Bayesian Optimization for Catalyst Discovery

Project Proposal

Erkin Gönültaş

28 September 2025

Aim

The aim of this project is to develop a machine learning–driven framework that accelerates the discovery of heterogeneous catalysts by applying Bayesian Optimization (BO) to multi-objective materials datasets. The project will demonstrate how probabilistic surrogate models and Pareto optimization can identify promising catalyst candidates with optimal trade-offs between activity and stability, given limited experimental or computational budgets.

Problem Statement

Catalyst design is inherently a multi-objective challenge:

• Activity (reaction rate) must be maximized to ensure efficient conversion.

• Stability (lifetime under reaction conditions) must also be maximized for practical deployment.

• Often, these objectives conflict — highly active catalysts may be unstable, while highly stable catalysts may be sluggish.

Traditional discovery approaches rely on high-throughput experiments or density functional theory (DFT) calculations, both of which are expensive and time-consuming. As a result, brute-force exploration of the vast chemical design space (millions of possible compositions, facets, adsorbates, supports) is infeasible.

Thus, there is a pressing need for data-efficient strategies that can guide the search toward the most informative experiments and help reveal the Pareto frontier of optimal trade-offs.

Proposed Solution

We propose to apply Bayesian Optimization with Gaussian Process (GP) surrogates to efficiently explore catalyst design spaces.

Key aspects of the solution:

• Design space (X): Catalyst composition (binary/ternary alloys), surface facet, adsorbate type.

• Objectives (Y):

• Activity proxy → adsorption or reaction energies (e.g., H adsorption free energy).

• Stability proxy → formation energy (from Materials Project or OQMD).

• Constraints: Feasibility filters (e.g., energy above hull < threshold).

• Optimization strategy: Multi-objective BO with q-Expected Hypervolume Improvement (qEHVI) to discover Pareto-optimal candidates efficiently.

• Outcome: A ranked set of candidate catalysts on the Pareto front, visualized with interactive dashboards and validated against baselines (random, grid).

This framework simulates how an AI-driven research assistant could accelerate materials discovery by prioritizing only the most promising candidates for expensive validation.

Core Functional Stack

1. **Data Layer**

• Sources:

• Materials Project: formation energies (stability)

• Catalysis-Hub: adsorption/reaction energies (activity)

• (Optional) Open Catalyst Project, OQMD, NOMAD for supplementary data

• Processing:

• Normalize units and conventions (eV, adsorption sign).

• Map compositions between Catalysis-Hub and MP.

• Generate composition-based features (mean Z, EN, radii, etc.).

• Output tidy dataset: catalyst_id, composition, facet, adsorbate, activity_score, stability_score, feasible_flag.

2. **Modeling Layer**

• Surrogate models:

• Multi-output Gaussian Processes (BoTorch/GPyTorch).

• Input transformations: normalization/standardization.

• Outcome transformations: scaling activity/stability to maximize.

• Acquisition:

• Multi-objective qEHVI (hypervolume improvement).

• Constraint-aware selection (feasibility filters).

3. **Optimization Loop**

1. Fit GP models on seed data.
2. Optimize acquisition to propose new candidates.
3. Query oracle (dataset values).
4. Update dataset, refit, iterate.

5. Monitor hypervolume improvement and Pareto set.

4. Visualization Layer

• Pareto front plots (activity vs stability).

• Optimization traces (hypervolume vs iterations).

• Candidate trajectory (points added iteration by iteration).

• Interactive dashboards (Streamlit) for composition sliders, Pareto updates, candidate inspection.

Scenarios

1. Accelerating Catalyst Discovery in Research & Industry

A research group in an industrial R&D lab is tasked with finding catalysts for hydrogen evolution reaction (HER). The challenge: there are hundreds of possible binary and ternary alloy compositions, each requiring costly DFT simulations or experiments to evaluate.

• Without guidance, screening even 5% of the space would take months.

• With Bayesian Optimization:

• Start from a small seed set of ~20 candidates.

• Iteratively select the most informative next experiments using qEHVI.

• Within 50–100 evaluations, the algorithm identifies near-optimal trade-offs between activity (ΔG_H^* close to zero) and stability (low formation energy).

• Researchers then focus lab validation only on Pareto candidates, saving time and cost.

This scenario highlights the general utility: turning a brute-force search into a data-efficient exploration, guiding both academic and industrial discovery pipelines.

