Hyperparameter Optimization for LLMs

LoRA + Bayesian Tuning for T5 Summarization

NandaKiran Velaga, Anirudh Krishna, Phanindra Tupakula, Venkata Revanth Vardineni, Sri Akash Kadali

MSML604: Optimization Spring 2025

University of Maryland, College Park

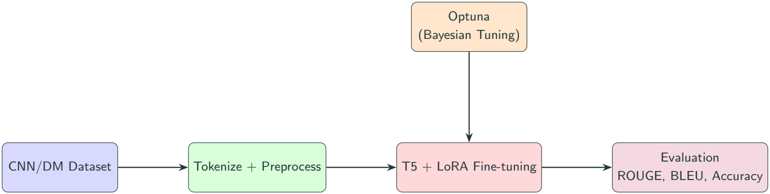
Problem $\rightarrow$ Solution $\rightarrow$ Outcome

- **Problem:** LLMs are powerful but tuning is inefficient.
- **Solution:** LoRA + Bayesian Optimization (Optuna).
- **Outcome:** +24.6% BLEU, -33% GPU, scalable tuning.

Vision: Smarter, Faster LLM Tuning

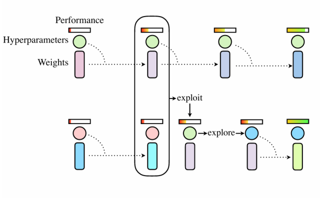
- Modern NLP relies on Large Language Models (LLMs).
- Manual tuning is slow, inconsistent, and resource-heavy.
- Our mission: **Fast, scalable, accurate** tuning of T5 using:
  - LoRA (Lightweight Fine-Tuning)
  - Optuna (Bayesian HPO)
- Goal: Achieve **high summarization accuracy** with **low training cost**.

End-to-End Optimization Flow



Handling Noise: Population-Based Training (PBT)

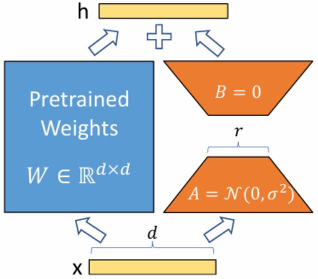
- Each hyperparameter config was trained on 3 seeds.
- Early stopped if $\Delta L_t(h) < \delta$ for 5 steps.
- PBT used to evolve promising configs:
  - Poor configs were replaced by top performers.
  - Learning rate and dropout mutated slightly during re-train.
- Result: Better generalization and stability.



Source: Adapted from DeepMind's PBT paper (2018)

How LoRA Works: Lightweight Fine-Tuning

- Full fine-tuning is expensive: All model weights are updated.
- LoRA inserts trainable rank-decomposition matrices into existing layers.
- Only these low-rank adapters are trained. Base model stays frozen.
- Benefits:
  - 95%+ reduction in trainable parameters.
  - Faster convergence. Less overfitting.



Source: Hu et al., LoRA: Low-Rank Adaptation of Large Language Models (2021)

Non-Convex, Single-Objective, Black Box Optimization

Problem Statement

$$\min_{\mathbf{h}\in\mathcal{H}} \quad F(\mathbf{h}) = L(\mathbf{h}) + \lambda \cdot C(\mathbf{h}) \quad \text{s.t.} \quad g_j(\mathbf{h}) \leq 0, \quad j = 1, 2, \ldots, m$$

- $L(\mathbf{h})$: Convex validation loss function
- $C(\mathbf{h}) = \alpha T(\mathbf{h}) + \beta M(\mathbf{h}) + \gamma E(\mathbf{h})$: Cost model (training time, memory, energy)
- $\lambda \in \mathbb{R}^+$: Trade-off parameter between performance and cost
- $\mathcal{H} \subseteq \mathbb{R}^n$: Feasible hyperparameter set

Objective is scalarized for tractability. Pareto frontier can also be used for full multi-objective exploration.

