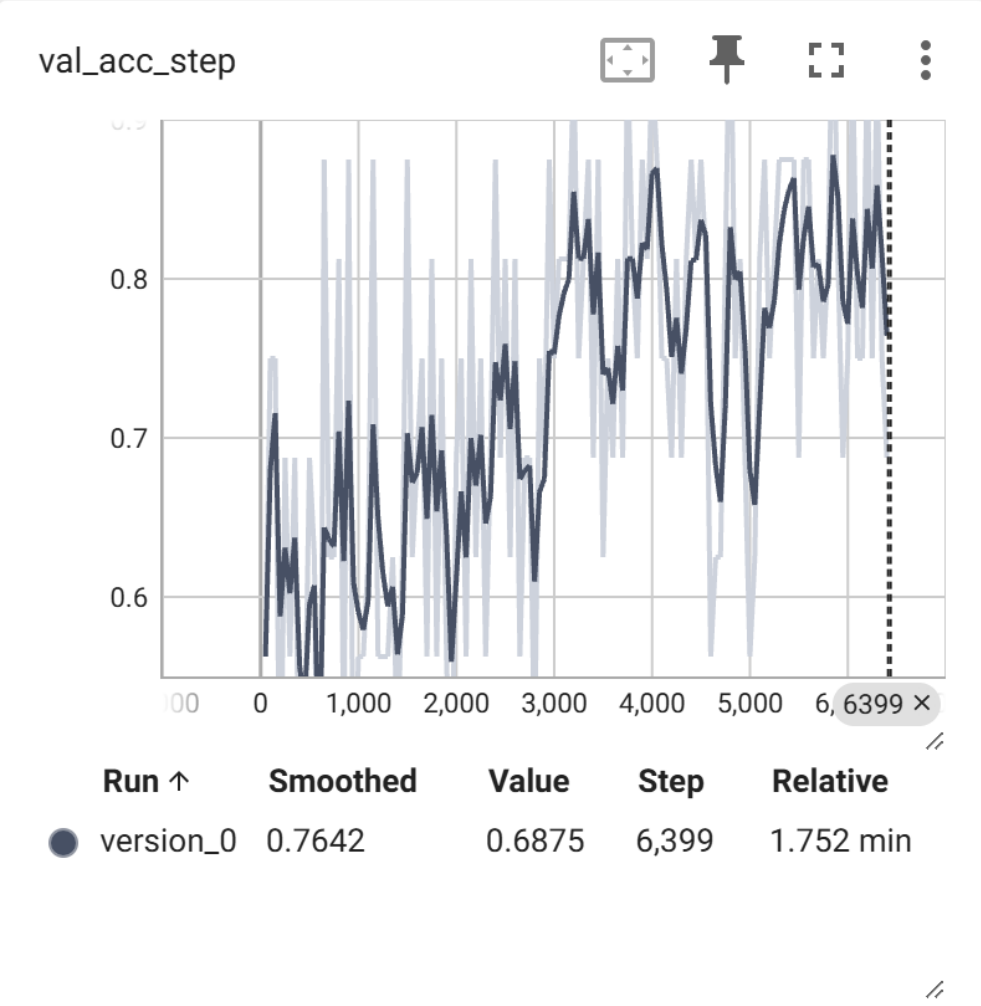

Figure 11. Validation set accuracy for LSTM using SGD optimizer

The accuracy starts at **0.5**, peaks at **0.7642** (step 6,399), with noticeable fluctuations during training

It shows a lower final accuracy compared to RMSprop (-20%)

