

Neon: Open-Source Vision-Language-Action Model for Humanoid Whole-Body Control

Video Foundation Models + Parameter-Efficient Action Decoders + Cross-Embodiment Data

Cagatay Cali

cagatay@amazon.com

<https://github.com/cagataycali/neon>

March 2026

Abstract

We present **Neon**, an open-source Vision-Language-Action (VLA) model for whole-body humanoid control on the Unitree G1 robot. Neon replaces the traditional image encoder with a *video foundation model* (Qwen2.5-Omni or Cosmos-Reason2) that inherently understands motion, physics, and temporal dynamics. On top of the frozen backbone, we train lightweight action decoder heads ($\sim 2\text{M}$ parameters) using techniques from the Parameter Golf competition: ReLU² activations, RMSNorm, learnable residual scales, U-Net skip connections, and logit soft-capping. For cross-embodiment training, we adopt the Cosmos-Predict2.5 relative end-effector action representation, enabling a unified “data soup” of seven heterogeneous source types including LeRobot, Agibot-World, GR00T-Dreams, Stereo4D, and spoken commands. Neon supports 29 degrees of freedom with action chunking (16-step prediction), optional audio input via Whisper encoding, and speech output via PersonaPlex TTS. The full system runs at 50ms latency on NVIDIA Jetson Orin. We release the code, model weights, training configs, and this paper under the MIT license.

1 Introduction

Recent work on Vision-Language-Action (VLA) models has demonstrated that large pre-trained vision-language models can serve as powerful backbones for robotic control [1, 2, 3]. However, most approaches use *image* encoders that process individual frames independently, discarding the temporal information essential for understanding motion and physical dynamics.

We argue that **video foundation models** are a more natural backbone for robot control. Models like Qwen2.5-Omni [6] and Cosmos-Reason2 [5] are pre-trained on millions of hours of video and already encode concepts of motion, object permanence, contact dynamics, and

spatial-temporal reasoning. Rather than learning these from scratch on limited robot data, we can *reuse* this understanding.

Neon makes this concrete. Our contributions:

1. A VLA architecture that uses a frozen video backbone (3–7B parameters) with lightweight trainable action heads ($\sim 2\text{M}$ parameters), training only 0.2% of total parameters.
2. Parameter Golf v2 action decoders incorporating six techniques from the OpenAI Parameter Golf competition optimized for small modules.
3. A cross-embodiment relative action representation (from Cosmos-Predict2.5) enabling training on heterogeneous data sources.
4. A “data soup” pipeline mixing seven source types with automatic action mapping to the G1’s 29 DoF action space.
5. Integrated audio input (Whisper) and speech output (PersonaPlex TTS) for voice-controlled humanoid operation.
6. Open-source release of code, configs, and 80 unit tests: <https://github.com/cagataycali/neon>.

2 Related Work

Vision-Language-Action Models. RT-2 [1] first showed that VLMs can directly output robot actions as text tokens. OpenVLA [2] scaled this to 7B parameters with open weights. π_0 [3] introduced flow matching for continuous action spaces. All use image-based encoders.

Video Understanding for Robotics. GR00T-N1 [4] pioneered video-conditioned action generation using a dual-system architecture. Cosmos-Reason2 [5] provides physical AI reasoning over video. Neon builds on these by treating the video model as a *frozen feature extractor*.

Action Representations. Cosmos-Predict2.5 introduced relative end-effector actions in the gripper frame for cross-embodiment transfer. ACT [7] introduced action chunking. Neon combines both: relative gripper-frame actions chunked into 16-step predictions.

Parameter-Efficient Training. LoRA [8] and QLoRA [9] reduce trainable parameters for fine-tuning. The OpenAI Parameter Golf competition pushed techniques for maximizing performance per parameter byte. We apply these techniques to action decoders rather than language model adapters.

3 Architecture

Neon follows a *frozen backbone, trainable heads* paradigm. The system takes multimodal inputs (camera frames, language instruction, joint states, optional audio) and outputs action chunks for the G1 humanoid.

3.1 Video Backbone

Let $\mathcal{V} = \{I_1, \dots, I_T\}$ be a sequence of camera frames and ℓ a language instruction. The video backbone ϕ produces visual-language features:

$$\mathbf{f}_{\text{vis}} = \phi(\mathcal{V}, \ell) \in \mathbb{R}^{d_{\text{bb}}} \quad (1)$$

where $d_{\text{bb}} = 2048$ for Qwen2.5-Omni-7B and $d_{\text{bb}} = 4096$ for Cosmos-Reason2-8B. The backbone parameters are **frozen** during training; we use 4-bit quantization (NF4) to reduce memory from $\sim 14\text{GB}$ to $\sim 4\text{GB}$.