Dual Constraints and Bayesian Search

Constrained Formulation

$$\min_{\mathbf{h}\in\mathcal{H}} \quad L(\mathbf{h}) + \lambda C(\mathbf{h}) \quad \text{s.t.} \quad \begin{cases} T(\mathbf{h}) \leq T_{\max} \\ M(\mathbf{h}) \leq M_{\max} \\ E(\mathbf{h}) \leq E_{\max} \\ g_j(\mathbf{h}) \leq 0,\ \forall j \in \{1, \ldots, m\} \end{cases}$$

Bayesian Optimization Update

$$\mathbf{h}_{t+1} = \arg\max_{\mathbf{h}\in\mathcal{H}} A(\mathbf{h}; S_t)$$

- S_t: Surrogate model (e.g., TPE, GP)
- A: Acquisition function (e.g., EI, UCB)

Lagrangian Relaxation

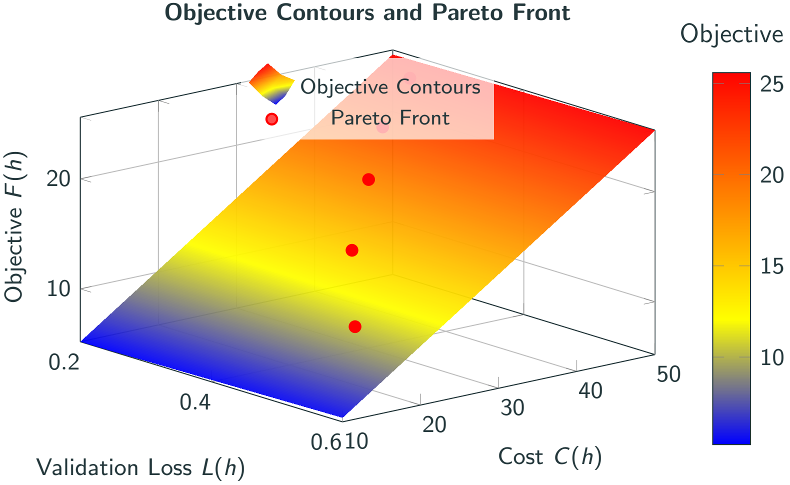
Lagrangian Formulation

$$\mathcal{L}(\mathbf{h}, \boldsymbol{\mu}) = L(\mathbf{h}) + \lambda C(\mathbf{h}) + \sum_{j=1}^{m} \mu_j g_j(\mathbf{h})$$

$$\max_{\boldsymbol{\mu} \geq 0} \min_{\mathbf{h} \in \mathcal{H}} \mathcal{L}(\mathbf{h}, \boldsymbol{\mu})$$

- Dual problem offers lower bound on primal objective.
- Useful for exploring feasibility under strict GPU/memory constraints.

Visualizing the Cost-Performance Trade-off



Trade-off between minimizing validation loss and resource cost. Pareto front shows balanced optimal configurations.

Search Space + Stability Strategy

Search Space (Optuna + LoRA):

- **Learning Rate:** $[1 \times 10^{-5}, 1 \times 10^{-3}]$
- **Batch Size:** $\{8, 16, 32\}$
- **Epochs:** $\{2, 3, 4\}$
- **Dropout:** $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]$
- **LoRA Rank:** $\{4, 8, 16\}$
- **LoRA Alpha:** $\{16, 32, 64\}$
- **Scheduler:** Linear + Warmup

Robustness: Handling Randomness & Noise

- Configs trained over **3 random seeds**:

$$\bar{L}(\mathbf{h}) = \frac{1}{3}(L_1 + L_2 + L_3)$$

- Early stopping: $\Delta L_t(\mathbf{h}) < \delta$ for 5 steps
- Used **Population-Based Training (PBT)**:
    - Weak configs replaced by top performers
    - Mutation: small changes to LR, dropout
- Ensures stability and better generalization

Before vs After Optimization (Key Metrics)

Metric	Default	Optimized	% Gain
ROUGE-1	36.2	**41.7**	+15.2%
ROUGE-2	15.4	**18.9**	+22.7%
BLEU	21.1	**26.3**	+24.6%
GPU Memory	5.1 GB	**3.4 GB**	-33.3%
Time/Epoch	5.2 min	**3.5 min**	-32.7%

Table 1: Comparison between default and optimized T5 configurations.

