Notes on the training optimisation

Emma Ai

September 29, 2025

1 The essential changes related to the model optimisation

The optimisation of the GPT-2 style model proceeded through a sequence of targeted changes, each motivated by improving training stability, convergence, and generalisation performance. The key changes, their rationale, and the corresponding results are summarised below.

1.1 Optimiser Adjustment

- **Change:** Replace *SGD* with *AdamW*.
- **Reasoning:** AdamW decouples weight decay from the gradient update and uses per-parameter adaptive learning rates, which improves convergence stability and generalisation over plain SGD. In GPT-2 style transformers, many parameters experience *sparse or infrequent* updates (e.g., embeddings and attention projections on subpopulations of tokens); AdamW's adaptive scaling handles such sparsity far better than SGD with a single global learning rate, while decoupled weight decay provides cleaner L_2 regularisation of dense and sparse weight vectors alike.
- **Results:** Validation loss was reduced to 1.4893. The validation and training losses curves are shown in Fig. 1, run `mainrun_2025-09-22T12-05-47.log`.

1.2 Hyperparameter Tuning with Optuna

- **Change:** Introduce *Optuna* [1] to sweep learning rate, dropout rate, and block size.
- **Additional Setting:** $weight\_decay = 0.01$ in AdamW.
- **Reasoning:** Automated search provides systematic exploration of hyperparameters that strongly affect overfitting and stability. The implementation leveraged the official Optuna GitHub repository: `https://github.com/optuna/optuna`.
- **Results:** Validation loss was reduced to 1.3544 with the set hyperparameters after tuning. The validation and training losses curves are shown in Fig. 1, run `mainrun_2025-09-23T03-25-45.log`.

1.3 Sharpness-Aware Minimisation (SAM) and Warm-Up

- **Change:** Add SAM optimiser [2] and introduce learning-rate warm-up schedule.
- **Reasoning:** SAM encourages flatter minima and improved generalisation; warm-up stabilises early training. The PyTorch implementation `https://github.com/davda54/sam` is used.
- **Results:** Validation loss was reduced to 1.327 with the set of hyperparameters after tuning. The validation and training losses curves are shown in Fig. 1, run `mainrun_2025-09-23T08-09-24.log`.

1.4 Adaptive SAM

- **Change:** Replace standard SAM with adaptive SAM [3].
- **Reasoning:** Adaptive SAM adjusts perturbations based on parameter magnitudes, yielding better scaling for deep networks, specifically for large language models (LLMs) where weight vector scales can vary substantially across layers. The PyTorch implementation `https://github.com/davda54/sam` is used.
- **Results:** Validation loss was reduced to 1.311 with the same set of hyperparameters as SAM. The validation and training losses curves are shown in Fig. 1, run `mainrun_2025-09-23T09-36-58.log`.

1.5 Model Size Reduction (Phase I)

- **Change:** Reduce model size: $vocab\_size = 4\text{K}, n\_head = 6, d\_model = 384$.
- **Additional Setting:** Exclude weight decay for bias, norm, and embedding layers. Plato minimum learning rate at 1×10^{-5}.
- **Reasoning:** The training corpus contains only $\sim$1.1 M tokens. Smaller models reduce capacity (mitigating overfitting on limited data) and lower computational load. Excluding weight decay for bias, LayerNorm, and embedding parameters preserves scale-sensitive statistics and representational capacity.
- **Results:** Validation loss was reduced to 1.2935 with the set of hyperparameters after tuning. The validation and training losses curves are shown in Fig. 1, run`mainrun_2025-09-24T08-55-04.log`.