2. Binary Ni–Cu Alloys for Hydrogen Adsorption

For this project demo, we focus on binary nickel–copper alloys on the (111) facet as a case study.

- Design space:
    - Variable Ni–Cu atomic ratios ($Ni_x Cu_{(1-x)}$, $x \in [0,1]$).
    - Fixed surface facet: (111).
    - Adsorbate: hydrogen (*H).
    - Objectives:
    - Activity proxy: ΔG_{H^*} adsorption energy (from Catalysis-Hub).
    - Stability proxy: bulk formation energy per atom (from Materials Project).
- Constraint: Only consider alloys with formation energy < 0.1 eV/atom above hull.
- Optimization workflow:
    1. Start with 10 seed points sampled via Latin Hypercube.
    2. Train Gaussian Process surrogates on activity + stability.
    3. Use qEHVI to propose 5 new alloy compositions.
    4. Evaluate true values from dataset, update model.
    5. Repeat for 10 iterations.
- Outcome:
    - Pareto front showing Ni-rich alloys with better stability, Cu-rich alloys with stronger adsorption, and intermediate compositions that balance both.
    - Visualization of how the Pareto set grows over iterations.

This specific scenario gives the project a tangible, reproducible example that ties together data sourcing, modeling, and visualization — while still being extensible to ternary alloys or other adsorbates in future versions.

Technology Stack

Data & sourcing

• Python 3.11 — modern typing/async, wide lib support.

• pandas + numpy + pyarrow — fast, columnar I/O (.parquet) and reliable dataframe ops for ML-ready tables.

• pymatgen + mp-api (Materials Project) — canonical toolkit to query MP and parse structures/thermo; MP's official client is stable and well-documented.

• Catalysis-Hub client (CatHub) or REST — programmatic access to adsorption/reaction energetics; exactly what we need for activity labels.

• matminer — battle-tested materials featurization (composition stats, elemental properties) so we don't reinvent feature engineering.

• (Optional) OC20/OC22 loaders — to pull adsorbate–surface data or learned embeddings if we extend beyond v1.

Feature engineering

• matminer featurizers (Composition, ElementProperty) — rich, interpretable descriptors (mean EN, atomic radius, etc.) that work well with small data and GPs.

• scikit-learn preprocessing — StandardScaler, OneHotEncoder for facet/site/support when included; keeps the pipeline transparent and reproducible.

Modeling & Bayesian optimization

• BoTorch + GPyTorch — state-of-the-art Gaussian Process stack with first-class support for multi-objective (qEHVI), batching, constraints, and GPU acceleration.

• GP kernels — Matern/RBF with ARD lengthscales (interpretability: which features matter); outcome/input transforms for stable training.

• Baselines — scikit-learn (RandomForest, GradientBoosting) and XGBoost/LightGBM to compare against BO (important for showing value add).

• Acquisition — qEHVI for Pareto hypervolume; optional feasibility constraint for hull-based filters; supports q-batching for realistic lab batches.

Optimization loop & experiment control

• Typer (CLI) — simple, modern CLIs (bo run, bo plot) with type hints.

• Hydra or Pydantic-settings — clean config management for sweeps (kernels, batch size, constraints) without code edits.

• MLflow (local) — lightweight experiment tracking (params, metrics, artifacts) so results are auditable and figures auto-logged.

Visualization & reporting

• matplotlib + plotly — static figs for the paper/README and interactive plots for exploration.

• python-ternary — crisp ternary composition maps when we add ternaries.

• Altair (optional) — declarative charts for fast faceting/linked views.

• Streamlit — a 90-second demo app: sliders for composition bounds, toggles for constraints, live Pareto/trace plots. Great portfolio moment.

• SHAP (optional) — model attribution for tree baselines; for GPs we'll report ARD lengthscales + partial dependence for interpretability.

Data quality, testing, and CI

• pytest + hypothesis — unit + property tests (e.g., score monotonicity, acquisition invariants).

• pydantic — schema validation for dataset rows; prevents silent unit/ sign mistakes on adsorption energies.

• pre-commit: ruff + black + isort — fast lint/format; keeps the repo clean.

• GitHub Actions — run tests/lint on pushes; build docs; smoke-test the Streamlit app headlessly.

Reproducibility & packaging

• uv or poetry — locked environments; reproducible installs.

• conda-env.yml (alt path) — handy for folks in materials ecosystems on HPC.

