Benyamin Gidanian

Los Angeles, CA · (310) 613-3655 · bgidanian@gmail.com
GitHub · LinkedIn · Live demo · Hugging Face

SUMMARY

Applied ML engineer who takes models the whole way: dataset audit, fine-tuning, honest evaluation, and deployment as a live service under CI/CD. Work spans NLP (transformer fine-tuning, token classification), computer vision (transfer learning under class imbalance), and embedding-based retrieval at 500k+ document scale. Six trained checkpoints published to Hugging Face Hub with documented per-class metrics and latency benchmarks. Every number below is measured on held-out data, and where an earlier number was not, the correction is documented in the repository.

TECHNICAL SKILLS

ML & Deep Learning: PyTorch, PyTorch Lightning, TensorFlow, Keras, Hugging Face Transformers, DeBERTa, BERT, ViT, ConvNeXt, EfficientNet, timm, CUDA, bf16 mixed precision
Modeling Techniques: Transfer learning, token classification, two-phase fine-tuning, class-imbalance handling (weighted loss, weighted sampling), hyperparameter search, transformer embeddings, hybrid retrieval (BM25 + vector), deep Q-learning
Evaluation: Macro and per-class F1, ROC-AUC, precision and recall, held-out and leakage-safe splitting, latency benchmarking, MLflow, Weights & Biases
ML Engineering: FastAPI model serving, Docker, GitHub Actions CI/CD, Hugging Face Hub, Weaviate, model cards, Pytest, Ruff, Pyright, uv
Cloud: AWS (ECS Fargate, ECR, IAM, SSM Parameter Store), Cloudflare (DNS/TLS), Linux (Arch), Bash, Git
Data: NumPy, Pandas, dataset auditing, feature engineering, preprocessing pipelines
Languages: Python, SQL, JavaScript, C
Agentic Development: Claude Code as the implementer on four shipped applications, directed through written specs, per-repo instruction files recording architecture and invariants, and required tests, with every change reviewed and verified; on the ML projects, used for design review and tradeoff analysis while I wrote the code

EXPERIENCE

Independent Machine Learning Engineering, Self-Directed

Los Angeles, CA · March 2025 to Present

- Worked full time on self-directed ML engineering: sourced and audited datasets, trained and evaluated models, and deployed them as maintained production services, owning scope and delivery on each project end to end.
- Published six model checkpoints to Hugging Face Hub with full model cards documenting per-entity and per-class metrics, GPU latency benchmarks, and distribution-shift behavior.
- Shipped four full applications by directing Claude Code, including a mobile app with an AI coaching service and MCP server, and a multi-tenant invoicing and inventory platform.

One-on-One Talmud Tutor

Mesivta Bircas Yitzchok · Sept 2023 to March 2025

- Delivered individualized instruction over an 18-month engagement, building structured curricula to accelerate comprehension and long-term retention.

PROJECTS

PII Redaction: Fine-Tuned NER in Production

PyTorch · DeBERTa-v3 · Hugging Face Trainer/Hub · FastAPI · Docker · AWS ECS Fargate
Code · Live demo · base · small · xsmall

- Fine-tuned three DeBERTa-v3 models on `ai4privacy/pii-masking-300k` for named entity recognition across 27 PII entity types (55 BIO labels), reaching macro F1 of **0.9557 (base)**, **0.9517 (small)**, and **0.9424 (xsmall)** on the held-out test set, a size and accuracy ladder letting a deployment trade 1.3 points of F1 for a materially smaller model.
- Built a custom `WeightedTokenClassificationTrainer` with class-balanced cross-entropy to counter severe `O`-token imbalance, training in two phases (frozen backbone, then full fine-tune) with bf16 mixed precision and W&B logging of per-entity F1 alongside interactive HTML prediction tables.
- Designed inference-time chunking for documents exceeding the context window: overlapping token windows with paragraph- and sentence-aware boundary alignment, resolving overlap-region predictions by maximum confidence so entities spanning a chunk boundary are neither split nor double-counted.
- Published all three checkpoints with model cards documenting per-model F1, GPU latency, per-entity tradeoffs, and measured behavior under distribution shift, including the name entities (`GIVENNAME`, `LASTNAME`) that score 0.10 to 0.20 below macro F1 on sparse training support.
- Served the models from an authenticated FastAPI application with typed Pydantic schemas and per-IP rate limiting, deployed to AWS ECS Fargate with Cloudflare TLS and a GitHub Actions pipeline shipping every merge to `main` into production.

Image Classification: 11-Architecture Comparison and a Corrected Evaluation

PyTorch Lightning · MLflow · ViT · ConvNeXt · EfficientNet · FastAPI · Docker
Code · Models

- Trained and compared 11 backbone architectures across 71 MLflow-tracked runs on the 102-class Oxford Flowers dataset, reaching **0.9837 accuracy / 0.9722 macro F1** with ViT-B/16 and selecting ConvNeXt-Tiny (0.9821 / 0.9694) as the serving default, a third of the weights for 0.3 points of macro F1.
- Caught and fixed a train/validation/test leak that had inflated every metric in the project to near 1.0: the data module constructed all three splits from the training subset, so validation and test were scoring the model against its own training images. Re-evaluated the published models on honest splits, corrected the published model cards, and added a regression test asserting the three index sets are pairwise disjoint.
- Migrated a hand-rolled training loop to PyTorch Lightning with a registry-driven config, making backbone, optimizer, and scheduler choices data rather than code, and implemented staged backbone unfreezing as a callback with a separate parameter group at a lower learning rate.
- Reported macro F1 as the primary metric rather than accuracy, since class counts range from 40 to 258 images and the gap between the two is where the rare classes are failing.

