MANTRA — Manifold-Aware Network TRAit Model

A research-grade framework for **geometry-aware gene regulatory modeling** using:

- **EGGFM** (Energy-Guided Geometric Flow Model) to learn a *cell-state manifold* from unperturbed K562 cells
- **GNN-based GRN propagation** for regulator → gene → program → trait inference
- **cNMF program discovery** for gene programs
- **SMR-derived trait readout vectors**
- A systematic **ablation pipeline** evaluating manifold constraints, GRN structure, HVG choices, and embedding choice (HVG expression vs PCA-50, etc.).

This README is designed for new researchers, LLM agents, or collaborators who need a **full conceptual overview** of the system architecture, motivations, inductive biases, and experimental protocol.

---

1. Scientific Context

The central objective of MANTRA is to model how **perturbations to gene regulators propagate through the gene regulatory network (GRN), alter cellular programs, and ultimately impact hematopoietic traits** (MCH, RDW, IRF).

We integrate two main data sources:

1. **GWPS (Regulator → ΔGene expression)**
   Provides causal β-like effects of targeted regulator perturbations on downstream gene expression (ΔE).

2. **SMR RBC trait genetics (Gene → trait effect sizes)**
   Provides trait-weight vectors learned from human genetic burden studies, mapping gene expression shifts to trait shifts.

These together enable a *predictive causal chain*:

> **Regulator → ΔExpression → ΔPrograms → ΔTraits**

The core MANTRA innovation is adding a **manifold constraint** so that predicted ΔE respects the geometry of actual transcriptomic variation learned from **unperturbed** K562 cell states.

---

2. Overall Architecture

2.1 Conceptual pipeline

```
Regulator
   ↓ (GWPS ΔE)
Predicted ΔGene Expression (ΔE)
   ↓ (optional manifold regularization via EGGFM)
Geometry-constrained ΔE
   ↓ (cNMF Wᵀ)
ΔProgram Activity (Δa)
   ↓ (SMR-derived θ^(trait))
ΔTrait Prediction
```

Each layer is **separately testable** and participates in ablation experiments.

---

3. EGGFM: Geometry Learning on Unperturbed K562

EGGFM is trained via **denoising score matching (DSM)** on *unperturbed* (non-targeting control) K562 cells.
It learns a smooth energy function (E(x)) whose gradient (-∇E(x)) acts as a **score field** and induces a **geometry / metric** on the cell-state manifold.

You can train EGGFM either on **HVG expression** (`ad.X` in the HVG-restricted space) or directly in a **precomputed embedding** stored in `ad.obsm` (e.g. `"X_pca"` or `"X_diffmap"`). This allows a clean comparison between:

- EGGFM in raw HVG space, versus
- EGGFM on top of a classical embedding (e.g. PCA-50).

3.1 HVG selection and ablations

We start from a QC'd K562 AnnData with ~3k HVGs annotated. We then subselect smaller HVG panels for EGGFM:

- HVG ∈ {50, 75, 100, 150, 200, 250, 500, 3000}

Empirically:

- Very large HVG counts (>500–3000) yield unstable DSM optimization and noisy geometry.
- Moderately sized HVG panels (≈75–150) provide a good tradeoff between:
  - expressiveness
  - stable DSM loss
  - computational tractability
  - reduced curvature pathologies

We currently focus on **HVG=75** and **HVG=100** as main EGGFM spaces, and compare them via:

- DSM loss curves
- Downstream GRN training behavior (expr vs geo loss)

One of these will be chosen as the **baseline HVG space**; the other remains as an explicit ablation.