• jupytext — pair notebooks with .py so diffs are reviewable; notebooks remain lightweight.

• LICENSE + DATA_SOURCES.md — clarify usage rights (MP, Catalysis-Hub) and proper attribution.

Compute & acceleration

• CUDA-enabled PyTorch (optional) — GP training and MC acquisition sampling benefit from GPU if available; falls back to CPU gracefully.

• HPC-friendly — everything runs offline on files; easy to port to clusters without network access.

Stretch / future

• BoTorch cost-aware BO — integrate metal price or simulation time into acquisition.

• Multi-fidelity BO — cheap vs. expensive labels (e.g., different DFT functionals).

• Prefect (lightweight) — if we need orchestration for larger ETL/experimentation runs.

• Weights & Biases — if you want cloud experiment dashboards and sharing.

Project Roadmap

Phase 0 — Scope lock & scaffolding (Day 0–1)

Goal: Stand up a clean, reproducible repo with the agreed problem framing.

Tasks

• Finalize target demo: Ni–Cu (111) + H*; activity = ΔG_H*; stability = formation energy (or E_hull).

• Initialize repo: materials-bo/ with /src, /data, /notebooks, /results, /app.

• Environment: uv/poetry (or conda) lockfile; Python 3.11; pin BoTorch/GPyTorch versions.

• Docs: drop in the proposal (aim, problem, solution, scenarios, tech stack), DATA_SOURCES.md, LICENSE.

• DevX: pre-commit (ruff, black, isort), pytest skeleton, mlflow local tracking.

Deliverables

• Repo skeleton + env lock + CI (lint/test) passing.

• README.md with project summary and "quick start".

DoD (Definition of Done)

• Fresh clone + uv/poetry install works.

• pytest -q green; pre-commit run --all-files clean.

Phase 1 — Data acquisition & curation (Day 2–4)

Goal: Build one tidy, joinable table with activity & stability labels.

Tasks

• Catalysis-Hub pull: adsorption or reaction energies for H* on Ni–Cu (111); keep fields for facet/site, calculation notes.

• Materials Project pull: formation energy (or energy above hull), crystal info for Ni–Cu bulk.

• Reconciliation: map compositions (normalize formulas), align units (eV), confirm sign conventions, deduplicate by (composition, facet, adsorbate).

• Feasibility: compute feasible = (E_hull <= 0.1–0.2 eV/atom); keep both raw and derived.

• Schema (CSV/Parquet):

catalyst_id, composition, x_Ni, x_Cu, facet, site, adsorbate,
E_ads_eV, formation_energy_eV_per_atom, energy_above_hull_eV, feasible

Deliverables

• /data/raw_catalysis_hub.csv, /data/raw_mp_thermo.csv, /data/processed.parquet.

• src/data/fetch_ch.py, src/data/fetch_mp.py, src/data/join_clean.py.

• notebooks/01_data_inspect.ipynb (distributions, missingness, duplicates).

DoD

• ≥300 unique rows after cleaning; no missing values in target columns; unit tests for sign/units pass.

Phase 2 — Feature engineering & splits (Day 4–5)

Goal: Create compact, interpretable features suitable for GPs.

Tasks

• Composition features via matminer (ElementProperty: mean/avg dev of Z, EN, covalent radius, valence e^-, etc.).

• Encode categories (facet/site) if retained; one-hot minimal.

• Scale inputs (0–1) and targets (standardize).

• Splits:

  • Seed set: space-filling (Latin Hypercube) 10–20 points.

• Pool: remaining candidates for simulated evaluations.

• Hold-out: small sanity set (optional).

Deliverables

• /data/processed_with_features.parquet.

• src/features/featurize.py, transformer serialized (/artifacts/transformers.joblib).

• notebooks/02_feature_checks.ipynb (correlations, variance, leakage check).

DoD

• No NaNs after transforms; feature ranges [0,1]; data card with column descriptions.

Phase 3 — Baselines & evaluation harness (Day 5–6)

Goal: Establish simple, transparent baselines and metrics.

Tasks

• Random / LHS search: simulate N evaluations, track hypervolume (HV) over iterations.

• Model baselines: RF/GBM regressors per objective → greedy selection (pick top-k predicted HV increment).

• Metrics: HV, Pareto set size, best-seen per objective; 10 seeds for variability.

Deliverables

• src/baselines/random_search.py, src/baselines/tree_greedy.py.

