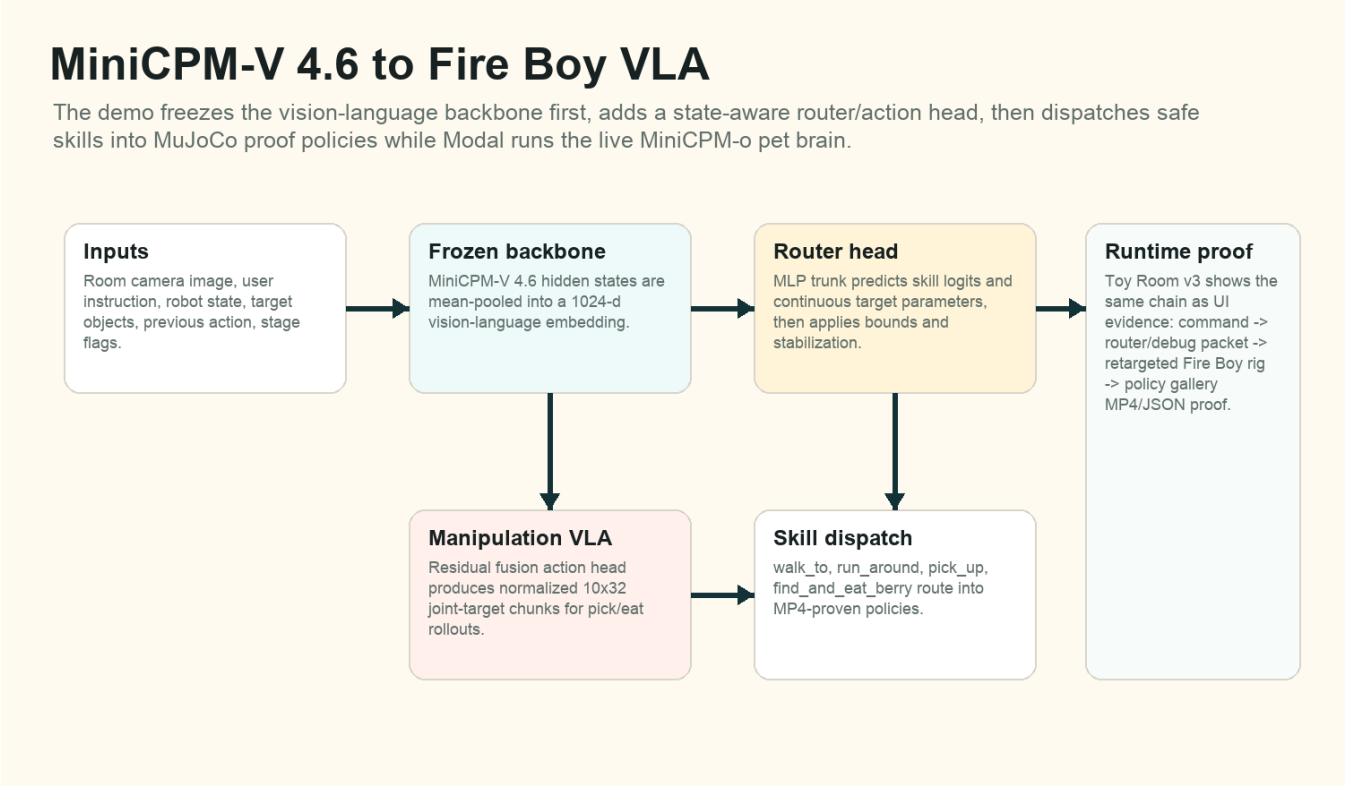
Turning MiniCPM-V 4.6 Into a Fire Boy Vision-Language-Action Policy

Technical report v3 - A practical, arXiv-style account of AI-age virtual pets, frozen vision-language backbones, SAM-to-rig asset construction, residual action heads, skill-parameter routers, MuJoCo/Newton simulation lanes, RunPod training evidence, Modal inference, and OpenAI Codex-assisted experiment iteration.



Abstract. This paper documents the path from a general multimodal model, MiniCPM-V 4.6, to an embodied VLA controller for Fire Boy, a small rigged virtual pet. The central engineering decision was to freeze the MiniCPM-V backbone first, extract a stable vision-language embedding, and train small action modules around it: a state-residual action head for contact-rich manipulation and a skill-parameter router for demo-safe open-ended commands. The best shipped route is not a single monolithic low-level policy. It is a router that selects proven skills and predicts continuous target parameters, while separate policy lanes supply movement, manipulation, retargeting, proof videos, and runtime UI evidence.

Core result. Single-step manipulation failed at 0/20. Chunked manipulation reached 20/20. A frozen MiniCPM-V residual action head reached 3/3 pick and 3/3 eat on RunPod. A LoRA manipulation variant also reached 3/3 and 3/3 in the eval-only gate. The final frozen skill-parameter router reached 512/512 skill decisions with mean parameter MAE 0.0170; the LoRA router kept perfect skill classification but had worse eval MAE at 0.0629, so the frozen router remained the safer promoted model.

Project Objective: An AI-Age Virtual Toy

The project objective came from a prior attempt to build a real robot. The practical question became: what if the most important part of the robot could be built first as a virtual toy? Fire Boy is meant to sit in the lineage of Talking Tom, Tamagotchi, and imaginary Pokemon-like companions, but updated for multimodal AI, physics, memory, and learned action. The goal is not a mascot over a chatbot. The goal is a small creature that sees its room, understands speech and text, forms habits, plays with objects, reacts physically, and makes bounded decisions on its own.