Typical DSM hyperparameters (in `configs/params.yml` under `eggfm_model` / `eggfm_train`):

- **σ (DSM noise scale)** ~ 0.15–0.2
- Hidden dimensions: e.g. `[256, 256]` MLP
- Early stopping on DSM loss with patience and min-delta

3.2 EGGFM outputs

Each `train_energy.py` run produces an EGGFM checkpoint:

- `state_dict` (EnergyMLP weights)
- `model_cfg` (hidden dims, etc.)
- `n_genes` (dimensionality of the feature space)
- `var_names` (HVG gene names used for this model)
- `mean`, `std` (standardization stats **in the chosen feature space**)
- `space` (e.g. `"hvg"` or `"X_pca"`)

Examples:

- `out/models/eggfm/eggfm_energy_k562_hvg_hvg75.pt`
- `out/models/eggfm/eggfm_energy_k562_hvg_hvg100.pt`
- `out/models/eggfm/eggfm_energy_k562_X_pca_hvg50.pt` *(EGGFM in PCA-50 space)*

These checkpoints are then used both as:

- **Energy priors** in the GRN loss (geometry term), and
- **Sources of HVG gene lists / feature names** for NPZ construction and cNMF alignment.

---

4. HVG Ablation Summary

> Note: numbers below are indicative / in-progress; final values will be reported in the project report.

HVG Count	Training Stability	DSM Loss (rough)	Interpretability	Notes
3000	fails	~1e5	too noisy	geometry not learned
500	fails early	~1e4	unstable	curvature too high
250	moderate	~3.8e3	decent	borderline
200	good	~2.9e3	strong	viable
150	very good	~2.0e3	excellent	strong candidate

HVG Count	Training Stability	DSM Loss (rough)	Interpretability	Notes
100	**very strong**	~1.2e3	**excellent**	good coverage + stability
75	**extremely good**	~8.3e2	very good	slightly lower gene coverage
50	over-simplified	~4.3e2	poor coverage	geometry too compressed

We keep **HVG=75** and **HVG=100** as main baselines, and explicitly compare:

- DSM loss curves
- GRN training loss decomposition (expr vs geo vs prog)
- Downstream ΔTrait performance.

---

5. Dimensionality Reductions & Embedding Ablations

After QC and HVG extraction (3k HVGs), we compute several **manifold baselines** on the HVG subset using `scripts/hvg_embed.py`. These embeddings are stored in `.obsm` of:

- `data/interim/k562_gwps_unperturbed_hvg_embeddings.h5ad`

5.1 Included views in `.obsm`

- `X_hvg_trunc` — raw expression of top-k HVGs (e.g. 150)
- `X_pca` — Scanpy PCA (e.g. 20 or 50 PCs)
- `X_diffmap` — Diffusion Map (20D)
- `X_umap` — UMAP on PCA space (20D)
- `X_phate` — PHATE (optional, if installed)
- `X_isomap` — Isomap (optional, slower)
- `X_spectral` — Laplacian eigenmaps (optional)

5.2 Excluded / future work

- **Dcol PCA** — too slow at this scale for now
- **Palantir** — pseudotime, not a pure embedding

These embeddings serve as **comparison manifolds** for:

- GRN propagation
- Trait prediction
- Geometry quality metrics (ARI, kNN overlap, geodesic distortion)

---

6. GRN Model (Regulator → Gene GNN)

A lightweight GNN (`GRNGNN`) propagates regulator effects through the gene network.

Inputs per sample (from NPZ):

- `reg_idx` — index of the perturbed regulator
- `deltaE` — regulator-level ΔE in HVG space [N, G]
- `deltaP_obs` — observed ΔProgram activity, if W is provided
- `deltaY_obs` — observed ΔTrait (currently stubbed)
- `dose` — dummy 0.0 for K562 (no explicit dose modeling yet)

Architecture:

- Gene embeddings (learned)
- Multi-layer message passing using adjacency `A ∈ ℝ^{G×G}`
- Optional **TraitHead** on top of program space

The GRN model is trained with a composite loss:

- **Expression term** (L_{\text{expr}}): ΔE reconstruction / fitting
- **Geometry term** (L_{\text{geo}}): encourages ΔE predictions to lie in low-energy regions under EGGFM
- **Program term** (L_{\text{prog}}): consistency with ΔP_obs via W
- **Trait term** (L_{\text{trait}}): trait supervision once ΔTrait labels are wired

Ablations test:

1. **Vanilla GRN** (no geometry term)
2. **GRN + EGGFM geometry prior**
3. **GRN + alternate manifolds** (e.g. PCA-50 vs raw HVG)
4. **Geometry-only baselines** (no GRN).