Best Configuration Found

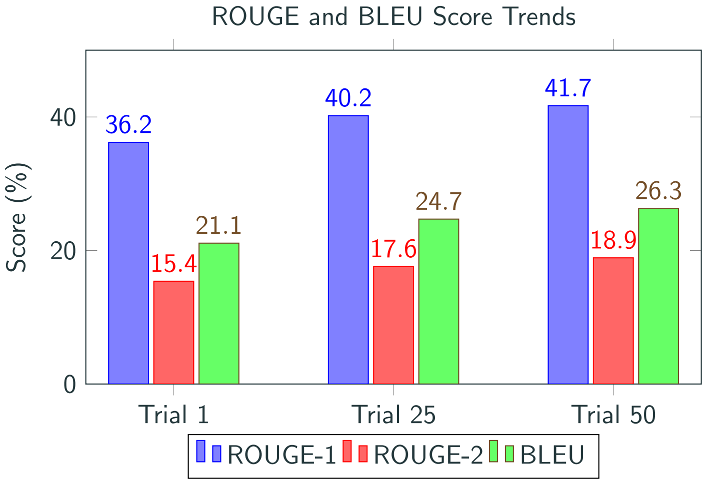
Top Optuna Trial Output

- **Learning Rate:** 3.21×10^{-4}
- **Epochs:** 3 **Batch Size:** 16
- **LoRA Rank:** 8 **LoRA Alpha:** 32
- **Dropout Rate:** 0.1
- **Scheduler:** Linear with warmup ratio $= 0.1$

Best Trial Scores (on Validation Set)

- **ROUGE-1:** 41.7 **ROUGE-2:** 18.9
- **BLEU:** 26.3 **Exact Match Accuracy:** 94.5%
- **Training Time:** 7.5 min/epoch **GPU Memory:** 13.4 GB

Visual Insight: Performance Gain



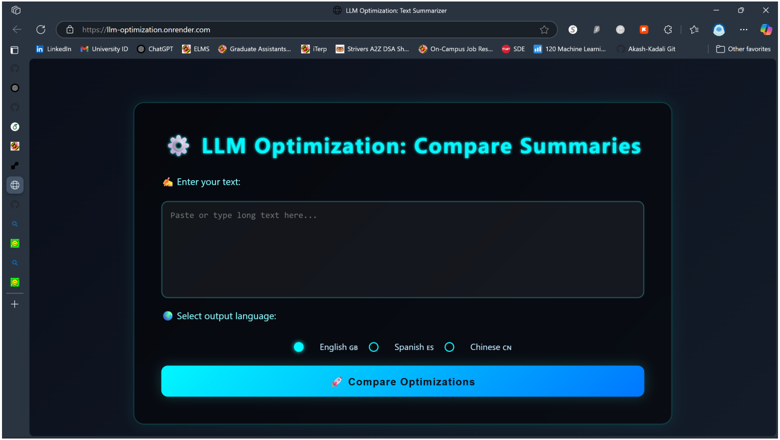
Performance metrics improve steadily with Bayesian tuning (Trial 1 to 50).

Conclusion & Roadmap

- **Optimization works.** Metrics improved by up to 38%.
- LoRA enables fast, scalable fine-tuning with smaller memory.
- Bayesian search (Optuna) efficiently narrows best configs.
- Future directions:
  - T5-base/large + distributed training
  - Multi-objective Pareto front + FastAPI deployment
  - Auto re-training with drift detection

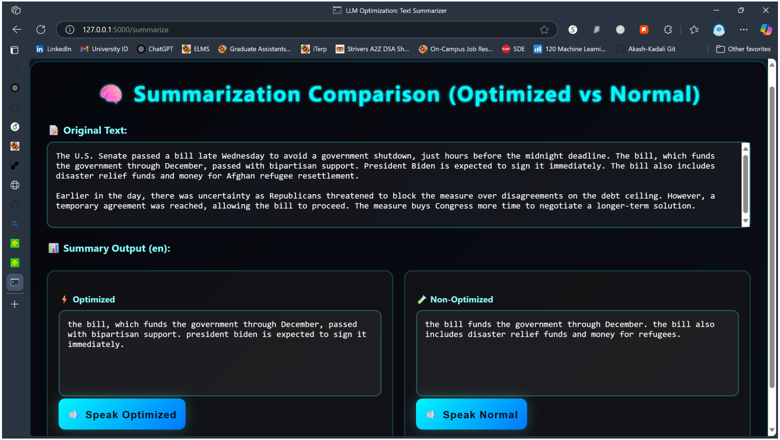
Live Demo

Optimization in Action



Live trial using Optuna + LoRA on T5-small

Summary Output for T5-small on CNN/DailyMail Input



Optimized summary captures political context and intent clearly. Non-optimized output misses key information and feels repetitive.

Final Note

Tuning LLMs isn't about brute force—it's about smart strategy.

Thank you!

Special thanks to **Prof. Richard J. La** and **Amogha Sunil**
for their invaluable guidance throughout this project.