PRISM-4D SOTA Enhancement Blueprints

Enhancement 1: Automated ABFE/RBFE Free Energy Perturbation Pipeline

Enhancement 2: Generative Pharmacophore-Conditioned Molecule Design

---

CRITICAL ARCHITECTURAL DECISION: ABFE, NOT RBFE

Before diving into implementation — the original spec says "RBFE for top 5 compounds." **This is wrong for PRISM's use case.** Here's why:

RBFE (Relative Binding Free Energy) requires a **congeneric series** — structurally similar ligands with shared scaffolds where you're comparing R-group modifications. It answers: *"Is analog A better than analog B?"*

ABFE (Absolute Binding Free Energy) works on **diverse scaffolds** with no chemical similarity requirement. It answers: *"How tightly does this molecule bind to this pocket?"*

PRISM detects NOVEL cryptic pockets. The compounds hitting these pockets will NOT be congeneric — they'll come from pharmacophore-guided generation (Enhancement 2) or diverse library screening. There is no existing congeneric series for a pocket nobody knew existed.

Therefore: PRISM needs ABFE first, RBFE second (only after lead optimization begins on confirmed hits).

Factor	RBFE	ABFE
Requires congeneric series	YES	NO
Works on diverse scaffolds	NO	YES
Accuracy (MUE)	~0.9-1.2 kcal/mol	~1.0-2.0 kcal/mol
GPU hours per ligand	~10	~100
Suitable for novel cryptic sites	NO	YES
When to use in PRISM	Lead optimization	Hit validation

References:

- Ross et al., "Large-Scale RBFE Benchmarks," JACS, 2023. FEP+ MUE = 0.9 kcal/mol on congeneric series.
- OpenFE Consortium benchmark (May 2025): 59 systems, 876 ligands, 1200 transformations — RBFE only.
- Cresset FEP Review (Aug 2025): RBFE ~10 GPU-hours/ligand, ABFE ~100 GPU-hours/ligand.
- OneOPES (JCTC 2024): ABFE with enhanced sampling explicitly identifies cryptic binding sites as promising future direction.

ENHANCEMENT 1: AUTOMATED FREE ENERGY PERTURBATION PIPELINE

1.1 Problem Statement

PRISM currently validates detected cryptic pockets using AutoDock Vina docking scores. Vina scores (e.g., -4.1 kcal/mol) are:

- Empirical scoring functions, NOT physics-based free energies
- Accuracy: ~2-3 kcal/mol RMSE at best (often worse)
- Cannot distinguish sub-kcal/mol differences between compounds
- Known to fail on non-standard binding sites (allosteric, PPI interfaces, cryptic)
- No thermodynamic rigor — no entropy, no explicit solvation, no protein flexibility

Required: Replace Vina with alchemical free energy calculations providing ΔG_bind with ±1 kcal/mol accuracy.

1.2 Tool Selection

Primary: OpenFE (Open Free Energy) — MIT License

Why OpenFE over alternatives:

Tool	License	RBFE	ABFE	GPU	Automation	Status
OpenFE	MIT	✓	✓ (v1.8)	Via OpenMM	CLI + Python API	Active, v1.8, 15 pharma partners
FEP+ (Schrödinger)	Commercial ($$$)	✓	✓	✓	Full	Industry gold standard — but closed
GROMACS GPU-FEP	LGPL	✓	✓	800% A100 speedup	Manual	ACS Omega 2025 — GPU-resident
AMBER GPU-TI	Commercial	✓	✓	✓	Semi-auto	pmemdGTI — fast but licensed
FEP-SPell-ABFE	MIT	✗	✓	Via AMBER	Snakemake	ChemRxiv 2024 — ABFE-only, AMBER req'd
OneOPES	GPL	✗	✓	Via GROMACS	Semi-auto	Enhanced sampling, mentions cryptic sites