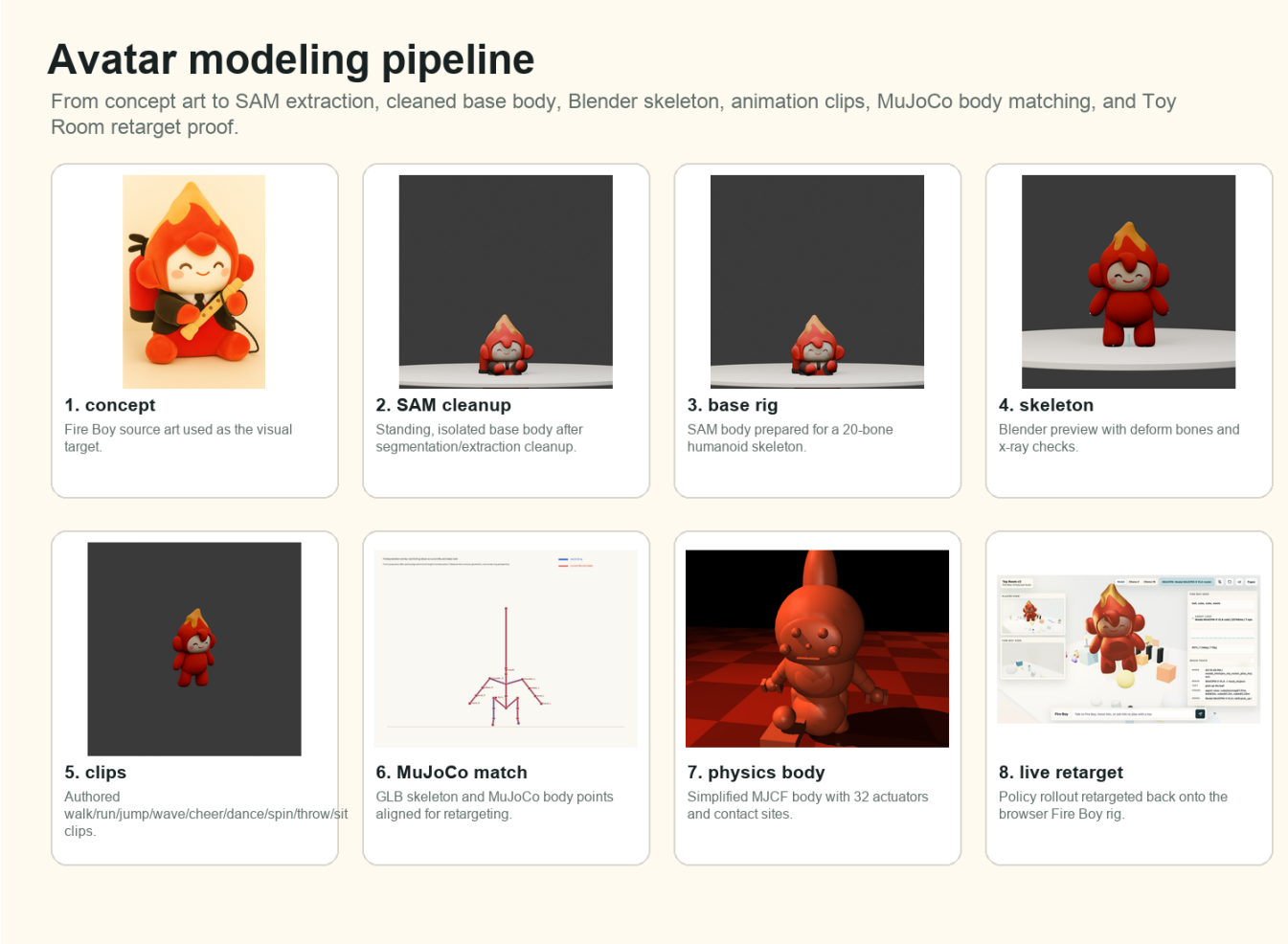
That motivation changes the technical requirements. A virtual pet cannot feel alive if every behavior is a hard-coded animation. It needs perception, action, feedback, memory, and a loop that can improve from rollouts. A real robot would add hardware cost and safety complexity immediately; a virtual body lets the project explore embodiment first: rigging, collisions, grasping, retargeting, reward design, state tracking, and personality. The hackathon artifact is intentionally small, but it tests the core thesis that a tiny VLA plus a toy-room physics stack can become the seed of a living digital companion.

Design goal	What it means in this build
Natural interaction	Typed/spoken commands map into PET actions, router decisions, and visible Fire Boy behavior.
Embodied agency	The pet receives room images, state, targets, prior action, and stage flags before acting.
Habits and personality	Future work adds memory, style rewards, curiosity rewards, and multi-agent toy interactions.
Small-model access	MiniCPM-class backbones keep experiments closer to consumer-grade deployment than giant VLA stacks.
Real-robot bridge	A virtual humanoid pet can eventually share training ideas with physical humanoid robots once hardware is ready.

Contributions

- A reproducible asset pipeline from concept image to SAM-derived base body, Blender skeleton, GLB animation clips, MuJoCo matching, and Toy Room browser retargeting.
- A MiniCPM-V 4.6 frozen-encoder VLA router that predicts four embodied skills and six continuous parameters with 814,090 trainable head parameters.
- A residual low-level action head with 1,078,912 trainable parameters for 10-step x 32-actuator manipulation chunks.
- A direct comparison between frozen-router and LoRA-router behavior, including exact adapter size: 4,743,168 LoRA trainable parameters across language-model layers 0-23.
- An evidence-first workflow: every promoted behavior is tied to manifest rows, eval JSON, screenshots, or rendered MP4/GIF proof.

Asset Modeling: From Concept Art to Rigged Avatar



The embodied policy only works because the visual character, browser rig, and simulator body share a consistent skeleton. Fire Boy began as a 2D source character, then moved through SAM-style segmentation and 3D extraction, cleanup, base-body rigging, skeleton placement, animation authoring, and MuJoCo body matching. The rig report records a single connected body component, 3,311 vertices, zero unweighted vertices, and no rigidified runaway shells for the Fire Boy base body.

Pipeline stage	Artifact	Implementation note
Source art	potential-char-images/fire-boy.png	Reference image for identity, silhouette, flame crown, proportions, and color.
SAM/extraction	assets/generated/previews/fire-boy-sam-cleaned.png	The raw generated/SAM body is cleaned into a standing unclothed base body.
Rig build	fire-boy-rig/working/build_rig.py	Builds Root -> Hips -> Spine -> Chest -> Neck -> Head -> Crown, plus shoulders, elbows, hands, hips, knees, and feet.
Animation export	fire-boy-rig/working/animate_and_export.py	Authors idle, walk, run, jump, wave, cheer, dance, spin, throw, and sit clips for glTF export.
Browser runtime	fire-boy-rig/fire-boy-rigged-full.glb	Three.js loads the same GLB for Toy Room v3 and the rigged-character inspector.

```
# Rebuild rigged avatars from the repo root
blender --background --python fire-boy-rig/working/build_rig.py
```

```
blender --background --python fire-boy-rig/working/animate_and_export.py

# Rig quality checks written by the build
fire-boy-rig/working/fire-boy-rig-report.json
fire-boy-rig/previews/fire-boy-fullrig-bones.png
```

Skeleton, MuJoCo Body, and Retarget Synchronization

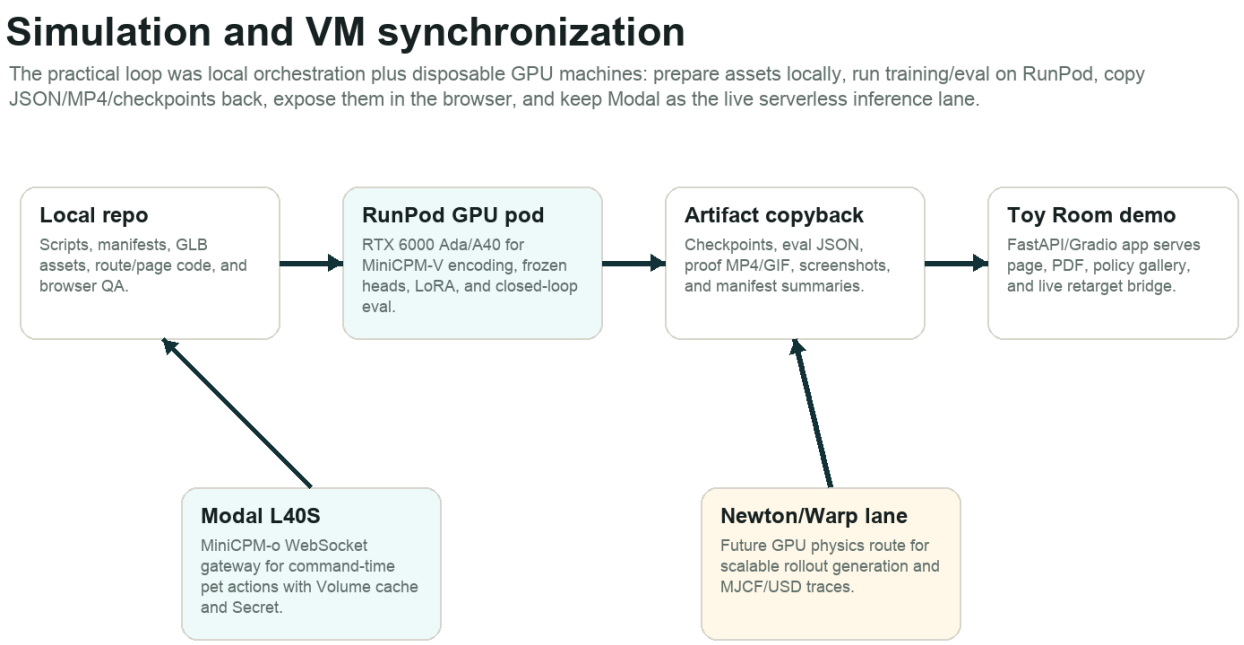
The MuJoCo model is deliberately not the original high-detail mesh. The training body is a simplified MJCF tree with robust primitive geometry, contact sites, and 32 actuators. This avoids fragile raw-mesh collision during early training. The skeleton inspection script maps the GLB rig points to the MuJoCo body points; the current alignment report has 22 common points and a max normalized point error of 0.0 for the exported matching pass.

