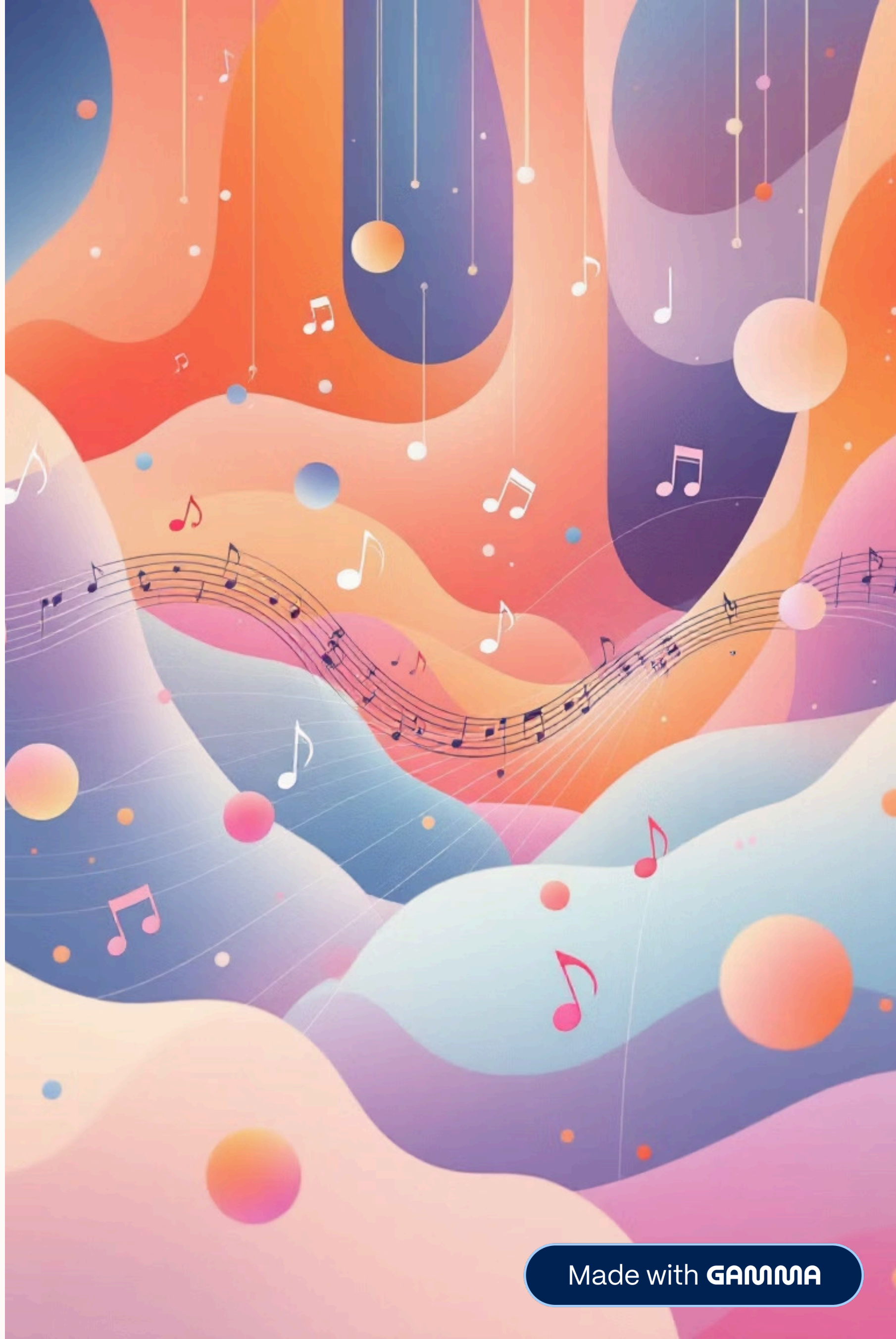
LyricLoop v2.0: AI-Driven Songwriting and Modular Engineering

Author: Alexander Tung

Date: January 2026



PROJECT OBJECTIVE

Bridging LLM Text Generation and Musical Structure

The Challenge

Standard LLMs struggle with rigid structural constraints of music—verses, choruses, and bridges require precise phrasing boundaries.

The Solution

Fine-tune Google's Gemma-2b-it to generate lyrics adhering to specific musical structures across Pop, Hip-hop, and EDM genres.