Decision: OpenFE + GROMACS GPU-FEP backend

- OpenFE provides the automated workflow (network planning, atom mapping, result gathering)
- GROMACS GPU-FEP provides 800% speedup on A100 (scales to RTX 5080)
- Both are open source
- OpenFE v1.8 supports ABFE (experimental) + RBFE (production)
- SepTop protocol for challenging transformations

Fallback: If OpenFE's ABFE module is too immature → FEP-SPell-ABFE (AMBER-based, MIT, automated Snakemake workflow)

References:

- Chen et al., "Acceleration of GROMACS FEP on GPUs," ACS Omega, 2025. DOI: 10.1021/acsomega.5c00151
- OpenFE Blog, "The Free Energy of Everything," May 2025. 59 systems benchmarked.
- Li et al., "FEP-SPell-ABFE," ChemRxiv, 2024. DOI: 10.26434/chemrxiv-2024-tkvrh [PREPRINT — not peer-reviewed]
- Hahn et al., "OneOPES ABFE," JCTC, 2024. DOI: 10.1021/acs.jctc.4c00851

1.3 Physics Basis

Thermodynamic Cycle (ABFE)

```
Protein + Ligand (solvated) ──→ Protein:Ligand complex (solvated)
      │                        │
      │ ΔG_solv(ligand)                │ ΔG_complex
      │                        │
      ▼                        ▼
Protein + Ligand (vacuum)  ──→ Protein:Ligand (vacuum)
```

ΔG_bind = ΔG_complex - ΔG_solv(ligand)

Each leg computed via alchemical transformation: gradually "turn off" ligand interactions (Coulomb annihilation + VdW decoupling) using λ-windows (typically 42 windows for complex, 31 for solvent).

Key equations:

- Free energy difference via MBAR: ΔG = -kT ln⟨exp(-β(U_{λ+1} - U_λ))⟩_λ
- Restraint corrections: Boresch restraints (6 DOF) with analytical correction ΔG_restr
- Charged ligand corrections: Poisson-Boltzmann finite-size correction (Rocklin et al., JCTC, 2013)

Why this matters for cryptic sites specifically:

1. Cryptic pockets are transient — protein must reorganize to expose the site. ABFE captures this reorganization entropy via enhanced sampling (REST2, OneOPES).
2. Water displacement at cryptic sites is often the dominant binding thermodynamic driver. Explicit solvent FEP captures this; Vina does not.
3. PPI interfaces (MYC-MAX, TEAD2-YAP) have shallow, broad pockets. Scoring functions fail here because binding is entropic/desolvation-driven, not dominated by polar contacts.

1.4 Implementation Roadmap

Phase 1: Infrastructure (Week 1-2)

```
scripts/
├── fep/
│   ├── prism_to_openfe.py      # Convert PRISM spike pharmacophore → OpenFE input
│   ├── prepare_abfe.py         # Automated ABFE setup from docking poses
│   ├── prepare_rbfe.py         # RBFE setup for congeneric optimization (Phase 2)
│   ├── run_fep.py              # Launch OpenFE calculations (local GPU or SLURM)
│   ├── analyze_fep.py          # Gather results, compute ΔG, error analysis
│   └── fep_report.py           # Generate publication-quality FEP results
```

Dependencies:

```bash
# Create dedicated conda environment
conda create -n prism-fep python=3.12
conda activate prism-fep

# OpenFE + dependencies
pip install openfe

# GROMACS with GPU-FEP (build from source for GPU support)
# Or use pre-built: conda install -c conda-forge gromacs

# Analysis
pip install alchemlyb arsenic
```

Phase 2: PRISM → OpenFE Bridge (`prism_to_openfe.py`)

This is the critical integration point. PRISM's spike pharmacophore output must drive FEP setup.