Quantity	Value	Meaning
GLB/MuJoCo common points	22	Named skeleton landmarks shared by browser rig and simulator body.
Max normalized point error	0.0	Alignment error after the matching transform in the inspected skeleton overlay.
MuJoCo bodies	23	Simplified physical body tree for stable control.
MuJoCo joints	33	Root slides/yaw plus limb, torso, wrist, gripper, and leg joints.
Actuators	32	The 32-dimensional action vector used by action chunks.

```
# Body proof lane
python fireboy-vla-physics/src/inspect_fireboy_alignment.py
python fireboy-vla-physics/src/render_articulated_fireboy.py --mode body

# Runtime bridge idea
MuJoCo qpos/action trace -> retargetTrajectory -> Three.js GLB bones
object state -> heldObjectAnchor -> browser toy interaction
```

Virtual Machines, Artifact Sync, and Reproducibility

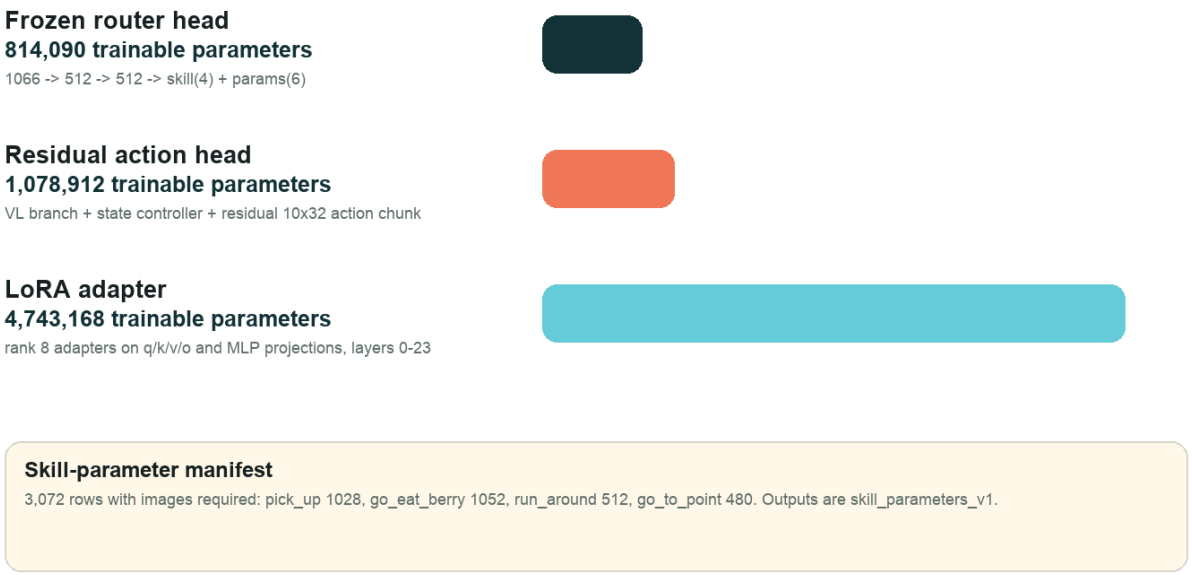


The system used local development for orchestration and UI, RunPod GPU pods for MiniCPM-V encoding/training/evaluation, and Modal for the live MiniCPM-o browser action brain. The important discipline was to copy artifacts back into stable repo paths before claiming success: checkpoints under runpod-artifacts/checkpoints, MP4/GIF evidence under runpod_artifacts, summaries under vla-rollouts, and browser-facing proof pages under frontend. Pods were treated as disposable; the repo became the memory of the experiment.

- Local machine: authored scripts, generated pages/PDF, served FastAPI/Gradio app, ran browser QA, and kept the policy registry visible.
- RunPod RTX 6000 Ada: scaled frozen residual MiniCPM-V action-head training and manipulation evaluation.
- RunPod NVIDIA A40: trained and evaluated the frozen and LoRA skill-parameter routers.
- Modal L40S: hosted the MiniCPM-o 4.5 WebSocket gateway with Volume cache and Secret-based model access.
- NVIDIA Newton/Warp: documented as the next GPU physics lane for rollout throughput and USD/MJCF-compatible traces.

VLA Head Sizes and Sampling

VLA head sizes and sampled rows



The promoted frozen router trains only the head: 814,090 parameters over a 1066-dimensional fused input. The input is 1024 MiniCPM-V vision-language features plus 42 state features. The residual action head is larger, 1,078,912 parameters, because it contains a state controller branch, a vision-language branch, and a residual fusion branch that outputs a 10-step by 32-actuator action chunk. The LoRA experiments add 4,743,168 adapter parameters at rank 8 across q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj modules on 24 language-model layers.

Model piece	Trainable parameters	Notes
Frozen skill-parameter router head	814,090	Two 512-wide MLP trunk layers plus skill and parameter heads.
Residual action head	1,078,912	State branch 441,152; VL branch 328,192; residual branch 309,568.
LoRA adapter	4,743,168	360 tensors across layers 0-23.
Base MiniCPM-V 4.6	Frozen for promoted router	The report trains heads/adapters, not the full backbone, in the promoted path.

```
# Router head architecture
features = concat(normalize(vl_embedding_1024), normalize(robot_state_42))
h = SiLU(Linear(1066, 512)(features))
h = SiLU(Linear(512, 512)(h))
skill_logits = Linear(512, 4)(h)
params = Linear(512, 6)(h)

loss = cross_entropy(skill_logits, skill_id) + 0.35 * mse(params, target_params)
```

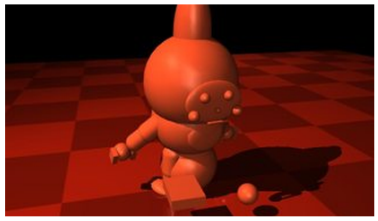
Sampling and Rollout Rows

Data and rollout evidence

Images are not decoration: they are the observation side of the VLA rows and the proof side of the browser demo.

sample frame

Rendered room observation used in VLA manifests.



navigation

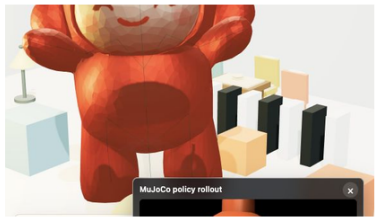
Target-relative frames sampled every 5 simulation steps.

action chunk

Rows pair image + state with a short action horizon.

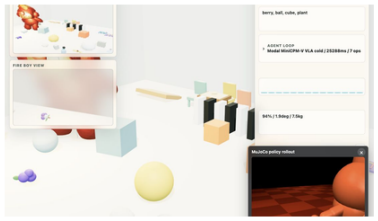
scene variation

Different camera/target arrangements for grounding.



pickup bridge

Closed-loop proof retargeted into Toy Room v3.