---

7. Trait Mapping (cNMF Programs + SMR)

We run **consensus NMF (cNMF)** on **unperturbed QC cells** to learn program loadings **W**.

We intentionally **align the gene space to the EGGFM HVG space** by using the EGGFM checkpoint's `var_names` and subsetting the QC AnnData before running cNMF. This guarantees:

- Rows of W correspond exactly to the genes and order used by ΔE and the energy prior.

7.1 cNMF outputs

For a given HVG space (e.g. 75 or 100 genes), cNMF produces:

- `W_consensus.npy` — `[G, K]` consensus gene-program loadings
- `programs_all.npy` — stacked program vectors across runs `[R*K, G]`
- `cluster_labels.npy` — cluster assignment per run-program
- optional: `program_counts.npy`, `run_coverage.npy` for stability diagnostics

- `genes.npy`, `cells.npy` — the gene and cell IDs used
- `manifest.yml` — lightweight YAML manifest with shapes, RMSE per run, etc.

We store these under e.g.:

- `out/programs/k562_hvg75_*`
- `out/programs/k562_hvg100_*`

W is used to map ΔE → ΔProgram activity:

[\Delta a = W^\top \Delta E.]

7.2 Trait readout via SMR

Trait vectors **$\theta^{(t)}$** (per trait t ∈ {MCH, RDW, IRF}) are derived from SMR:

- SMR yields **gene-level trait effects**.
- We regress these onto W to obtain trait weight vectors in program space:

[g^{(t)} \approx W \theta^{(t)}.]

Predicted trait deltas:

[\widehat{\Delta \text{Trait}}^{(t)} = \langle \theta^{(t)}, \Delta a \rangle.]

cNMF hyperparameters live in `configs/params.yml` under a dedicated `cnmf` block.

---

8. NPZ Construction (Streaming GWPS → ΔE NPZs)

Because the K562 GWPS file (`data/raw/k562_gwps.h5ad`) is large (~2M cells), we construct **NPZ datasets** via a streaming script:

- `scripts/make_grn_npz.py`

For a given EGGFM checkpoint (which defines the HVG gene list), we:

1. Load the checkpoint and extract `var_names` (HVG genes).
2. Map these genes into `raw.var_names` of `k562_gwps.h5ad`.
3. Apply QC on **per-cell obs** (e.g. `mitopercent`, `UMI_count`).
4. Split cells into **control** (`reg == 'non-targeting'`) vs **perturbed**.
5. Compute a **global control mean** in HVG space.
6. For each regulator r with enough QC'd cells:
    - Average its perturbed cells in HVG space → (x_r)
    - Compute (\Delta E_r = x_r - \bar{x}_{\text{ctrl}})
7. Optionally map ΔE → ΔP_obs via W (if `--cnmf-W` is passed).
8. Stub ΔY_obs for trait deltas (later replaced by real trait labels).
9. Split into train/val and write NPZs:

- `data/interim/grn_k562_gwps_hvg75_npz/{train.npz,val.npz}`
- `data/interim/grn_k562_gwps_hvg100_npz/{train.npz,val.npz}`

Each NPZ contains:

- `reg_idx` — regulator indices (int64)
- `deltaE` — ΔE [N, G] (float32)
- `deltaP_obs` — ΔP_obs [N, K] if W provided, else ΔE
- `deltaY_obs` — ΔTrait stubs [N, T]
- `dose` — dummy (0.0) for K562 (no dose modeling)

9. Ablation Matrix

We evaluate a **2×2 GRN × Manifold grid**, plus HVG and embedding ablations:

```
                         Manifold Constraint?
                    No                         Yes
GRN?
-------------------------------------------------------------
No          Program-only baseline       Manifold-smoothed ΔE
Yes         GRN baseline (ΔE)           GRN + EGGFM (full model)
```

Additional axes:

- **HVG ablations**: 75 vs 100
- **Embedding ablations**: raw HVG vs PCA-50 EGGFM
- **σ (noise) ablations**
- **Model-depth ablations** (GNN layers, hidden dims).