Input: PRISM spike JSON + docking results from gpu_dock.py **Output:** OpenFE-compatible protein-ligand systems ready for FEP

```python
"""
prism_to_openfe.py — Bridge PRISM spike pharmacophore to OpenFE ABFE/RBFE

Workflow:
1. Load PRISM spike pharmacophore (BNZ/TYR/CATION/ANION positions + intensities)
2. Load top-N docking poses from gpu_dock.py (UniDock/GNINA output)
3. For each ligand pose:
  a. Verify pose satisfies spike pharmacophore constraints
    (ligand features overlap spike centroids within tolerance)
  b. Prepare OpenFE ChemicalSystem (protein + ligand + solvent)
  c. Configure ABFE protocol with appropriate λ-schedule
4. Output: OpenFE AlchemicalNetwork ready for execution

CRITICAL: Docking box comes from spike cluster envelope (per gpu_dock.py spec).
FEP restraints must be anchored to residues at the PRISM-detected pocket,
NOT to arbitrary protein atoms.
"""
```

Key design decisions:

- Restraint atom selection: Use PRISM lining residues (from clustering output) to select Boresch restraint anchor atoms. This ensures restraints are physically meaningful for the cryptic pocket geometry.
- λ-schedule: Use OpenFE defaults initially (Coulomb annihilation → VdW decoupling), optimize later based on convergence diagnostics.
- Simulation length: 5 ns/λ-window minimum for ABFE (20 ns for challenging systems). Total: 42 windows × 5 ns = 210 ns per ligand.
- Repeats: 3 independent repeats per ligand for error estimation.

Phase 3: Execution Engine (`run_fep.py`)

Local GPU (RTX 5080):

- 1 ligand ABFE: ~6-12 hours on single RTX 5080 (extrapolating from A100 benchmarks)
- Top 5 compounds: ~30-60 hours sequential, ~12-24 hours with 2-ligand parallelism
- GROMACS GPU-FEP eliminates CPU-GPU data transfer bottleneck

Estimated wall-time per target:

Stage	Time (RTX 5080)	Notes
PRISM detection	~2 min	Existing pipeline
GPU docking (top 50)	~10 min	UniDock/GNINA
ABFE setup (top 5)	~5 min	prism_to_openfe.py
ABFE execution (5 ligands)	~30-60 hours	3 repeats each
Analysis + reporting	~10 min	analyze_fep.py
Total	**~1-3 days**	Per target

Scaling note: ABFE is embarrassingly parallel across λ-windows. With cloud burst to 4× GPU instances, total drops to ~8-15 hours.

Phase 4: Analysis & Reporting (`analyze_fep.py`)

Output per compound:

```
Compound: PRISM-KRAS-001
  ΔG_bind = -8.3 ± 0.6 kcal/mol (ABFE, 3 repeats)
  ΔG_bind (Vina) = -4.1 kcal/mol [DEPRECATED — for comparison only]
  Convergence: ✓ (overlap matrix > 0.03 for all adjacent λ-windows)
  Hysteresis: ✓ (forward/reverse within 0.5 kcal/mol)
  Restraint correction: -1.2 kcal/mol (Boresch analytical)
  Charge correction: 0.0 kcal/mol (neutral ligand)
  Spike pharmacophore match: 4/5 features within 2.0 Å
  Classification: NOVEL_HIT (pocket not in UniProt, ΔG < -6 kcal/mol)
```

QC gates (automatic rejection if ANY fail):

- λ-window overlap < 0.03 → insufficient sampling, extend simulation
- Forward/reverse hysteresis > 1.5 kcal/mol → convergence failure
- Protein RMSD > 4.0 Å during FEP → structural instability
- Ligand escapes pocket during any λ-window → restraint failure

Phase 5: RBFE for Lead Optimization (Later)

Once ABFE identifies hits (ΔG < -6 kcal/mol), synthesize initial compounds, get experimental Ki/IC50, then:

```bash
# Generate congeneric series from hit scaffold
openfe plan-rbfe-network -p protein.pdb -M analogs/ -o rbfe_network/

# Run RBFE (much faster — ~10 GPU-hours per ligand)
for json in rbfe_network/*.json; do
  openfe quickrun "$json" -d results/ -o "${json%.json}_result.json"
done

# Gather results
openfe gather results/ --report rbfe_results.tsv
```

1.5 PRISM-Specific Leverage

Why PRISM + FEP is uniquely powerful vs. competitors:

1. **Cryptic pocket = no prior binding data.** Competitors using RBFE need existing SAR. PRISM's ABFE pipeline works from zero prior knowledge — pocket detected blind, compounds generated de novo, binding validated from first principles.
2. **Spike pharmacophore constrains FEP.** PRISM's per-residue spike data tells us WHICH interactions matter. Standard ABFE treats all interactions equally. PRISM can weight restraints toward high-intensity spike positions, potentially improving convergence.
3. **Water network from PRISM MD.** PRISM already runs multi-nanosecond MD with explicit solvent. Water density maps from PRISM trajectories can initialize FEP solvent placement (critical for cryptic sites where water displacement drives binding).
4. **Sub-2-minute detection + 1-3 day validation.** Industry standard: weeks for pocket identification (CryptoSite, fpocket, MD) → weeks for FEP. PRISM collapses identification to minutes, making FEP the bottleneck — which is still 10-100× faster than the total traditional pipeline.

1.6 Risks and Mitigations

Risk	Impact	Mitigation
OpenFE ABFE module immature	Inaccurate results	Fallback: FEP-SPell-ABFE (AMBER) or manual GROMACS setup
RTX 5080 insufficient for 5 ABFE	>3 day wall time	Cloud burst (AWS p4d instances) or reduce to top-3
Cryptic pocket collapses during FEP	Ligand ejected	Enhanced sampling (REST2) + stronger restraints during λ=0
Force field failures (GAFF2/ OpenFF)	Systematic error	Consensus approach (Sage + GAFF + CGenFF) per JCIM 2024 benchmarks
Charged ligands	PB correction errors	Neutralize with counterion or use net-charge-preserving transformations

---

ENHANCEMENT 2: GENERATIVE PHARMACOPHORE-CONDITIONED MOLECULE DESIGN

2.1 Problem Statement

Current PRISM pipeline: detect pocket → extract pharmacophore → search PubChem library.

Limitations of library search:

- PubChem has ~110M compounds — seems large, but drug-like chemical space is ~10^60
- Known compounds optimized for KNOWN targets — unlikely to match NOVEL cryptic pocket geometry
- No IP — screening existing compounds yields known matter
- Pharmacophore feature matching is approximate at 3D level

Required: Generate novel molecules that PERFECTLY complement PRISM's spike pharmacophore features, creating patentable chemical matter purpose-built for each cryptic pocket.

2.2 Tool Selection

Primary: PhoreGen — Nature Computational Science, Aug 2025

PhoreGen (Peng et al., Nat. Comput. Sci., 2025) is the current SOTA for pharmacophore-conditioned 3D molecule generation:

- Diffusion model with asynchronous perturbation on atoms + bonds simultaneously
- Message-passing mechanism incorporating ligand-pharmacophore mapping during denoising
- Trained on ligand-pharmacophore pairs from 3D ligands, crystal complexes, and docked poses
- Generates chemically valid, drug-like, diverse 3D molecules aligned to pharmacophore constraints
- Open source: https://github.com/ppjian19/PhoreGen
- Published: Nature Computational Science (peer-reviewed, Aug 2025)
- Successfully identified new β-lactamase inhibitors in real-world validation

Complementary: PGMG — Nature Communications, 2023

PGMG (Zhu et al., Nat. Commun., 2023) — VAE-based pharmacophore-guided generation:

- 10,000 molecules in 30 seconds on single 2080Ti
- Takes pharmacophore as .posp format (type + 3D position) — directly compatible with PRISM spikes
- Open source: https://github.com/CSUBioGroup/PGMG
- Maximum 8 pharmacophore points per hypothesis
- Generates SMILES (1D) — requires 3D conformer generation downstream

Evaluation: MolSnapper — JCIM, 2025

MolSnapper (Ziv et al., JCIM, 2025) — diffusion conditioning for SBDD via 3D pharmacophores:

- Post-hoc conditioning of pretrained diffusion models
- Generates ~2× more valid molecules than alternatives
- Tested on CrossDocked and Binding MOAD datasets