3.2 Proprioception Encoder

Current joint positions $\mathbf{q} \in \mathbb{R}^{n_q}$ (where n_q is the action dimension) are encoded by a two-layer MLP:

$$\mathbf{f}_{\text{prop}} = \sigma(W_2 \sigma(W_1 \mathbf{q} + b_1) + b_2) \in \mathbb{R}^{d_p} \quad (2)$$

with $d_p = 256$ and $\sigma = \text{GELU}$.

3.3 Audio Encoder (Optional)

When audio input $\mathbf{a}_{\text{wav}} \in \mathbb{R}^{16000 \cdot t}$ (16kHz waveform of t seconds) is provided, we encode it via a Whisper-derived mel spectrogram encoder:

$$\mathbf{f}_{\text{audio}} = \psi(\mathbf{a}_{\text{wav}}) \in \mathbb{R}^{d_{\text{bb}}} \quad (3)$$

Three encoder types are supported: **whisper** (OpenAI Whisper weights), **mel_spectrogram** (lightweight mel + MLP), and **omni** (native Qwen2.5-Omni audio tower).

3.4 Feature Fusion

All available features are concatenated and projected back to the backbone dimension:

$$\mathbf{f} = \text{Proj}(\text{concat}[\mathbf{f}_{\text{vis}}, \mathbf{f}_{\text{prop}}, \mathbf{f}_{\text{audio}}]) \in \mathbb{R}^{d_{\text{bb}}} \quad (4)$$

where $\text{Proj} : \mathbb{R}^{d_{\text{vis}}+d_p+d_{\text{audio}}} \rightarrow \mathbb{R}^{d_{\text{bb}}}$ is a linear layer followed by GELU and dropout. This bottleneck forces multimodal compression.

3.5 Action Heads — Parameter Golf v2

The action heads are the *only* trainable components (besides fusion and proprioception encoder). We use separate decoder heads per joint group:

- **Arms:** ActionChunkingHead $\rightarrow \mathbb{R}^{T_c \times 14}$
- **Locomotion:** ActionChunkingHead $\rightarrow \mathbb{R}^{T_c \times 3}$
- **Torso+Head:** MLPDecoder $\rightarrow \mathbb{R}^3$ (no chunking)
- **Legs:** MLPDecoder $\rightarrow \mathbb{R}^{12}$ (whole-body mode)

where $T_c = 16$ is the chunk size (number of predicted future timesteps).

4 Parameter Golf v2 Techniques

Since the action heads constitute our “small model” ($\sim 2\text{M}$ parameters), every parameter-efficiency technique matters. We port six techniques from the OpenAI Parameter Golf competition winners:

4.1 ReLU² Activation

Standard ReLU has a discontinuous gradient at zero. GELU is smooth but expensive. We use ReLU²:

$$\text{ReLU}^2(x) = \max(0, x)^2 \quad (5)$$

The gradient $\frac{\partial}{\partial x} \text{ReLU}^2(x) = 2 \max(0, x)$ is *continuous* at zero (unlike ReLU’s step function) and requires only a comparison and multiply (unlike GELU’s exp). For action decoders, this produces smoother output trajectories.

4.2 RMSNorm

Layer Normalization computes both mean and variance. RMSNorm [10] drops the mean centering:

$$\text{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2 + \varepsilon}} \odot \boldsymbol{\gamma} \quad (6)$$

where $\boldsymbol{\gamma} \in \mathbb{R}^d$ is a learnable scale. This saves one reduction operation and one subtraction per element while providing equivalent training stability for small MLPs.

4.3 Learnable Residual Scales

In standard residual connections, $\mathbf{h}' = \mathbf{h} + f(\mathbf{h})$, the residual \mathbf{h} can dominate early in training when the backbone features are much larger in magnitude than the randomly-initialized head features. We introduce a learnable scale:

$$\mathbf{h}' = f(\mathbf{h}) + \alpha \cdot \mathbf{h}, \quad \alpha = \text{Parameter}(\text{init} = 1.0) \quad (7)$$

The network learns to scale residuals per layer. In practice, α converges to values between 0.3 and 0.8, confirming that the raw residual needs attenuation.