Recipe Genome: Hybrid Retrieval over 522k Documents *(in progress)*

Weaviate · BGE embeddings · FastAPI · Docker Compose
Code

- Built an offline pipeline parsing, cleaning, and indexing **522,517 recipes** into Weaviate, embedding through a GPU-backed `text2vec-transformers` sidecar (BAAI/bge-small-en-v1.5) so index-time and query-time vectors come from one model and cannot drift apart.
- Audited the source dataset and found roughly 196,000 records sharing a single identical boilerplate description; blanked it per row at insert time rather than dropping the field or the rows, so template text could not collapse unrelated documents together in embedding space.
- Diagnosed a silent indexing failure in which a capitalization mismatch between the declared schema and the property mapping caused all 522k objects to be stored with no vectors while reporting zero errors; added a startup assertion verifying schema, mapping, and vectorized fields agree before a load may begin.
- Starting from hybrid BM25 + vector search and holding a reranker back until a retrieval eval harness (recall@10, MRR against held-out queries) demonstrates the current configuration falls short. The inherited `alpha` default is treated as unmeasured, not as a choice.

Gravitational Lens Finding: ML4SCI DeepLense (GSoC 2026 Evaluation)

PyTorch · ConvNeXt-Tiny · EfficientNet-B0 · timm
Code

- Reached **0.9887 AUROC** on binary gravitational lens classification under 16.6:1 class imbalance with ConvNeXt-Tiny, and **0.98 macro AUROC** on 3-class substructure classification via two-phase fine-tuning of EfficientNet-B0.
- Addressed the imbalance with combined `WeightedRandomSampler` oversampling and calibrated `BCEWithLogitsLoss` positive weighting.

Agent-Built Applications (Claude Code)

Built by directing Claude Code: I set the specs, architecture, invariants and test requirements, and reviewed and verified each change. Claude wrote the code.

Hybete: Hybrid Running and Lifting App *(Flutter · FastAPI · Supabase · Claude API · MCP · Google Cloud Run)*

- iOS and Android app for GPS run tracking, gym logging and multi-week training plans, with a local-first store and optional Supabase sync isolated per user by row-level security.
- AI coach served from a FastAPI service on Cloud Run: Claude Sonnet with a read-and-propose tool layer, prompt caching, a per-user monthly spending cap and RevenueCat subscription gating. The model never writes directly; proposals land in an inbox the user accepts or dismisses.
- MCP server so users can connect their own AI assistant, secured with OAuth 2.1 and dynamic client registration, with tokens that row-level security refuses so outside clients reach data only through the tools.
- 170+ tests across app and server, including a check that fails when the server's copy of the app's plan format goes stale.

Invoicer: Multi-Tenant B2B Invoicing and Inventory *(Django · PostgreSQL · htmx · Alpine.js · WeasyPrint · Playwright)*

- Multi-tenant SaaS for small wholesalers covering the full buy and sell loop: purchase orders, receiving, vendor bills, quotes, sales orders with backorders, invoices, credit notes, split payments, and AR/AP aging with statement PDFs.
- FIFO inventory costing on an append-only stock ledger, using row locking against numbering and stock races. Voids and returns replay the original stock movements, so stock comes back to the lot it left at its original cost and never twice.
- Shaped the requirements by recovering a 1990s desktop ERP and six years of a real wholesaler's data (186k documents, 2,974 customers), mapping 1,588 legacy fields to separate deliberate gaps from oversights.
- 327 tests, including Playwright browser tests and a production-settings check.

Mise: Schema-Validated Recipe App *(Flutter · JSON Schema)* · Code

- Stores LLM-generated recipes as JSON validated against a JSON Schema before parsing. The import prompt embeds the live schema so model output can't drift from what the validator accepts.

Baker's Calculator *(Flutter · Dart · Riverpod)* · Code

- Rewrote my own hand-built bread calculator with Claude Code around a pure-Dart domain layer for baker's percentages, inverse flour solving for sourdough, scaling, and backward-scheduled bake timelines. Every output of my original version is kept as a golden regression test.

Fundamentals & Algorithms

- **ML Fundamentals Library** *(Python, NumPy)*: linear and logistic regression, k-means, decision trees, Gaussian anomaly detection, and a multi-layer neural network with hand-written backpropagation, implemented from first principles with no scikit-learn, TensorFlow, or PyTorch, and an API reference generated from the source tree. Code
- **Deep Q-Network for Lunar Landing** *(TensorFlow, Keras, Gymnasium)*: DQN agent with experience replay and soft-update target networks, solving LunarLander-v3 above a 200 rolling-average score. Code
- **Crossword Generator** *(Python)*: constraint satisfaction solver using AC-3 arc consistency with MRV and least-constraining-value heuristics, cutting backtracking search space by over 80%. Code
- **Network** *(Django, React, SQLite)*: full-stack social platform with authentication, a follow graph, and asynchronous likes and inline editing. Code

EDUCATION & COURSEWORK

Stanford University & DeepLearning.AI (Coursera), Machine Learning Specialization
Supervised learning, neural networks, decision trees, unsupervised learning, recommender systems

DeepLearning.AI, PyTorch for Deep Learning (Professional Certificate)
Neural networks and tensors, CNNs, transfer learning and fine-tuning, NLP with transformers and Hugging Face, advanced architectures (ResNet, DenseNet, Siamese networks, multi-head attention), model interpretability, ONNX export, pruning and quantization, experiment tracking

Harvard University (edX), CS50 Suite
CS50x (C, Python, SQL, data structures and algorithms) · CS50p (OOP, testing) · CS50ai (search, logic, optimization, constraint satisfaction) · CS50w (Django, React, CI/CD)