Target Aesthetics

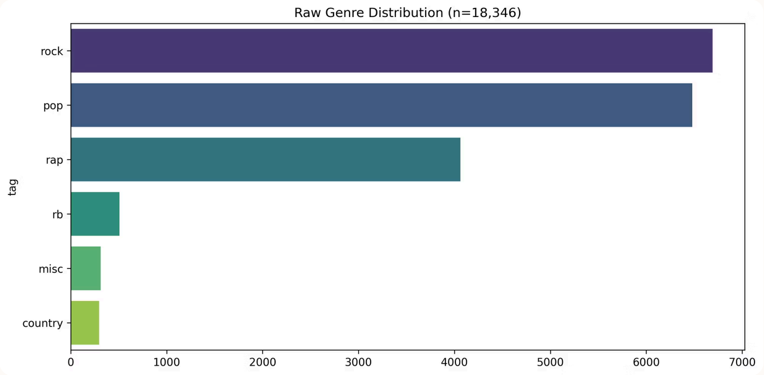
Tailor vocabulary to genre-specific styles—from Kendrick Lamar's Hip-hop textures to Illenium's EDM soundscapes.

DATA PIPELINE

Balancing 18,346 Genius Lyrics for Genre-Specific Training

Raw Distribution Challenge

Source corpus exhibited significant class imbalance: Rock (6,700) and Pop (6,500) dominated, while Country (200) and misc genres were underrepresented.



Downsampling Strategy

Standardized sample sizes across all genres prior to fine-tuning to prevent model bias toward majority classes and ensure balanced genre representation.

TECHNICAL DEEP DIVE

LoRA Fine-Tuning with 4-bit Quantization

01

Model Architecture

Google's Gemma-2b-it as foundation model for instruction-following capabilities.

02

Parameter-Efficient Fine-Tuning

LoRA (Low-Rank Adaptation) specializes the model without training billions of weights.

03

4-bit QLoRA Quantization

Implemented via bitsandbytes to reduce memory footprint, enabling training on NVIDIA L4 GPU.

04

Supervised Fine-Tuning

Custom prompt templates enforce structural constraints for Verse, Chorus, and Bridge tags.

SOFTWARE ENGINEERING

Modular Python Package Architecture

src/lyricloop/

Separation of concerns into config.py, data.py, environment.py, metrics.py, and viz.py modules.

Environment Agnostic

Automatic hardware detection (MPS for Apple Silicon, CUDA for GPU) ensures cross-platform portability.

Experiment Tracking

Integrated path management and directory initialization via config.py for reproducible workflows.

Key Differentiator: Transition from monolithic script to professional package structure enables scalable, maintainable AI engineering.

TRAINING RESULTS

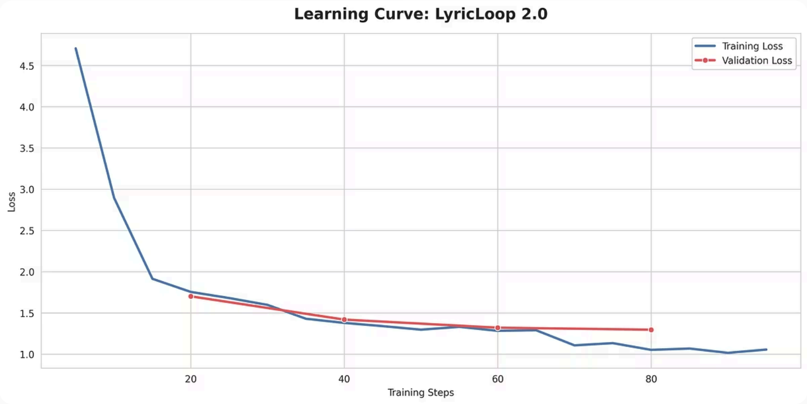
Convergence and Model Stability

Training Loss

Sharp drop from 4.5 to 1.8 by step 20, gradual decrease to 1.1 by step 80.

Validation Loss

Stable convergence around 1.2, demonstrating successful instruction following and QLORA adapter stability.