Also considered:

Tool	Type	Strength	Limitation
MEVO	VQ-VAE + diffusion + evolution	9.6B Enamine REAL training	arXiv July 2025 [PREPRINT]
PharmacoNet	DL pharmacophore modeling	Ultra-large virtual screening	Doesn't generate molecules
PharmacoForge	Automated pharmacophore elucidation	Frontiers Aug 2025	Generates pharmacophores, not molecules
DiffBridge	Diffusion bridge	Pharmacophore-guided	bioRxiv Dec 2024 [PREPRINT]
TargetDiff	Equivariant diffusion	ICLR 2023	Pocket-conditioned, no pharmacophore input

Decision: PhoreGen (primary) + PGMG (high-throughput fallback)

PhoreGen provides the highest quality 3D-aligned molecules. PGMG provides mass generation when you need 10K+ diverse scaffolds quickly for initial screening.

2.3 Physics Basis

Pharmacophore-Conditioned Diffusion

PhoreGen operates on the principle that molecular generation in 3D space can be guided by pharmacophore constraints during the denoising process:

Forward diffusion: Gradually add Gaussian noise to 3D molecular coordinates and atom/bond types:

$$q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$$

Reverse diffusion (conditioned): Denoise while satisfying pharmacophore constraints:

$$p_\theta(x_{t-1} \mid x_t, \Phi) = N(x_{t-1}; \mu_\theta(x_t, t, \Phi), \sigma_t^2 I)$$

where Φ = pharmacophore model (feature types + 3D positions + exclusion spheres)

PhoreGen's innovation: Asynchronous perturbation updates atoms and bonds at different rates during denoising, with a message-passing network that attends to pharmacophore-atom mapping. This ensures generated atoms are correctly positioned relative to required pharmacophore features (aromatic, H-bond donor/acceptor, positive/negative ionizable, hydrophobic).

Why this maps perfectly to PRISM spikes:

PRISM spike types → Standard pharmacophore features:

PRISM Spike	Pharmacophore Feature	PhoreGen Input
BNZ	Aromatic	AR (aromatic ring)
TYR	Aromatic + H-bond	AR + HBD/HBA
TRP	Aromatic (indole)	AR
PHE	Aromatic	AR
CATION (EFP)	Positive ionizable	PI
ANION (EFP)	Negative ionizable	NI
High water_density	H-bond donor/acceptor	HBD/HBA
SS (disulfide)	Hydrophobic	HY
UNK (LIF thermal)	Hydrophobic	HY

PRISM provides **3D coordinates** + **intensity weighting** for each spike. PhoreGen takes **3D pharmacophore coordinates** + **feature types**. The mapping is 1:1.

2.4 Implementation Roadmap

Phase 1: PRISM Spike → Pharmacophore Translator (Week 1)

```
scripts/
├── genphore/
│   ├── spike_to_pharmacophore.py   # PRISM spikes → PhoreGen/PGMG pharmacophore format
│   ├── run_phoregen.py            # Execute PhoreGen generation
│   ├── run_pgmg.py               # Execute PGMG high-throughput generation
│   ├── filter_generated.py        # Drug-likeness, PAINS, SA score filtering
│   ├── rank_molecules.py          # Multi-objective ranking
│   └── genphore_report.py         # Publication-quality output
```

`spike_to_pharmacophore.py` **— The critical bridge:**

```python
"""
Convert PRISM spike pharmacophore output to PhoreGen JSON and PGMG .posp formats.

Algorithm:
1. Load spike JSON from PRISM run
2. For each spike type, compute intensity-weighted centroid within detected pocket
3. Filter: only include centroids with total intensity > threshold (noise rejection)
4. Map spike type → pharmacophore feature type
5. Add exclusion spheres from pocket lining residues (prevents generation
   of atoms that would clash with protein)
6. Output:
   - PhoreGen JSON: {features: [{type, x, y, z}], exclusion_spheres: [{x, y, z, r}]}
   - PGMG .posp: one line per feature, "TYPE X Y Z"

CRITICAL:
- Coordinates must be in the SAME reference frame as the protein structure
- Exclusion spheres from protein heavy atoms within 4Å of pocket center
- Maximum 8 features for PGMG (select top-8 by intensity if more exist)
- PhoreGen has no feature count limit
"""
```