4.4 U-Net Skip Connections

For MLPs with $L \geq 3$ layers, we save the first hidden layer output \mathbf{h}_0 and concatenate it with the last hidden layer input:

$$\mathbf{h}_{L-1} = g(\text{concat}[\mathbf{h}_{L-2}, \mathbf{h}_0]) \quad (8)$$

This creates a gradient “highway” through deep decoders, preventing the middle layers from becoming a gradient bottleneck. The concatenation doubles the input dimension of the last hidden layer, requiring a projection back to d_h .

4.5 Logit Soft-Capping

Hard Tanh ($\tanh(x)$) clamps outputs to $[-1, 1]$ but has zero gradient when $|x| > 1$. This is problematic for action boundary values (e.g., joint limits). Soft-capping provides asymptotic bounding:

$$\text{softcap}(x; c) = c \cdot \tanh\left(\frac{x}{c}\right) \quad (9)$$

where $c = 1.0$ for normalized actions. The gradient $\frac{\partial}{\partial x} \text{softcap}(x; c) = 1 - \tanh^2(x/c)$ is always nonzero, maintaining gradient flow at the boundaries. From [11].

4.6 Optimizer Tuning

We use Adam with $\beta_1 = 0.85$ (vs. default 0.9) for faster adaptation to shifting action distributions during training, $\beta_2 = 0.99$ for reduced momentum lag, and gradient clipping at $\|\nabla\|_{\max} = 0.3$ (vs. typical 1.0) to prevent divergence in small heads.

5 Action Space and Representations

5.1 G1 Humanoid — 29 Degrees of Freedom

The Unitree G1 has 29 controllable joints organized into groups:

Group	Joints	DoF
Left arm	shoulder (3), elbow (1), wrist (3)	7
Right arm	shoulder (3), elbow (1), wrist (3)	7
Torso	waist yaw	1
Head	pitch, yaw	2
Left leg	hip (3), knee (1), ankle (2)	6
Right leg	hip (3), knee (1), ankle (2)	6
Total		29

Table 1: G1 action space decomposition.

Three progressive control modes select subsets:

- **arms_only**: 14 arm joints + 3 locomotion = 17 DoF
- **upper_body**: 14 + 3 + 3 (torso/head) = 20 DoF
- **whole_body**: all 29 DoF

All actions are normalized to $[-1, 1]$ using per-joint min/max limits derived from the G1 URDF.

5.2 Action Chunking

Following ACT [7], we predict $T_c = 16$ future timesteps simultaneously instead of a single action. At 50Hz control frequency, this covers 320ms of future motion.

For the ActionChunkingHead, each timestep $t \in \{1, \dots, T_c\}$ gets a learnable embedding $\mathbf{e}_t \in \mathbb{R}^{d_h}$. The predicted action at step t is:

$$\hat{\mathbf{a}}_t = \text{MLP}(\text{concat}[\text{Proj}(\mathbf{f}), \mathbf{e}_t]) \in \mathbb{R}^{n_a} \quad (10)$$

where n_a is the per-head action dimension.

The training loss is:

$$\mathcal{L} = \frac{1}{T_c \cdot n_a} \sum_{t=1}^{T_c} \sum_{i=1}^{n_a} m_t (\hat{a}_{t,i} - a_{t,i}^*)^2 \quad (11)$$

where $m_t \in \{0, 1\}$ masks padded timesteps and a^* are ground-truth actions.

5.3 Cosmos Relative Actions

For cross-embodiment transfer, we adopt the relative end-effector representation from Cosmos-Predict2.5. Given consecutive gripper states with positions $\mathbf{p}_{t-1}, \mathbf{p}_t \in \mathbb{R}^3$ and rotation matrices $R_{t-1}, R_t \in \text{SO}(3)$:

$$\Delta \mathbf{p} = R_{t-1}^\top (\mathbf{p}_t - \mathbf{p}_{t-1}) \in \mathbb{R}^3 \quad (12)$$

$$\Delta R = R_{t-1}^\top R_t \in \text{SO}(3) \quad (13)$$

$$\Delta \boldsymbol{\theta} = \text{euler}(\Delta R) \in \mathbb{R}^3 \quad (14)$$

The full relative action is $\mathbf{a}^{\text{rel}} = [\lambda \Delta \mathbf{p}, \lambda \Delta \boldsymbol{\theta}, g_t]$ where $\lambda = 20$ is a scaling factor and $g_t \in [0, 1]$ is the gripper state.