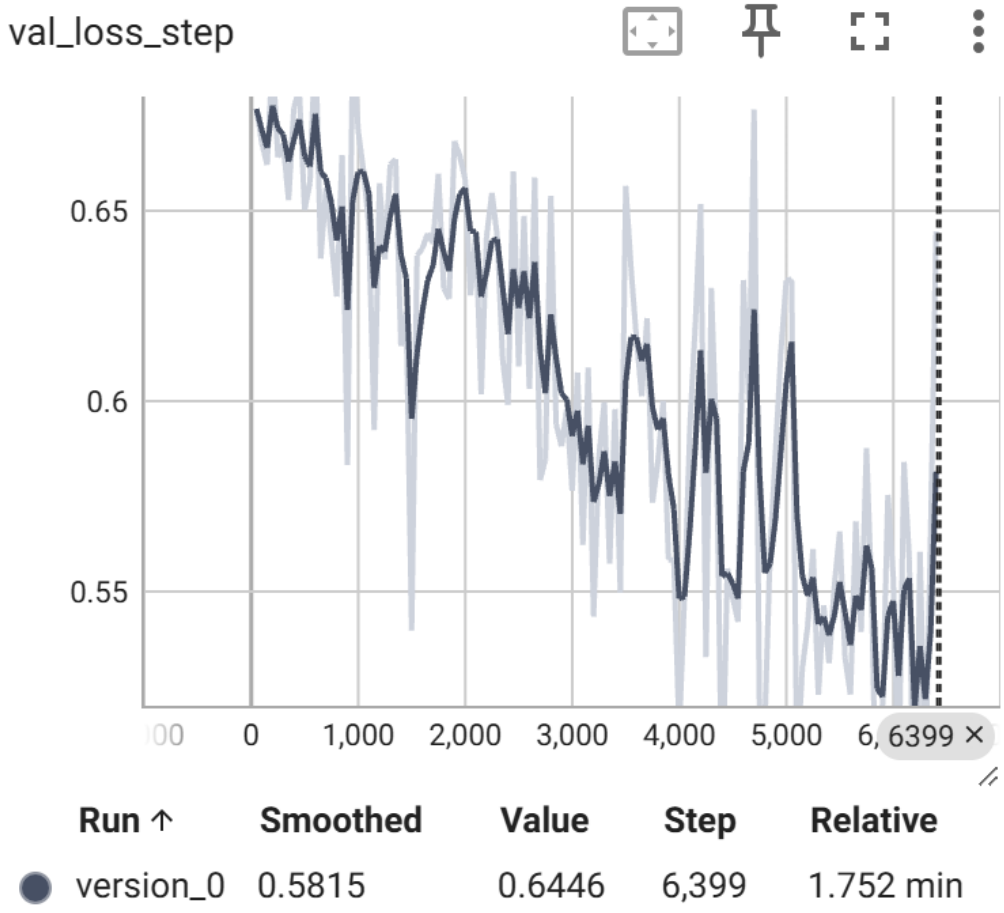

Figure 12. Validation set loss for LSTM using SGD optimizer\

Validation Loss Starts at 0.65 and plateaus around 0.5815 (step 6,399), indicating suboptimal convergence. Additionally, there is an obvious fluctuation in training process comparing with RMSprop.

SGD suggests struggles in gradient variability, and there is a poor loss reduction – indicating there is a unsatisfying performance on convergence.

Comparison on different optimizers.

Optimizer	Val Accuracy	Val Loss
Adam(default)	**95.1%**	**0.3707**
RMSprop	96.71%	0.3477
SGD	76.42%	0.5815

As I have covered the performance of Adam optimizer in Part 1, I will only simply summarize it in the chart and in this paragraph.

Accuracy in validation set: it is 95.1%, it is exceptionally well, but slightly behind RMSprop (96.71).

Loss: Adam performs a loss of 0.3707, slightly higher than RMSprop(0.3477), but still good.

Efficiency: 40 seconds/ epoch is much higher than RMS prop (1.725s/ epoch) and SGD (1.752s/epoch).

Conclusion: Both Adam and RMS prop shows an outstanding performance: smooth and stable learning. However, SGD shows a big fluctuation, suggesting a poor adaptation to model training.

RMS prop is the strongest in model performance, Adam is slightly behind, while SGD is the worst. The cause might be that Adam and RMSprop are able the do adaptive learning.