PhoreGen input format (JSON):

```json
{
  "features": [
    {"type": "AR", "x": 65.2, "y": 62.1, "z": 40.8, "weight": 0.95},
    {"type": "PI", "x": 67.1, "y": 64.3, "z": 42.0, "weight": 0.72},
    {"type": "NI", "x": 63.8, "y": 61.5, "z": 39.2, "weight": 0.68},
    {"type": "HBA", "x": 66.0, "y": 63.8, "z": 41.5, "weight": 0.81}
  ],
  "exclusion_spheres": [
    {"x": 64.5, "y": 62.0, "z": 41.0, "r": 1.5},
    ...
  ]
}
```

PGMG input format (.posp):

```
AR 65.2 62.1 40.8
PI 67.1 64.3 42.0
NI 63.8 61.5 39.2
HBA 66.0 63.8 41.5
```

Phase 2: PhoreGen Integration (Week 2)

Installation:

```bash
git clone https://github.com/ppjian19/PhoreGen.git ~/tools/PhoreGen
cd ~/tools/PhoreGen
conda create -n phoregen python=3.10
conda activate phoregen
pip install -r requirements.txt
# Download pre-trained weights (see PhoreGen README)
```

Generation workflow (`run_phoregen.py`):

```python
"""
1. Load pharmacophore JSON from spike_to_pharmacophore.py
2. Run PhoreGen sampling:
  - N = 1000 molecules per pharmacophore model
  - Batch size tuned for RTX 5080 VRAM (16GB)
3. Output: .sdf files with 3D coordinates aligned to pharmacophore
4. Each molecule includes:
  - 3D structure (aligned to pocket reference frame)
  - Pharmacophore match score
  - Chemical validity check
"""
```

Expected performance on RTX 5080:

- PhoreGen: ~500-1000 molecules in 5-10 minutes (diffusion sampling)
- PGMG: 10,000 molecules in ~30 seconds (VAE decoding)

Phase 3: Filtering Pipeline (`filter_generated.py`)

Multi-stage filtering (1000 generated → ~20-50 candidates → top 5 for FEP):

```
Stage 1: Chemical validity (RDKit)
  - Sanitize → remove invalid structures
  - Expected pass rate: ~80-90% (PhoreGen), ~70-80% (PGMG)

Stage 2: Drug-likeness
  - Lipinski Ro5: MW < 500, logP < 5, HBD ≤ 5, HBA ≤ 10
  - QED score > 0.3
  - Synthetic accessibility (SA) score < 6.0

Stage 3: PAINS filter (pan-assay interference)
  - Remove promiscuous scaffolds
  - RDKit PAINS filter

Stage 4: Pharmacophore re-validation
  - Re-score 3D overlap with PRISM spike pharmacophore
  - Require ≥ 3/N features matched within 1.5Å

Stage 5: Novelty check
  - Tanimoto similarity to PubChem < 0.85 (ensures novel IP)
  - Check against existing patent databases (optional)

Stage 6: Diversity selection
  - Cluster remaining molecules (Butina clustering, Tanimoto cutoff 0.4)
  - Select 1-2 representatives per cluster
  - Maximize scaffold diversity in final set

→ Top 5 diverse, drug-like, pharmacophore-matched novel molecules
→ Feed directly to Enhancement 1 (ABFE) for binding validation
```

Phase 4: Integration with FEP Pipeline (Week 3)