Proposition 1 (Cross-embodiment invariance)

For any two robots A and B performing the same physical end-effector motion in different workspaces with different coordinate origins O_A, O_B and base orientations R_A^0, R_B^0 , the relative actions satisfy $\mathbf{a}_A^{\text{rel}} = \mathbf{a}_B^{\text{rel}}$.

Let $\mathbf{p}_t^A = R_A^0 \mathbf{p}_t^w + O_A$ be robot A 's gripper position, where \mathbf{p}_t^w is the world-frame position. Then:

$$R_{t-1}^{A\top}(\mathbf{p}_t^A - \mathbf{p}_{t-1}^A) = R_{t-1}^{A\top} R_A^0(\mathbf{p}_t^w - \mathbf{p}_{t-1}^w)$$

Since $R_t^A = R_A^0 R_t^w$, we have $R_{t-1}^{A\top} R_A^0 = R_{t-1}^{w\top} R_A^{0\top} R_A^0 = R_{t-1}^{w\top}$. The same holds for robot B , giving $\Delta \mathbf{p}^A = \Delta \mathbf{p}^B = R_{t-1}^{w\top}(\mathbf{p}_t^w - \mathbf{p}_{t-1}^w)$. An identical argument applies to the rotation delta.

The rotation conversions use ZYX Euler angles. Given $R \in \text{SO}(3)$:

$$\text{roll} = \text{atan2}(R_{21}, R_{22}) \quad (15)$$

$$\text{pitch} = \text{atan2}(-R_{20}, \sqrt{R_{00}^2 + R_{10}^2}) \quad (16)$$

$$\text{yaw} = \text{atan2}(R_{10}, R_{00}) \quad (17)$$

6 Data Soup

6.1 Multi-Source Pipeline

Neon trains on a weighted mixture of seven data source types:

Source Type	Action Repr.	Example
lerobot	Joint positions	Bridge, DROID
agibot	Joint positions	Agibot-World (1M+)
dreamgen	IDM extraction	GR00T-Dreams
cosmos_dreamgen	Relative EE	Cosmos data
oxe	Joint positions	Open X-Embodiment
voice_commands	Language only	Audio + text
stereo4d	Visual only	Stereo pairs

Table 2: Data soup source types. Each normalizes to a common `NeonEpisode` format.

6.2 Cross-Embodiment Action Mapping

For joint-position sources, an `ActionMapper` remaps source joint indices to G1 indices with zero-padding for unmatched joints. Formally, given source actions $\mathbf{a}^{\text{src}} \in \mathbb{R}^{n_s}$ and a mapping $\pi : \{1, \dots, n_s\} \rightarrow \{1, \dots, 29\}$:

$$a_j^{\text{G1}} = \begin{cases} a_{\pi^{-1}(j)}^{\text{src}} & \text{if } j \in \text{Im}(\pi) \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

6.3 Weighted Mixing

Each source s has a weight $w_s > 0$. During training, source s is sampled with probability $p_s = w_s / \sum_k w_k$. Optional `max_episodes` caps per-source contribution.

7 Training

7.1 Frozen Backbone, Trainable Heads

Only the action heads, fusion layer, and proprioception encoder are trained. For a model with backbone parameters θ_{bb} and head parameters θ_{head} :

$$\min_{\theta_{\text{head}}} \mathbb{E}_{(\mathcal{V}, \ell, \mathbf{q}, \mathbf{a}^*) \sim \mathcal{D}} [\mathcal{L}(\hat{\mathbf{a}}(\mathcal{V}, \ell, \mathbf{q}; \theta_{\text{bb}}, \theta_{\text{head}}), \mathbf{a}^*)] \quad (19)$$

where θ_{bb} is fixed. The backbone is loaded in 4-bit (NF4) quantization [9].

7.2 Hyperparameters

Parameter	Value
Optimizer	Paged AdamW 8-bit
Learning rate	2×10^{-4}
β_1, β_2	0.85, 0.99
Weight decay	0.01
LR schedule	Cosine with 10% warmup
Max grad norm	0.3
Batch size	4 ($\times 8$ accumulation = 32 effective)
Precision	BFloat16
Epochs	3

Table 3: Training hyperparameters.