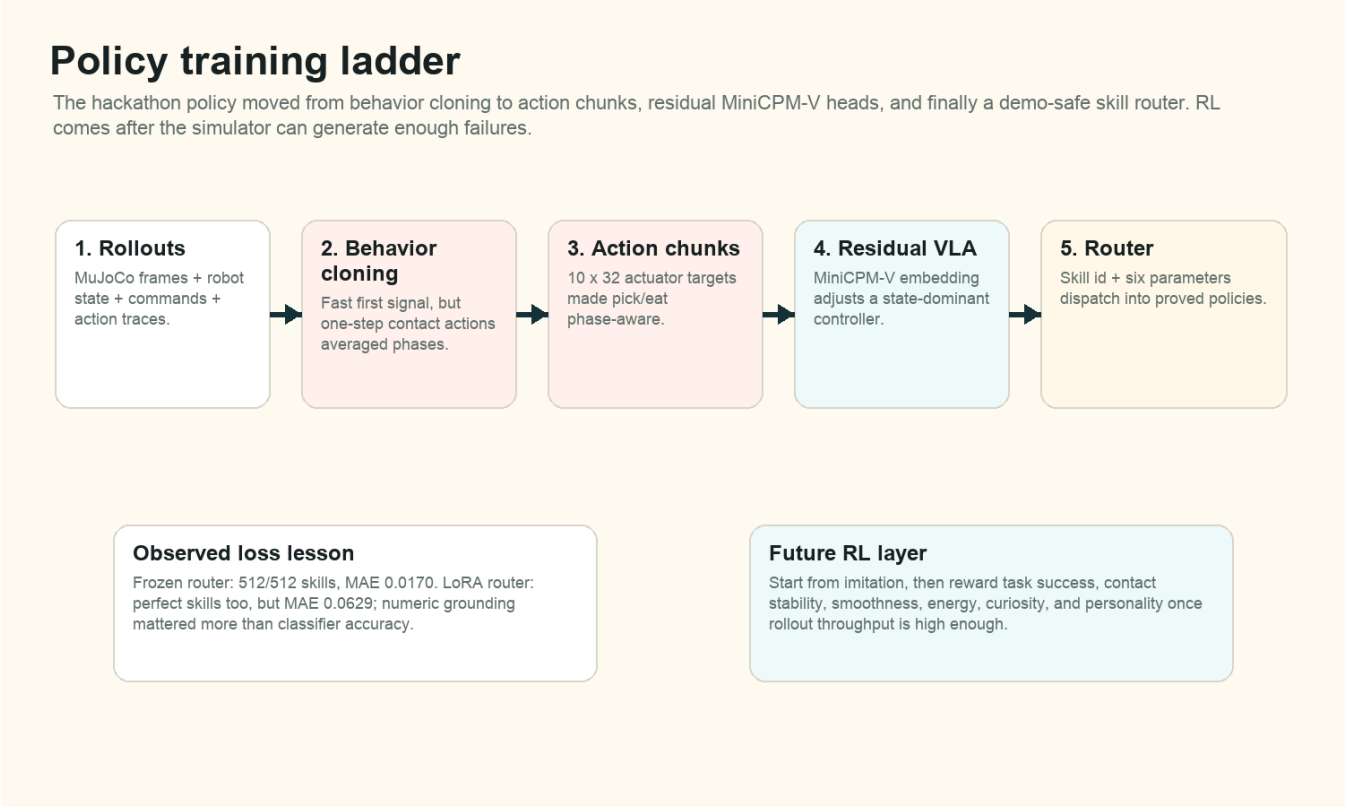


eat berry

Manipulation rollout with object/mouth state.

The dataset is assembled from rendered rollout observations rather than from still screenshots alone. The first four-skill manifest contained 64 episodes and 2,368 image/manifest rows, with 16 episodes per task and a stride of 5 simulation steps. The focused manipulation manifest expanded contact tasks to 144 episodes and 6,192 rows, still using 10-step chunks. The all-skill skill-parameter manifest wrote 3,072 rows with images required: 1,028 pick_up, 1,052 go_eat_berry, 512 run_around, and 480 go_to_point.

Policy Training Ladder



The policy was trained as a staged system instead of a single heroic end-to-end model. Behavior cloning was the first useful tool because it directly pairs observations with successful actions. It made the data path inspectable, but it also exposed why one-step control is weak for contact. Action chunks fixed the phase problem for manipulation by asking the head to emit a short future trajectory. The residual VLA head then let MiniCPM-V features adjust a state-dominant controller. The skill-parameter router became the public demo layer because it routes into skills that already have closed-loop proof.

- Why behavior cloning first: it is stable, cheap, and lets every row be inspected as image + command + state + label.
- Why action chunks: they preserve approach, contact, lift, and transfer phases that a single next action averages away.
- Why a router for the demo: the router can choose the right proved skill while low-level policies continue improving separately.
- Why RL later: reinforcement learning is powerful only after the simulator can generate enough randomized successes, failures, recoveries, and reward diagnostics.

1. Problem Statement

MiniCPM-V 4.6 already sees images and reads language, but it does not naturally emit MuJoCo joint targets, stable navigation parameters, or a browser-safe action contract. A VLA conversion therefore has three jobs: preserve the multimodal understanding, expose robot state, and constrain the model's output into actions that can be executed, evaluated, and debugged. Fire Boy makes this concrete: the model sees a toy room frame, reads a command such as “pick up the berry” or “walk toward me,” receives body and target state, and must produce either a short joint-action chunk or a skill with continuous parameters.

- Inputs: image, instruction, robot state, target/object metadata, previous action, task and stage flags.
- Backbone: MiniCPM-V 4.6 with a 1024-dimensional pooled vision-language feature in the training artifacts.

- Outputs: either normalized joint target chunks, or a high-level skill id plus target_x, target_y, target_z, radius, speed_hint, and object_is_berry.
- Runtime contract: every model decision must fit the Toy Room PET action path, policy gallery proof, and trace/debug panels.

2. General Recipe: Turning Any VLM Into a VLA

The Fire Boy system is useful as a recipe because it does not require the base model to be born as a robot model. In theory, any model with stable image-language hidden states can become a VLA if the surrounding system supplies proprioception, action labels, loss functions, and closed-loop evaluation.

Step	Purpose	Fire Boy implementation
Freeze first	Avoid destabilizing visual-language knowledge while the action interface is still immature.	MiniCPM-V parameters are frozen for the first router/action-head runs; only compact heads learn.
Add state	A VLM cannot infer joint velocity, grasp status, root pose, or previous action from pixels alone.	State vectors include navigation geometry, hand/mouth/object positions, task flags, stage flags, grasp/eaten bits, and previous action.
Choose action granularity	Low-level actions are expressive but brittle; skills are safer for demos and sparse data.	The project tested one-step actions, action chunks, residual low-level heads, and finally a skill-parameter router.
Train on successful rollouts	Behavior cloning needs aligned observations and actions from trajectories that actually solve the task.	RunPod-generated MuJoCo rollouts and MiniCPM-V image manifests produced JSONL rows with images, state, commands, and labels.
Close the loop	Offline loss is not enough for physics. The model must survive compounding error.	Every promoted claim required eval JSON and MP4/GIF proof.
Adapt later	LoRA can tune the backbone once the head and labels are reliable.	LoRA rank 8 / alpha 16 runs were tested after frozen residual/router baselines.

