Project 1: Hyperparameter Tuning

In this project, I fine-tuned a DistilBERT model on the MRPC paraphrase detection task from the GLUE benchmark. The goal was to systematically optimize the model's hyperparameters to maximize validation accuracy through three stages: identifying the three most impactful hyperparameters, manually tuning them and then applying an automated tuning. I tracked my runs with Weights & Biases (W&B).

Assumptions

For this project, the following assumptions were made to guide the experiments and analysis:

1. I assumed that meaningful improvements in performance arise mainly from interactions between hyperparameters rather than from individual changes.
2. I assumed that validation accuracy alone is sufficient to assess model performance for the MRPC task. Other metrics like F1-score or precision/recall were not considered critical, given that the dataset is relatively balanced and the main goal was overall classification correctness.

Additional Parameters

To improve training control, I introduced three additional hyperparameters:

- **Adam epsilon** - stabilizes the optimizer by preventing division by very small numbers.
- **Max sequence length** - controls input truncation/padding, affecting both efficiency and model performance.
- **Gradient clipping -** prevents large updates that could destabilize training.

This resulted in ten total parameters, with seven candidates for tuning: learning rate, weight decay, Adam epsilon, warmup steps, dropout, max sequence length, and gradient clipping. I fixed epochs (3), as required by the task, and fixed the batch sizes to ensure comparability and reduce search complexity.

Methodology

To see which hyperparameters most affected model accuracy, I used a systematic baseline comparison:

1. Established a baseline run with default values.
2. For each parameter, ran two variations (lower and higher) while keeping others constant.
3. Compared results in W&B to measure deviations from baseline.
4. Selected the three parameters with the greatest performance impact.

This one-factor-at-a-time approach made it easy to isolate each parameter's influence.

Three Most Important Hyperparameters

These were the three most impactful hyperparameters:

1. **Learning rate** - Most impactful; values from 2e-05 to 2e-04 caused major accuracy differences.
2. **Warmup steps** - Influenced early stability and convergence; 50-100 steps performed best.
3. **Weight decay** - Affected generalization; optimal range [0.05, 0.1, 0.2].

Manual Fine-Tuning

I manually fine-tuned the three selected hyperparameters using a coarse grid search:

- Learning rate: [1e-5, 2e-5, 5e-5]
- Weight decay: [0.0, 0.01, 0.05]
- Warmup steps: [100, 200]

This resulted in 18 total combinations (17 new runs after baseline). This approach captured interactions between parameters, such as how optimal weight decay depends on learning rate. After refining top-performing configurations, my result with the highest validation accuracy was: **Learning rate = 2e-5, Weight decay = 0.0, Warmup steps = 50**.

Automatic Fine-Tuning

For automated optimization, I used W&B Sweeps with Bayesian optimization, which learns from prior results to explore promising regions of the hyperparameter space.

Defined search ranges:

- Learning rate: 1e-5–5e-5 (log uniform)
- Weight decay: 0.0–0.1 (uniform)
- Warmup steps: 0–500 (integer uniform)

With a budget of 20 runs, Bayesian optimization gradually improved validation accuracy, identifying a strong configuration: **Learning rate = 2.01e-5, Weight decay = 0.0797, Warmup steps = 14**. This setup revealed that very low warmup values with moderate weight decay could outperform typical manual configurations, illustrating the benefit of automated exploration.

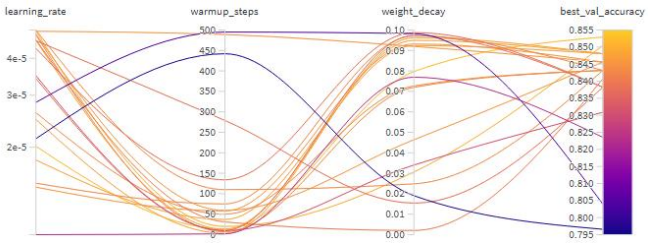

Figure 1: Parallel coordinates plot of hyperparameters vs. validation accuracy; lighter lines indicate higher accuracy.

Results

The “baseline configuration” provided by the teacher achieved a validation accuracy of **0.85049**. The best configuration from manual tuning achieved a validation accuracy of **0.85784** (learning rate 2e-5, weight decay 0.0, warmup steps 50), while the best run from the automated sweep reached **0.85294** (learning rate 2.01e-5, weight decay 0.0797, warmup steps 14). Both approaches successfully improved upon the baseline, with the manual method achieving a 0.7 percentage point improvement and the automated method a 0.2 percentage point improvement. However, the automated sweep showed steady improvement throughout the 20 runs, indicating it would likely have found better configurations with additional runs. The manual approach benefited from targeted refinement around known good values, while the Bayesian optimization discovered unconventional parameter combinations that also performed well.

Figure 2: Training metrics of the best run logged on Weights & Biases, showing loss and accuracy trends over three epochs.

Reflection

What Went Well:

The systematic three-stage approach (baseline comparison, manual fine-tuning and automated sweep) effectively identified the most impactful hyperparameters and improved validation accuracy over the baseline. Using Weights & Biases made tracking and comparing runs straightforward, especially because I already had some experience with it. The manual grid search successfully captured interactions between key parameters and the automated sweep highlighted unconventional but effective configurations that manual tuning might have missed.

Areas for Improvement:

The 20-run budget limited the Bayesian optimization's potential - more runs would likely have found better configurations. I would also warm start the automated sweep with the best manual results and explore a wider range of warmup steps, as the automated approach discovered that very low values worked surprisingly well. Additionally, further experimenting with learning rate schedules or alternative optimization strategies could help achieve more consistent improvements.