10. Repository Structure

```
data/
  raw/
    k562_gwps.h5ad                    # Full perturbed + control GWPS
(backed)
  interim/
    k562_gwps_unperturbed_qc.h5ad     # QC'd non-targeting controls, 3k
HVGs
    k562_gwps_unperturbed_hvg_embeddings.h5ad
                                      # PCA, DiffMap, UMAP, PHATE, etc.
    grn_k562_gwps_hvg75_npz/
      train.npz
      val.npz
    grn_k562_gwps_hvg100_npz/
      train.npz
      val.npz
out/
  models/
    eggfm/
      eggfm_energy_k562_hvg_hvg75.pt
      eggfm_energy_k562_hvg_hvg100.pt
      eggfm_energy_k562_X_pca_hvg50.pt   # PCA-50 EGGFM ablation
    grn/
      hvg75/
        grn_k562_energy_prior.pt
      hvg100/
        grn_k562_energy_prior.pt
  programs/
    k562_hvg75_W_consensus.npy
    k562_hvg100_W_consensus.npy
configs/
  params.yml
  env.yml
scripts/
  qc.py                   # QC + HVG selection on raw GWPS
  hvg_embed.py            # PCA / DiffMap / UMAP / PHATE on HVGs
  train_energy.py         # EGGFM (HVG or embedding space)
  make_grn_npz.py         # Streaming ΔE NPZ construction
  cnmf_programs.py        # cNMF program discovery (W aligned to EGGFM genes)
  train_grn.py            # GRN GNN training with energy prior
src/
  mantra/
    eggfm/                # EGGFM models, trainer, dataset, inference
    grn/                  # GRN GNN, priors, trainer, dataset
    programs/             # cNMF config + utilities
    embeddings/           # embedding config helpers
    qc/                   # QC/provenance helpers
    utils/                # misc utilities
```

11. Current Status Summary

Completed

- ✔ Full QC + HVG selection (3k HVGs) for unperturbed K562
- ✔ Dimensionality reductions computed (PCA, DiffMap, UMAP, PHATE, etc.)
- ✔ EGGFM training across HVG ablations (50–500, 3000)
- ✔ HVG=75 & HVG=100 identified as leading candidates
- ✔ Streaming NPZ construction (ΔE per regulator) aligned to EGGFM HVG space
- ✔ cNMF program discovery wiring (W aligned with EGGFM HVGs)
- ✔ GRN training loop with energy prior and loss decomposition (expr / geo / prog / trait stubs)
- ✔ Repo packaging, CLI scripts, and config-driven reproducibility

In Progress

- ⟳ Final comparison HVG-75 vs HVG-100 (DSM and GRN metrics)
- ⟳ PCA-50 EGGFM ablations (energy in PCA space)
- ⟳ Wiring real ΔTrait targets and trait loss
- ⟳ Full GRN × Manifold ablation grid

Next Steps

1. Choose baseline HVG (75 vs 100) based on:
    - DSM loss curves
    - GRN loss behavior and generalization
2. Train GRN with:
    - HVG-EGGFM prior
    - PCA-50 EGGFM prior
      and compare ΔTrait performance.
3. Produce figures:
    - DSM loss curves across HVGs
    - Embedding visualizations (PCA, DiffMap, UMAP, EGGFM)
    - Trait prediction scatter plots
    - Performance deltas vs ablations
4. Integrate SMR-derived θ^(trait) and evaluate end-to-end Reg → ΔTrait.