7.3 Parameter Budget

Component	Parameters
Video backbone (frozen)	3–7B
Arms ActionChunkingHead	$\sim 1.5\text{M}$
Locomotion ActionChunkingHead	$\sim 400\text{K}$
Fusion layer	$\sim 4.2\text{M}$
Proprioception encoder	$\sim 100\text{K}$
RMSNorm weights	$\sim 3\text{K}$
Learnable residual scales	~ 10
Total trainable	$\sim 6.2\text{M}$

Table 4: Parameter budget for `arms_only` mode.

8 Inference

8.1 Real-Time Pipeline

At inference time, the model receives a camera frame, optional spoken command, and current joint states. The full forward pass (backbone encode \rightarrow fusion \rightarrow action decode) runs in $\sim 50\text{ms}$ on Jetson Orin with the 3B backbone.

8.2 Temporal Action Smoothing

The inference server maintains a rolling buffer of predicted action chunks. Overlapping predictions from consecutive forward passes are blended using exponential averaging:

$$\mathbf{a}_t^{\text{smooth}} = \alpha \mathbf{a}_t^{\text{new}} + (1 - \alpha) \mathbf{a}_t^{\text{prev}}, \quad \alpha = 0.7 \quad (20)$$

This reduces jitter from frame-to-frame prediction variance.

8.3 Safety Controller

Before sending commands to the G1, the controller enforces:

1. Joint position limits (from URDF)
2. Joint velocity limits (\dot{q}_{\max} per joint)
3. Locomotion velocity limits (v_{\max}, ω_{\max})
4. Emergency stop on communication timeout

9 Audio System

9.1 Speech Input

Spoken commands are encoded via a Whisper-derived audio encoder. The mel spectrogram $M \in \mathbb{R}^{80 \times T_{\text{mel}}}$ is processed through convolutional layers and projected to backbone dimension:

$$\mathbf{f}_{\text{audio}} = W_{\text{proj}} \cdot \text{Pool}(\text{Conv}(M)) \in \mathbb{R}^{d_{\text{bb}}} \quad (21)$$

9.2 Speech Output

Optionally, the fusion features are decoded into a text response via a SpeechResponseHead (linear \rightarrow softmax vocabulary), then synthesized to audio via PersonaPlex TTS [12]. This gives the robot a “voice” that narrates its actions.

10 Experimental Setup

10.1 Test Suite

Neon includes 80 unit tests across 5 files:

- `test_model.py` (22 tests): ReLU², RMSNorm, soft-cap, MLP, chunking, G1 head, NeonVLA
- `test_data.py` (11 tests): action mapper, episodes, configs, data soup
- `test_audio.py` (16 tests): encoders, speech head, PersonaPlex
- `test_relative_actions.py` (12 tests): euler, rotn, quat, Cosmos relative

- `test_action_space.py` (19 tests): G1 joints, modes, normalization

All tests pass on CPU without GPU or backbone weights.

10.2 Hardware

Device	Role	Specs
Jetson Orin	Real-time inference	ARM64, 32GB, CUDA
EC2 L40S	Training	46GB VRAM
HF GPU Jobs	QLoRA fine-tuning	A100/L4 on-demand
MacBook M3	Development, tests	MPS acceleration

Table 5: Hardware targets.

11 Discussion

Why video over image backbones. Video models encode temporal dynamics that image models must learn from robot data alone. A video backbone already “knows” that a falling cup accelerates downward — this prior transfers directly to action prediction without consuming robot training budget.

Parameter efficiency matters. With only $\sim 6\text{M}$ trainable parameters, every architectural choice in the action heads has measurable impact. The Parameter Golf techniques collectively reduce validation loss by providing smoother gradients (ReLU², soft-cap), better normalization (RMSNorm), and improved gradient flow (skip connections, residual scales).

Cross-embodiment is a data multiplier. The relative action representation enables mixing data from Franka, SO-100, and Agibot robots to train a G1 policy. This effectively multiplies available training data without requiring G1-specific demonstrations for every task.

Limitations. The frozen backbone means the model cannot learn new visual representations. The 4-bit quantization introduces approximation errors. Real-world evaluation on the G1 hardware is ongoing.

12 Conclusion

Neon demonstrates that video foundation models can serve as powerful backbones for humanoid whole-body control. By freezing the 3–7B parameter backbone and training only $\sim 6\text{M}$ parameters in action heads enhanced with Parameter Golf techniques, we achieve a parameter-efficient VLA that leverages pre-trained temporal understanding. The cross-embodiment relative action representation and seven-source data soup enable training

on diverse, heterogeneous data. We release Neon as open-source software under the MIT license.