1.6 Model Size Reduction (Phase II)

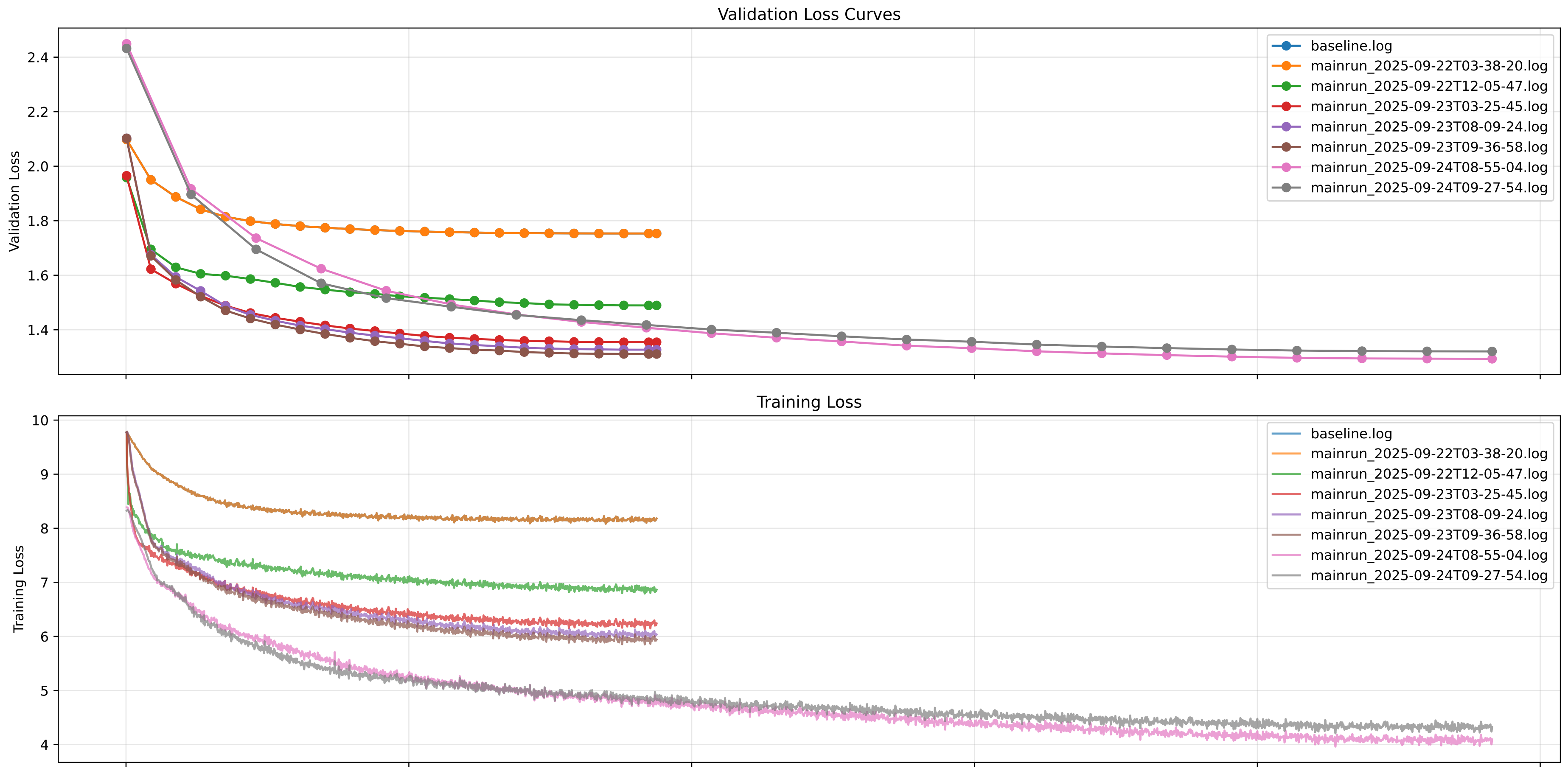
- **Change:** Further reduce depth and width: $n\_layer = 4, n\_head = 3, d\_model = 192$.
- **Reasoning:** Experiment with lightweight configurations to test lower bounds of model capacity while retaining stable training.
- **Results:** Validation loss was increased to 1.3206 with the set of hyperparameters after tuning. The validation and training losses curves are shown in Fig. 1, run `mainrun_2025-09-24T09-27-54.log`.

Held-out evaluation on in-domain titles. The two reduced-size models were evaluated on 20,000 titles held out from the same corpus (not used in training). The results are on par with validation performance; perplexity is reported as ppl = exp(loss).

Table 1: Held-out results on 20,000 in-domain titles.

Checkpoint	**Loss**	**Perplexity**
`models/model_midsize.pth`	1.3021	3.6771
`models/model_smallsize.pth`	1.3302	3.7816

Figure 1: GPT-2 Model Optimisation Experiments



Model Meta Info Table

	n_layer	n_head	d_model	block_size	batch_size	vocab_size	dropout	lr	weight_decay	min_lr	warmup_ratio	epochs	parameters_count	device	final_val_loss
baseline.log	6	8	512	128	64	16000	0.1	0.006	0.0			7	27172864	cpu	1.7533
mainrun_2025-09-22T03-38-20.log	6	8	512	128	64	16000	0.1	0.006	0.0			7	27172864	cpu	1.7533
mainrun_2025-09-22T12-05-47.log	6	8	512	128	64	16000	0.1	0.006	0.0			7	27172864	mps	1.4893
mainrun_2025-09-23T03-25-45.log	6	8	512	128	64	16000	0.2012	0.0003	0.01			7	27172864	mps	1.3544
mainrun_2025-09-23T08-09-24.log	6	8	512	128	64	16000	0.153	0.0005	0.01			7	27172864	mps	1.327
mainrun_2025-09-23T09-36-58.log	6	8	512	128	64	16000	0.153	0.0005	0.01			7	27172864	mps	1.311
mainrun_2025-09-24T08-55-04.log	6	6	384	64	64	4000	0.051	0.0011	0.01	1e-05	0.1	7	12208128	mps	1.2935
mainrun_2025-09-24T09-27-54.log	4	3	192	64	64	4000	0.0603	0.0021	0.01	1e-05	0.1	7	2560128	mps	1.3206

2 The other essential code and configurations

- `helper.py`: Utilities to plot the model *architecture* and diagnose *tokeniser* efficiency (coverage, OOV rate, length stats).
- `model_analysis.py`: Reproduces Fig. 1 (training/validation loss curves, run annotations).
- `model_evaluate.py`: Evaluates saved checkpoints on held-out data and reports loss/perplexity tables.
- `train_config.yaml`: Experiment configuration (model hyperparameters, tune/train, hyperparameter DB and model saving path).
- `hyperparameter.db`: SQLite database (Optuna study) tracking trials, sampled hyperparameters, and best scores.
- `hptune.sh`: A simple shell to run N trials.

The full codebase including running environment (updated docker file and `docker_compose.yml`) to replicate these experiments is in repo `https://github.com/emmaai/mainrun`.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[3] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5905–5914. PMLR, 2021.