2.1. VLA Conversion Methods Beyond This Build

The current method is deliberately conservative: freeze MiniCPM-V, expose robot state, and train compact heads. It is a good hackathon route because it is cheap and debuggable, but it is not the only way to build a VLA. The next serious study should compare several routes under the same simulator, same objects, and same closed-loop scoring.

Method	Mechanism	Main advantage	Main risk
Frozen encoder + head	Train MLP/router/action heads on top of MiniCPM-V embeddings.	Stable and cheap; good for first demos.	May not adapt visual-language features to action semantics.
LoRA / QLoRA	Add low-rank adapters to attention/MLP layers plus heads.	More expressive while still much cheaper than full fine-tuning.	Can preserve skill accuracy while degrading coordinates, as this run showed.
Action-token fine-tuning	Teach the model to emit structured skill tokens, coordinates, or action chunks.	Closer to a single-brain controller.	Needs strict validation or malformed actions can enter the runtime.
Diffusion / flow policy head	Generate trajectory chunks from a learned action distribution.	Good for playful or multimodal behavior where many actions are valid.	Harder to debug and usually needs more data.

Method	Mechanism	Main advantage	Main risk
World model + planner	Predict future latent states, then plan or search over actions.	Best for long-horizon pet goals and habits.	More moving pieces and more failure modes.
Imitation + RL	Start from behavior cloning, then improve with simulated rewards.	Can discover recovery behavior and style.	Reward hacking if the sim and reward are immature.

3. Model and Router Architecture

The final router is intentionally small. MiniCPM-V supplies a pooled multimodal embedding. A state vector supplies the body and task variables the image cannot reliably encode. The router concatenates normalized vision-language and state features, passes them through a SiLU MLP trunk, and emits two heads: a categorical skill head and a continuous parameter head.

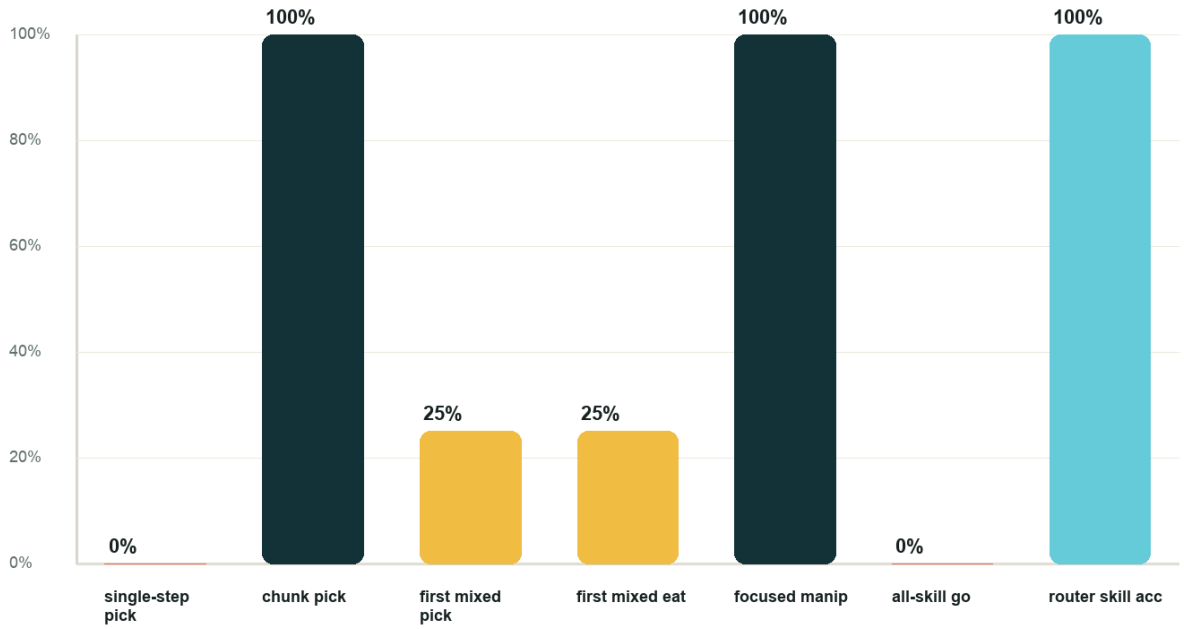
Router part	What it does	Why it exists
MiniCPM-V encoder	Runs the image/instruction through MiniCPM-V 4.6 and mean-pools final hidden states with the attention mask.	Provides visual-language grounding without training the whole model on the first pass.
State constructor	Builds direct or derived features from robot_state, scene targets, nav clock, hand/mouth/object positions, task flags, stage flags, previous action, grasp/eaten bits.	Prevents the VLM from hallucinating proprioception and gives the head closed-loop context.
MLP trunk	Linear -> SiLU -> Linear -> SiLU over the fused vector.	Small enough to train fast on RunPod while still learning task-conditioned parameter maps.
Skill head	Predicts walk_to, run_around, pick_up, or find_and_eat_berry.	Turns open-ended language into a bounded action vocabulary with proof videos.
Parameter head	Predicts target_x, target_y, target_z, radius, speed_hint, object_is_berry.	Carries continuous grounding into the skill policies.
Bounds and stabilizers	Clips parameters to scene-safe ranges, copies explicit target coordinates from parsed scene metadata, and prefers heuristic skill when commands are explicit unless forced neural.	Makes the demo robust: the learned router participates without allowing unsafe or incoherent outputs.
Dispatch	Maps the chosen skill to registry:walk_to, registry:run_around, registry:pick_up, or registry:find_and_eat_berry.	Connects model choice to the same execution path used by Toy Room v3 and the policy gallery.

This router is not a cheat around VLA. It is one of the two useful forms of VLA for small embodied demos: instead of emitting every motor command directly, it emits the next embodied skill and its grounded parameters. That is still vision-language-action: pixels and language condition a physical action, but the action is represented at the level where the available data is reliable.

4. Action Heads and Why Chunks Won

The low-level branch trained action heads that output joint targets. The failed first attempt predicted a single action per state. For manipulation, that collapses phases such as approach, reach above, descend, close, lift, and mouth transfer into an averaged control. The result looked plausible in loss but failed closed-loop contact. The fix was action chunking: predict a short horizon of future joint targets so the head can represent phase progression.