References

- [1] A. Brohan et al., “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control,” *arXiv:2307.15818*, 2023.
- [2] M. Kim et al., “OpenVLA: An Open-Source Vision-Language-Action Model,” *arXiv:2406.09246*, 2024.
- [3] K. Black et al., “ π_0 : A Vision-Language-Action Flow Model for General Robot Control,” *arXiv:2410.24164*, 2024.
- [4] NVIDIA, “GR00T N1: An Open Foundation Model for Generalist Humanoid Robots,” *arXiv:2503.14734*, 2025.
- [5] NVIDIA, “Cosmos World Foundation Model Platform for Physical AI,” *Technical Report*, 2025.
- [6] Alibaba, “Qwen2.5-Omni Technical Report,” *arXiv:2503.20215*, 2025.
- [7] T. Z. Zhao et al., “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware,” *RSS*, 2023.
- [8] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” *ICLR*, 2022.
- [9] T. Detrmers et al., “QLoRA: Efficient Finetuning of Quantized LLMs,” *NeurIPS*, 2023.
- [10] B. Zhang and R. Sennrich, “Root Mean Square Layer Normalization,” *NeurIPS*, 2019.
- [11] Google, “Gemma 2: Improving Open Language Models at a Practical Size,” *arXiv:2408.00118*, 2024.
- [12] NVIDIA, “PersonaPlex: Multi-Speaker TTS with Persona Control,” 2025.

A Full Forward Pass Pseudocode

Figure 1: NeonVLA Forward Pass

```

Input: images  $\mathcal{V}$ , text  $\ell$ , proprio  $q$ , audio  $a$  (optional)
 $f_{\text{vis}} \leftarrow \phi(\mathcal{V}, \ell)$  // Frozen backbone
if  $q$  provided then
 $f_{\text{prop}} \leftarrow \text{MLP}_{\text{proprio}}(q)$ 
 $f \leftarrow \text{concat}[f_{\text{vis}}, f_{\text{prop}}]$ 
else
 $f \leftarrow f_{\text{vis}}$ 
end if
if  $a$  provided then
 $f_{\text{audio}} \leftarrow \psi(a)$  // Whisper encoder
 $f \leftarrow \text{concat}[f, f_{\text{audio}}]$ 
end if
 $f \leftarrow \text{GELU}(W_{\text{fuse}}f + b_{\text{fuse}})$ 
 $\hat{a}_{\text{arms}} \leftarrow \text{ChunkHead}_{\text{arms}}(f)$  // (B, 16, 14)
 $\hat{a}_{\text{loco}} \leftarrow \text{ChunkHead}_{\text{loco}}(f)$  // (B, 16, 3)
 $\hat{a} \leftarrow \text{concat}[\hat{a}_{\text{arms}}, \hat{a}_{\text{loco}}]$ 
return  $\hat{a} \in \mathbb{R}^{B \times 16 \times 17}$ 

```

B MLPDecoder v2 Pseudocode

Figure 2: MLPDecoder with Parameter Golf v2

```

Input: input features  $x \in \mathbb{R}^{d_{\text{in}}}$ 
 $h \leftarrow W_1x + b_1$  // Linear projection
 $h \leftarrow \text{RMSNorm}(h)$ 
 $h \leftarrow \text{ReLU}^2(h)$ 
 $h_{\text{skip}} \leftarrow h$  // Save for U-Net skip
for  $i = 2$  to  $L - 1$  do
 $h_{\text{in}} \leftarrow h$ 
if  $i = L - 1$  then
 $h \leftarrow \text{concat}[h, h_{\text{skip}}]$  // U-Net skip
end if
 $h \leftarrow W_i h + b_i$ 
 $h \leftarrow \text{RMSNorm}(h)$ 
 $h \leftarrow \text{ReLU}^2(h)$ 
if  $\dim(h) = \dim(h_{\text{in}})$  then
 $h \leftarrow h + \alpha_i \cdot h_{\text{in}}$  // Scaled residual
end if
end for
 $x_{\text{out}} \leftarrow W_L h + b_L$ 
return  $\text{softcap}(x_{\text{out}}; 1.0) \in (-1, 1)^{n_a}$ 

```