• results/baseline_hv_traces.csv + plots.

• notebooks/03_baselines.ipynb.

DoD

• Plots show stable median HV curves; runs are deterministic under fixed seeds.

Phase 4 — BO core (qEHVI) with constraints (Day 6–8)

Goal: Implement the multi-objective BO loop using BoTorch/GPyTorch.

Tasks

- Surrogates: two SingleTaskGPs (activity, stability) with Normalize/ Standardize transforms; ARD lengthscales.
- Acquisition: qEHVI with feasibility constraint using energy_above_hull threshold.
- Loop: batch size q=5; T=10–15 iterations; evaluate from the pool (oracle = dataset).
- Tracking: MLflow log params, HV, selected candidates, and artifacts.

Deliverables

- src/bo/models.py, src/bo/acq.py, src/bo/runner.py.
- CLI:
    - bo run --iters 12 --batch 5 --seed 2025
    - bo plot --runs latest
    - results/hv_trace.csv, results/pareto_points.csv, figures saved.

DoD

- BO ≥ 2× median HV gain vs random at equal budget (target; tune if needed).
- Scripts run end-to-end from clean checkout.

Phase 5 — Visualization & interpretability (Day 8–9)

Goal: Communicate results clearly and interactively.

Tasks

- Pareto plots (activity vs stability) with Pareto hull; color by composition fraction.
- Optimization traces: HV vs iteration; best-seen per objective.
- Attribution: report GP ARD lengthscales (feature importance) + simple partial dependence for top features.

Deliverables

• src/viz/pareto.py, src/viz/traces.py, src/viz/ternary.py (optional).

• results/figures/pareto_final.png, hv_vs_iter.png, feature_importance.png.

• notebooks/04_visuals.ipynb.

DoD

• Figures reproducible; captions explain takeaways in ≤3 bullets each.

Phase 6 — Streamlit demo app (Day 9–10)

Goal: A lightweight, portfolio-ready interactive.

Tasks

• Sliders for composition bounds; toggle feasibility; show live Pareto + candidate table.

• "Add next batch" button to simulate one BO iteration (uses stored posterior/acquisition).

• Download buttons: Pareto CSV, top-N candidates.

Deliverables

• /app/streamlit_dash.py with requirements-app.txt.

• GIF screen-capture for README ("90-second demo").

DoD

• streamlit run app/streamlit_dash.py works locally; README includes demo GIF.

Phase 7 — Documentation, packaging & release (Day 10–11)

Goal: Make it easy for others to run, inspect, and learn.

Tasks

• README: hero figures, 1-paragraph abstract, quick start, results, FAQs.

• How-to: docs/ or README sections for data fetch, featurization, BO run, plotting, app.

• Releases: tag v1.0.0, attach figures; note dataset licenses & citations.

Deliverables

• Polished README + DATA_SOURCES.md + CITATION.cff.

• GitHub Release v1.0.0.

DoD

• Fresh machine can reproduce v1.0.0 results following README only.

Phase 8 — Extensions (optional, Week 2+)

Menu (pick 1–2)

• Ternary alloys (Ni–Cu–Pt): add python-ternary heatmaps, 2D slices.

• Cost-aware BO: include metal price as cost; optimize HV per cost.

• Multi-fidelity BO: cheap (semi-empirical) vs expensive (DFT) labels.

• OC20 features: add learned embeddings for surfaces to boost surrogate accuracy.

DoD

• New figure(s) and ablation showing measurable benefit (e.g., HV↑, stability of rankings).

Risk register & mitigations

• Sparse activity data: widen chemistry (add Ni–Pt, Ni–Pd) or switch to O*/OH* tasks; relax to binary first.

• Sign/units mismatch: pydantic schema + unit tests; assert expected ranges (e.g., $\Delta G\_H^* \in [-1.5, +1.5]$ eV).

• BO underperforms: tune kernels, standardization, batch size; try qNEHVI; increase seed diversity; verify constraint calibration.

• Repro issues: freeze random seeds; pin library versions; ship repro.sh.

Success criteria (acceptance)

• Data: curated dataset with documented provenance; ≥300 rows.

• Performance: BO achieves ≥2× median HV gain vs random at equal budget (or a clearly explained close result).

• Clarity: Pareto figure + HV trace tell the story at a glance.

• Usability: one-command run and an interactive demo.

• Reproducibility: clean checkout can reproduce v1.0.0 figures.