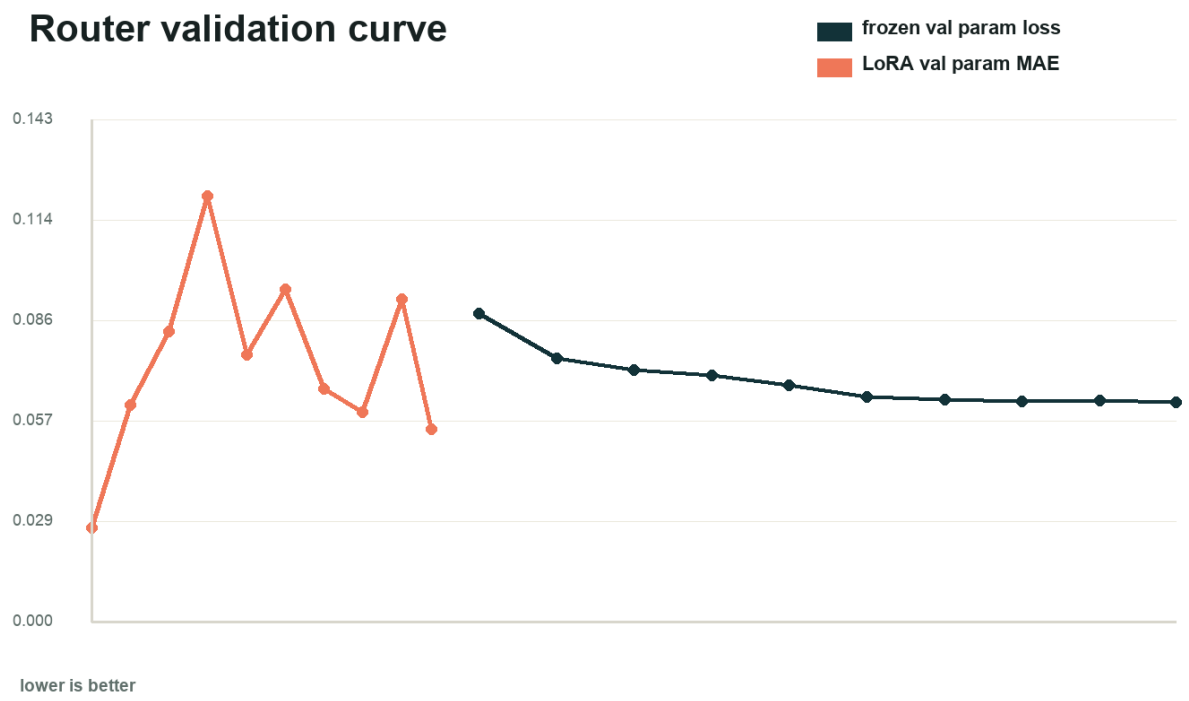
Experiment success rates

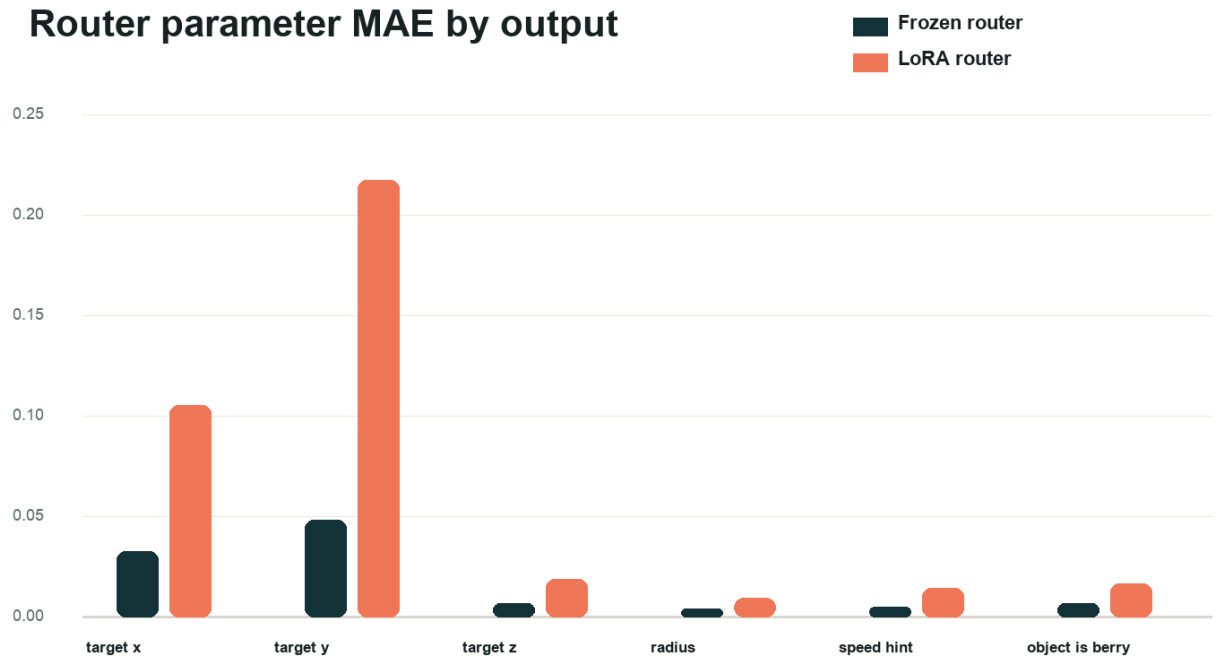


Experiment	Data / architecture	Closed-loop result	Interpretation
Single-step pick/eat	State-action policy, no chunk horizon.	pick_up 0/20; go_eat_berry 0/20.	Averaged across contact phases; could not close grasp/eat loops.
Chunked manipulation	Action chunks for pick_up and go_eat_berry.	pick_up 20/20; go_eat_berry 20/20.	Temporal chunking made phase transitions learnable.
First mixed manifest	64 episodes, 2368 image rows, four tasks.	pick_up 2/8, eat 2/8, run 8/8, go_to 7/8.	Pipeline worked, but contact tasks were underrepresented.
Focused manip manifest	144 episodes, 6192 rows, pick/eat focus.	pick_up 12/12; go_eat_berry 12/12.	Data balance mattered more than model novelty.
Frozen residual MiniCPM-V head	2048 rows, state_residual_fusion_v1, VL residual scale 0.12, action_std_floor 0.01.	pick_up 3/3; go_eat_berry 3/3.	State-dominant controller plus small VL residual was stable.
All-skill direct action head	3072 rows across manipulation and movement.	pick_up 2/2, run 2/2, eat 0/2, go_to 0/2.	One shared low-level head was not reliable enough for the demo.

5. Losses, MAE, and What the Curves Say

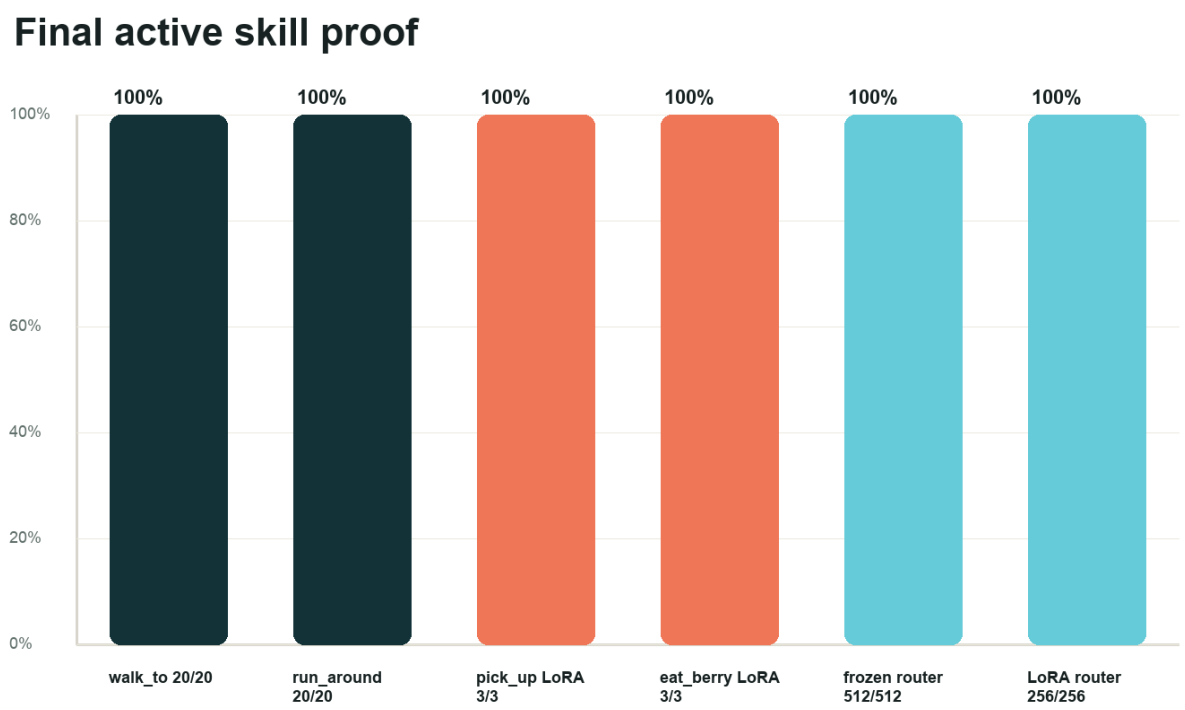
The promoted frozen router trained on 768 rows (645 train, 123 validation) and evaluated on 512 rows. Its skill accuracy was 1.000, with overall parameter MAE 0.0170. The LoRA router trained on 512 rows and evaluated on 256 rows. It also achieved perfect skill accuracy, but its parameter MAE rose to 0.0629. That is why the frozen router is the safer public demo router even though the LoRA run proved the backbone can be adapted.