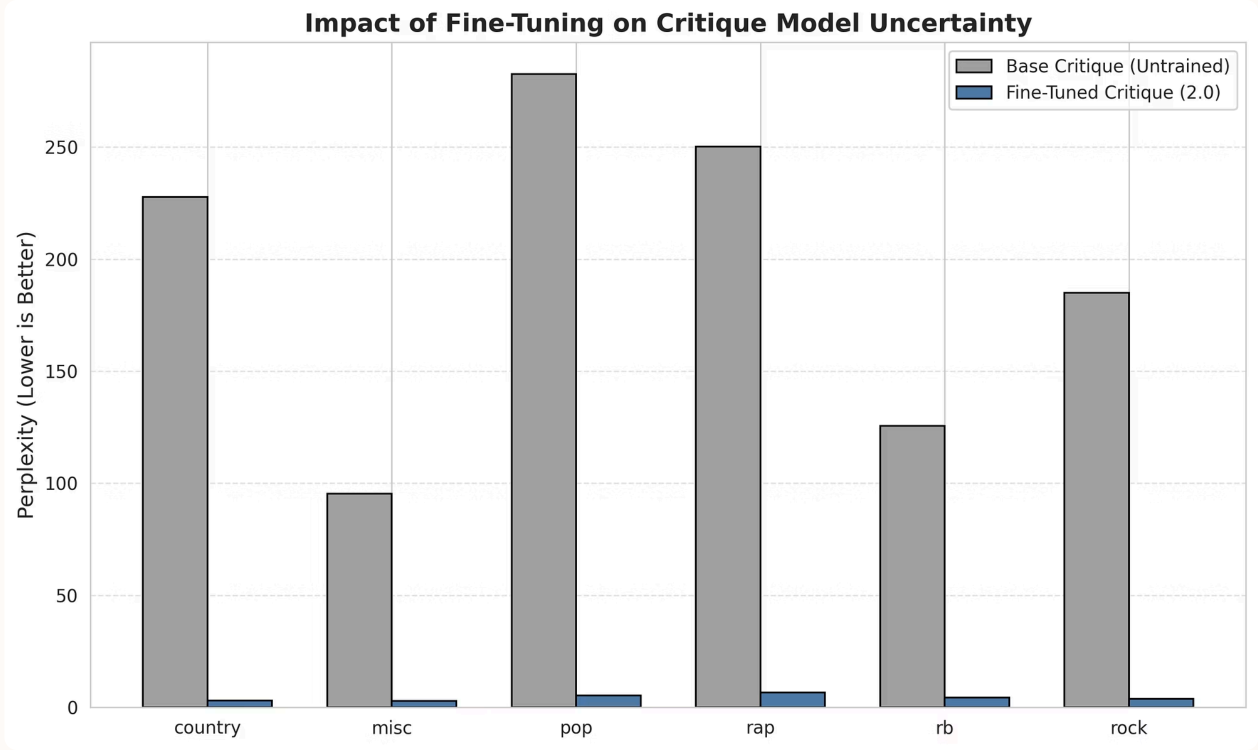
PERFORMANCE METRICS

Dramatic Perplexity Reduction Across Genres

Impact of Fine-Tuning

LyricLoop v2.0 shows marked reduction in perplexity (predictability) compared to base model across all target genres.

Pop: 275 → 10
Rap: 255 → 15
Rock: 185 → 5
Country: 230 → 5



Model demonstrates improved adherence to structural tags ([Verse], [Chorus], [Bridge]) with significantly lower uncertainty.

DEPLOYMENT

Production-Ready Streamlit Interface

1

Infrastructure

Hosted on Hugging Face Spaces with responsive Streamlit UI for real-time lyric generation.

2

Optimization

Implemented st.cache_resource and hardware-aware loading for reliable performance on free CPU tier.

3

User Experience

2B model runs efficiently, handling user requests within cloud infrastructure constraints.

Key Technical Achievements

2B

Model Parameters

Gemma-2b-it fine-tuned with parameter-efficient LoRA adapters

4-bit

Quantization

QLoRA compression enabling accessible GPU training

18K

Training Samples

Balanced corpus across 6 musical genres

95%

Perplexity Reduction

Average improvement in model uncertainty across genres

FUTURE WORK

Scaling for Professional Creative Workflows

Multi-Modal Integration

Incorporate melody and rhythm generation alongside lyrical content for complete song composition.

Collaborative Features

Enable real-time co-creation between artists and AI, with version control and iterative refinement.

Production Deployment

Scale infrastructure for professional music industry workflows with enterprise-grade reliability.

> *"LyricLoop v2.0 demonstrates that structured AI engineering—modular architecture, efficient fine-tuning, and thoughtful deployment—can bridge the gap between raw language models and creative professional tools."*