End-to-end automated pipeline:

```
PRISM Detection (2 min)
    │
    ▼
Spike Pharmacophore Extraction
    │
    ├──→ spike_to_pharmacophore.py
    │
    ▼
PhoreGen Generation (10 min, 1000 mols)
    │
    ▼
Filtering Pipeline (5 min, → top 5)
    │
    ├──→ gpu_dock.py (pose generation for top 5)
    │
    ▼
ABFE Validation (1-3 days, top 5)
    │
    ▼
Publication-Quality Report
  - Novel molecule structures (SMILES + 3D SDF)
  - ΔG_bind ± error for each
  - Pharmacophore match analysis
  - Drug-likeness metrics
  - Novelty assessment
  - IP status (Tanimoto to nearest known)
```

2.5 PRISM-Specific Leverage

Why PRISM + Generative Pharma is uniquely powerful:

1. **Spike pharmacophore = physics-derived constraints.** Standard pharmacophore models come from co-crystal structures (requires existing ligand) or computational docking (circular). PRISM's spike pharmacophore comes from UV-excited aromatic dynamics, EFP electrostatics, and LIF thermal fluctuations — physical measurements on the APO pocket. No prior ligand knowledge contamination.
2. **Novel pockets → novel molecules → novel IP.** The entire chain is unprecedented: PRISM finds a pocket nobody knew about, generates molecules nobody has made, validates binding from first principles. Every hit is potentially patentable.
3. **Intensity weighting provides confidence ranking.** Not all pharmacophore features are equal. PRISM's spike intensities encode how strongly each molecular interaction is "demanded" by the pocket. High-intensity BNZ spikes mean the pocket REQUIRES an aromatic interaction there. PhoreGen can use weights to prioritize feature matching.
4. **Per-residue resolution.** PRISM doesn't just say "aromatic here" — it says "TYR-142 spike at (65.2, 62.1, 40.8) with intensity 0.95, wavelength 274nm, water_density 0.3." This residue-level detail enables exclusion sphere placement that prevents steric clashes with specific protein atoms.

2.6 Risks and Mitigations

Risk	Impact	Mitigation
PhoreGen quality on PRISM-derived pharmacophores	Poor feature matching	PGMG fallback + manual pharmacophore refinement
Generated molecules not synthesizable	Useless hits	SA score filter + Enamine REAL overlap check
GPU memory for PhoreGen diffusion	OOM on RTX 5080	Reduce batch size, use PGMG for bulk
Pharmacophore feature mapping incorrect	Wrong molecule chemistry	Validate on known targets (KRAS/TEAD2) first
PhoreGen not maintained	Dependency rot	Pin versions, fork repo, PGMG as backup

---

COMBINED PIPELINE: PRISM-4D → GENPHORE → FEP

Full Workflow Specification

```
┌──────────────────────────────────────────────┐
│ PRISM-4D Detection (BLIND)                   │
│ nhs_rt_full --multi-stream 20 --multi-scale  │
│ --rt-clustering --lining-cutoff 8.0 --fast -v │
│ Output: Spike JSON + lining residues + pocket coords │
└──────────────────────┬───────────────────────┘
                       │
                       ▼
┌──────────────────────────────────────────────┐
│ Spike → Pharmacophore Translation            │
│ spike_to_pharmacophore.py                    │
│ Maps: BNZ→AR, CATION→PI, ANION→NI, etc.      │
│ Adds: exclusion spheres from lining residues │
│ Output: PhoreGen JSON + PGMG .posp           │
└──────────────────────┬───────────────────────┘
                       │
              ┌────────┴────────┐
              ▼                 ▼
┌──────────────────┐  ┌──────────────────┐
│ PhoreGen (1000)  │  │ PGMG (10000)     │
│ High quality 3D  │  │ Fast diversity   │
│ ~10 min          │  │ ~30 sec          │
└────────┬─────────┘  └────────┬─────────┘
         │                     │
         └──────────┬──────────┘
                    ▼
┌──────────────────────────────────────────────┐
│ Filtering Pipeline                           │
│ Validity → Drug-likeness → PAINS → Pharmacophore │
│ → Novelty → Diversity                        │
│ Output: Top 20-50 candidates                 │
└──────────────────────┬───────────────────────┘
                       │
                       ▼
┌──────────────────────────────────────────────┐
│ GPU Docking (UniDock/GNINA)                  │
│ gpu_dock.py — spike-constrained docking box  │
│ Pose generation + initial scoring            │
│ Output: Top 5 with docked poses              │
└──────────────────────┬───────────────────────┘
                       │
                       ▼
┌──────────────────────────────────────────────┐
│ ABFE Free Energy Validation                  │
│ OpenFE + GROMACS GPU-FEP                     │
│ 42 λ-windows × 5 ns × 3 repeats per ligand   │
│ Output: ΔG_bind ± error (kcal/mol)           │
└──────────────────────┬───────────────────────┘
                       │
         │
         ▼
┌─────────────────────────────────────────────┐
│ Classification & Report                  │
│ Per compound: ΔG, novelty, drug-likeness, IP status │
│ Per pocket: NOVEL/RECAPITULATED/CONFIRMED          │
│ Anti-leakage audit trail                 │
└─────────────────────────────────────────────┘
```