The LoRA router's confusion matrix was still perfect, so the failure mode was not command classification. The problem was parameter precision, especially target_y. For embodied control, small coordinate errors become visible because the downstream skill has to move through space and interact with objects. This is a good example of why VLA evaluation cannot stop at language accuracy.

6. Final Demo Policy Stack



Final skill	Policy source	Evidence
walk_to	mujoco_articulated_policy go_to_point clock policy	20/20 eval success, local Toy Room trajectory retargeting, player-camera target support.
run_around	mujoco_articulated_policy run_around	20/20 eval success and live route retargeting.
pick_up	MiniCPM-V LoRA manipulation checkpoint / rollout manifest bridge	3/3 eval gate; Toy Room v3 retargeted object anchoring to Fire Boy hands.
find_and_eat_berry	MiniCPM-V LoRA manipulation checkpoint / rollout manifest bridge	3/3 eval gate; live bridge reports eaten state and mouth transfer.
router	Frozen MiniCPM-V skill-parameter head	512/512 skill decisions, parameter MAE 0.0170, perfect 4x4 confusion matrix.

The runtime bridge retargets proof trajectories onto the Fire Boy GLB body. Movement skills use articulated MuJoCo rollout trajectories. Pick and eat use successful MiniCPM-V rollout-manifest joint states while preserving the object interaction. This distinction matters: the public demo is honest that contact-rich grasping is proven through recorded/evaluated policy paths and retargeting, not through a fully generalized learned grasp planner for every new object.

7. Simulation: MuJoCo, NVIDIA Newton, and Why Physics Was the Bottleneck

The project treated physics as the source of truth. MuJoCo generated the concrete policy proof for this build: eval JSON, MP4/GIF rollouts, body rendering, and Toy Room trajectory retargeting. NVIDIA Newton was investigated as the next GPU physics lane because it is designed around NVIDIA Warp/OpenUSD and can align with MuJoCo Warp-style

workflows. The plan was to load or adapt the Fire Boy MJCF/URDF asset, validate CUDA rollouts, export qpos/action traces, and produce MP4/USD/NPZ artifacts. The final artifact remains MuJoCo-backed; Newton is documented as the GPU scaling path rather than overclaimed as the source of the shipped proof.

- Newton lane target: Fire Boy asset -> Newton/MuJoCo Warp load test on cuda:0 -> rollout qpos/action trace -> MP4/USD/NPZ proof.
- Fallback rule: if direct Fire Boy adapter fails, validate Newton with a supported robot example and keep Fire Boy in the MuJoCo lane.
- Reason to care: VLA quality is bounded by simulation quality. Better GPU rollout throughput means more diverse image-state-action data and more closed-loop failures found before the browser demo.

8. Hardware and Cloud Runtime

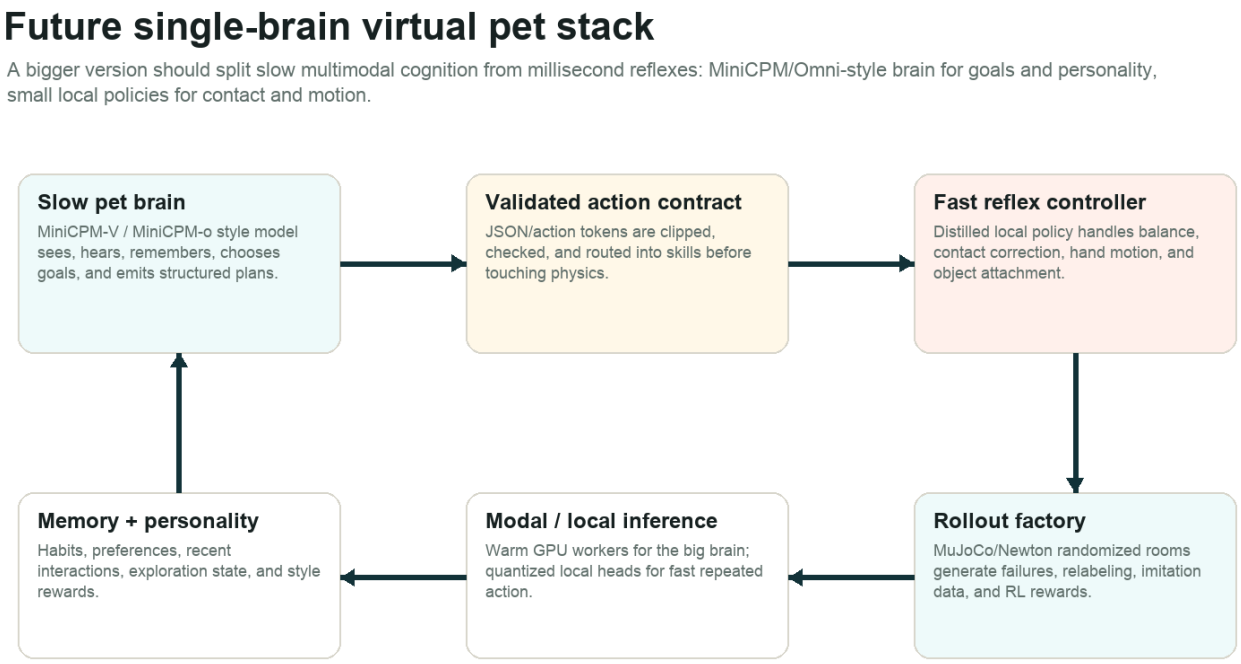
Component	Hardware / service	Role
Modal MiniCPM-o runtime	Modal L40S GPU, Modal Volume minicpm-omni-cache, Modal Secret for Hugging Face token.	Live Toy Room v3 action-brain gateway for MiniCPM-o 4.5, with serverless scale-to-zero behavior.
MiniCPM-V residual action head	RunPod NVIDIA RTX 6000 Ada Generation.	Frozen MiniCPM-V 4.6 manipulation head, 2048 rows, 3/3 pick and 3/3 eat.
Frozen skill-parameter router	RunPod NVIDIA A40.	MiniCPM-V 4.6 frozen router, 512-row held-out eval with perfect skill accuracy.
LoRA skill-parameter router	RunPod NVIDIA A40.	Rank 8 / alpha 16 adapter experiment; perfect skill selection but worse parameter MAE.
Local app	FastAPI/Gradio-style local server plus Three.js/CANNON frontend.	Playable browser surface, screenshot evidence, PDF/page delivery, policy gallery, and debug APIs.