---

12. How to Train Models (Canonical Commands)

12.1 QC + HVG selection (unperturbed controls)

```
python scripts/qc.py \
  --params configs/params.yml \
  --ad data/raw/k562_gwps.h5ad \
  --out data/interim/k562_gwps_unperturbed_qc.h5ad \
  --pet   # restrict to non-targeting controls
```

12.2 Embeddings (PCA-50)

```
python scripts/hvg_embed.py \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --out data/interim/k562_gwps_unperturbed_hvg_embeddings.h5ad \
  --n-components 50 \
  --n-neighbors 30 \
  --seed 7
```

12.3 Train EGGFM (HVG space; main DSM ablations)

```
# HVG = 75
python scripts/train_energy.py \
  --params configs/params.yml \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --out out/models/eggfm \
  --space hvg

# HVG = 100
# (set eggfm_train.max_hvg: 100 in configs/params.yml before this run)
python scripts/train_energy.py \
  --params configs/params.yml \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --out out/models/eggfm \
  --space hvg
```

12.4 Train EGGFM (PCA-50 ablation)

```
python scripts/train_energy.py \
  --params configs/params.yml \
  --ad data/interim/k562_gwps_unperturbed_hvg_embeddings.h5ad \
  --out out/models/eggfm \
  --space X_pca
```

12.5 Build GRN NPZs (ΔE per regulator)

```
# HVG 75 NPZ
python scripts/make_grn_npz.py \
  --ad-raw data/raw/k562_gwps.h5ad \
  --energy-ckpt out/models/eggfm/eggfm_energy_k562_hvg_hvg75.pt \
  --out-dir data/interim/grn_k562_gwps_hvg75_npz \
  --reg-col gene \
  --control-value non-targeting \
  --max-pct-mt 0.2 \
  --min-umi 2000 \
  --min-cells-per-group 10 \
  --val-frac 0.2 \
  --seed 7

# HVG 100 NPZ
python scripts/make_grn_npz.py \
  --ad-raw data/raw/k562_gwps.h5ad \
  --energy-ckpt out/models/eggfm/eggfm_energy_k562_hvg_hvg100.pt \
  --out-dir data/interim/grn_k562_gwps_hvg100_npz \
  --reg-col gene \
  --control-value non-targeting \
  --max-pct-mt 0.2 \
  --min-umi 2000 \
  --min-cells-per-group 10 \
  --val-frac 0.2 \
  --seed 7
```

12.6 Build W (cNMF programs) for HVG-75 and HVG-100

```
# HVG 75: build W aligned to the 75-gene EGGFM / NPZ space
python scripts/cnmf.py \
  --params configs/params.yml \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --energy-ckpt out/models/eggfm/eggfm_energy_k562_hvg_hvg75.pt \
  --out out/programs/k562_hvg75

# HVG 100: build W aligned to the 100-gene EGGFM / NPZ space
python scripts/cnmf.py \
  --params configs/params.yml \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --energy-ckpt out/models/eggfm/eggfm_energy_k562_hvg_hvg100.pt \
  --out out/programs/k562_hvg100
```

12.7 Train GRN (GNN with energy prior)

```
# Example for HVG 100
python scripts/train_grn.py \
  --params configs/params.yml \
  --out out/models/grn/hvg100 \
  --ad data/interim/k562_gwps_unperturbed_qc.h5ad \
  --train-npz data/interim/grn_k562_gwps_hvg100_npz/train.npz \
  --val-npz   data/interim/grn_k562_gwps_hvg100_npz/val.npz \
  --adj       data/interim/A_k562_hvg100.npy \
  --cnmf-W    out/programs/k562_hvg100_W_consensus.npy \
  --energy-ckpt out/models/eggfm/eggfm_energy_k562_hvg_hvg100.pt
```

---

13. Citation Notes

If using this repo in research, please cite:

- GWPS foundational papers
- SMR methods for trait mapping
- cNMF / gene program discovery work
- Any future MANTRA preprint or publication

End of README