Hardware Requirements

Component	Minimum	Recommended
GPU	RTX 3080 (10GB)	RTX 5080 (16GB) ✓
VRAM for PhoreGen	8 GB	12+ GB
VRAM for FEP (GROMACS)	4 GB	8+ GB
System RAM	32 GB	128 GB ✓
Storage	100 GB per target	500 GB NVMe ✓
Wall time (full pipeline)	~2-4 days/target	~1-2 days/target

Dependency Stack

```
# Core
Python 3.10-3.12
CUDA 12.x (RTX 5080)
conda/mamba

# PRISM-4D (existing)
Rust + CUDA toolchain

# PhoreGen (new)
PyTorch 2.x
RDKit
e3nn (equivariant neural networks)

# PGMG (new)
PyTorch
RDKit
DGL (Deep Graph Library)

# OpenFE (new)
OpenMM 8.x
openfe >= 1.8
GROMACS 2024+ (GPU-FEP branch)
alchemlyb
arsenic

# Shared
RDKit (all stages)
NumPy, SciPy
matplotlib (reporting)
```

Implementation Priority

Priority	Task	Timeline	Dependency
P0	spike_to_pharmacophore.py	Week 1	PRISM spike output format
P0	Install PhoreGen + PGMG	Week 1	CUDA, conda
P1	filter_generated.py	Week 1-2	RDKit
P1	Install OpenFE	Week 2	conda
P1	prism_to_openfe.py	Week 2-3	OpenFE, gpu_dock.py
P2	run_fep.py + analyze_fep.py	Week 3	OpenFE working
P2	End-to-end integration test	Week 3-4	All components
P3	RBFE pipeline (lead opt)	Week 5+	ABFE validated
P3	Cloud scaling (AWS/Lambda)	Week 5+	Pipeline stable

Validation Plan

Test on known targets first (KRAS, TEAD2) before claiming novel results:

1. **KRAS G12C** (sotorasib pocket — known cryptic allosteric site)
    - Run PRISM blind → should detect Switch II pocket
    - Generate molecules → should produce sotorasib-like features
    - ABFE → compare to experimental ΔG of sotorasib (-11.2 kcal/mol, IC50 = 21 nM)
    - Classification: RECAPITULATED (expected, validates pipeline)
2. **TEAD2-YAP** (VT-103/K-975 pocket — known PPI inhibitor site)
    - Run PRISM blind → should detect lipid pocket
    - Generate molecules → compare to known TEAD inhibitor pharmacophores
    - ABFE → compare to known IC50 data
3. **Novel target** (PRISM-discovered pocket with no known ligands)
    - Full blind pipeline
    - Classification: NOVEL only if passes all anti-leakage firewalls

Success metrics:

- ABFE on known compounds: within 2 kcal/mol of experimental ΔG
- Generated molecules: ≥ 80% chemical validity, ≥ 50% drug-like
- Pharmacophore match: ≥ 3/N features within 1.5 Å
- Novelty: Tanimoto < 0.85 to nearest known compound
- Pipeline wall time: < 3 days per target on single RTX 5080