9. Modal Inference and Toy Room Integration

Modal is used for serverless multimodal inference rather than batch training in the shipped browser path. The Modal app wraps the OpenBMB MiniCPM-o 4.5 demo stack, caches model weights in a Modal Volume, and exposes a WebSocket chat gateway. Toy Room v3 sends one command-driven /api/pet-action call per typed, spoken, or quick-button command. The server opens one Modal turn, validates streamed model output into the PET action JSON contract, records tokens/latency/debug fields, and sends a bounded update back to the browser.

This matters for demo quality because the model is load-bearing but not mysterious. The page can show which brain mode is active, how many tokens were used, whether a fallback policy ran, and which target or skill was selected. The same transparency pattern is used for the VLA router: the compact router result contains skill, confidence, neural skill, parameters, dispatch, policy kind, model id, device, and latency.

9.1. Full Fine-Tuning and Single-Brain Future



The long-term version can use a MiniCPM-V, MiniCPM-o, or future MiniCPM/Omni-class model as the slow high-level pet brain. Full fine-tuning would train on multimodal episodes: observation frames, state summaries, user utterances, action tokens, reward tags, memory events, and outcome labels. The model could learn to emit structured plans such as choose_goal, choose_skill, target_object, target_pose, and style. A validated runtime would then translate those plans into fast motor policies.

A single brain is attractive because it can unify perception, memory, language, and action. The practical architecture should still be hierarchical. The slow brain can run on Modal or a warm GPU worker and decide what the pet wants to do. A small local reflex policy should run in milliseconds and handle continuous motion, contact correction, object attachment, and animation blending. This split is what makes the pet feel responsive while still letting a multimodal model provide personality and reasoning.

Optimization target	Technique
Startup cost	Warm Modal containers, cached model weights, preloaded tokenizer/processor, and persistent volume cache.

Optimization target	Technique
Per-turn latency	Compress state, reuse image/frame embeddings, stream output, and validate compact action JSON instead of long prose.
Millisecond actions	Distill learned skills into local heads; run WebGPU/WASM/ONNX or a small server-side policy in the inner loop.
Cost	Quantization, LoRA adapters, smaller MiniCPM-class backbones, and routing only hard perception moments to the big brain.
Quality	Mine failed rollouts, add recovery demonstrations, run DAgger-style relabeling, and use RL only after reward checks are robust.

10. Codex as the Experiment Partner

OpenAI Codex was not only used to write prose at the end. The repo history shows Codex-assisted implementation and hardening across the build: shipping Toy Room v3, adding the Fire Boy command loop, wiring a MiniCPM-V action brain, routing Toy Room v3 through Modal MiniCPM, adding trace diagnostics, hardening Modal WebSocket timeouts, keeping the MiniCPM-V loop live, making locomotion and pickup physical, grounding pickup targets, and adding grounded gestures. In practical terms, OpenAI Codex helped turn a messy research loop into commit-sized steps: scaffold a path, run the app, inspect artifacts, patch the exact boundary that failed, and preserve the evidence trail.

Commit theme	What Codex helped stabilize
Toy Room v3 ship path	Single Fire Boy demo page, controls, command loop, and playable action surface.
MiniCPM-V action brain	Vision/action route, endpoint configurability, and fallback honesty.
Modal MiniCPM route	Serverless WebSocket adapter, action JSON validation, timeout hardening, and trace metrics.
Physics grounding	Pickup targets, locomotion, gestures, policy gallery proof, and browser-visible debug panels.
Research artifacts	Runbooks, experiment summaries, screenshots, PDF/page generation, and presentable demo narrative.

11. What Failed and Why That Was Useful

The failures are the most important part of the paper because they explain the final architecture. Direct low-level VLA is seductive, but small embodied agents expose every weakness: contact discontinuities, phase averaging, root drift, saturation, and data imbalance. The direct go_to_point MiniCPM-V variants failed repeatedly at 0/5 in one-step, root-velocity, and recovery-data settings. The all-skill low-level head failed eat and go_to_point despite passing pick and run. These failures argued for a router-first public demo and a continued research lane for low-level policies.

- One-step actions lost phase information; action chunks fixed manipulation.
- A single mixed low-level head underfit the different control regimes of contact manipulation and navigation.
- LoRA improved adaptability but did not automatically improve numeric grounding; frozen router parameters were better in the held-out eval.
- Closed-loop evaluation found problems that offline loss alone would have hidden.

12. Limitations and Next Work

The current demo is best described as a practical VLA system, not a solved general robotics model. It can route commands, ground targets, and show evaluated policy rollouts in action. It does not yet prove arbitrary contact-rich

grasping of every object, long-horizon household planning, or Newton-native Fire Boy training at scale. The right next work is to push the Newton/Warp lane, generate larger randomized rollouts, keep the router as the safe outer policy, and train specialized low-level heads per skill until each skill can survive broader closed-loop randomized tests.

- Promote frozen router for demo; keep LoRA router as a research checkpoint until parameter MAE improves.
- Expand movement data with randomized targets including player-camera, object affordances, and spatial references.
- Train skill-specific low-level heads rather than forcing one shared head to solve all regimes.
- Use Newton/Warp or another GPU physics path to multiply rollout throughput and discover sim failures earlier.
- Keep every claim backed by eval JSON plus rendered proof video, then reflect that evidence on the page.
- Move beyond the virtual pet level with randomized object physics, multi-room tasks, language-conditioned reward models, longer action chunks, tactile/contact labels, and a fully Newton-native training/eval lane.
- Distill the router into smaller on-device policies once the skill interface stabilizes, while keeping MiniCPM-V as a sparse visual planner for hard perception moments.
- Add long-lived memory and habit formation so Fire Boy can develop preferences, routines, and a recognizable personality over repeated sessions.
- Revisit the original multi-agent toy idea: several virtual pets interacting with each other, the user, and shared objects in ways that can be observed but not fully scripted.
- Use the same virtual-pet control stack as a preparation ground for future humanoid robots: perception, state, safety bounds, action tokens, low-level controllers, and recovery policies transfer conceptually even when hardware changes.

References

Reference	Why it matters here
OpenBMB MiniCPM-V 4.6 model card, https://huggingface.co/openbmb/MiniCPM-V-4.6	Backbone used for the frozen vision-language encoder and LoRA experiments.
Kirillov et al., Segment Anything, arXiv:2304.02643, https://arxiv.org/abs/2304.02643	Segmentation/asset extraction inspiration for the SAM-based character workflow.
MuJoCo documentation, https://mujoco.readthedocs.io and https://mujoco.org	Physics simulator used for articulated body, rollouts, and policy proof.
NVIDIA Newton Physics Engine, https://developer.nvidia.com/newton-physics and https://newton-physics.github.io/newton/stable/	Future GPU physics lane built around Warp/OpenUSD and MuJoCo Warp integration.
Modal docs, https://modal.com/docs	Serverless GPU/runtime infrastructure used by the MiniCPM-o gateway.
OpenAI Codex, https://openai.com/codex/	Agentic coding environment used to scaffold, debug, document, and package the experiment.

Appendix: key artifact paths include Fireboy-training-policy-vla/minicpm-vla-action-head-scaffold.md, Fireboy-training-policy-vla/overnight-goal-runpod-newton-kimodo-vla-plan.md, Fireboy-training-policy-vla/runpod-results-2026-06-15.md, fireboy-vla-physics/src/vla_router_runtime.py, src/vla_router_policy.py, and fireboy-vla-physics/build/fireboy-policy-proof-bundle/summary.json.