Мультимодальные и поведенческие модели

Архитектуры Visual Language Action (VLA) моделей

MIPT VLA 2026 | 14 сентября 2026

Алексей Староверов



МФТИ

Center for Cognitive Modeling

План лекции

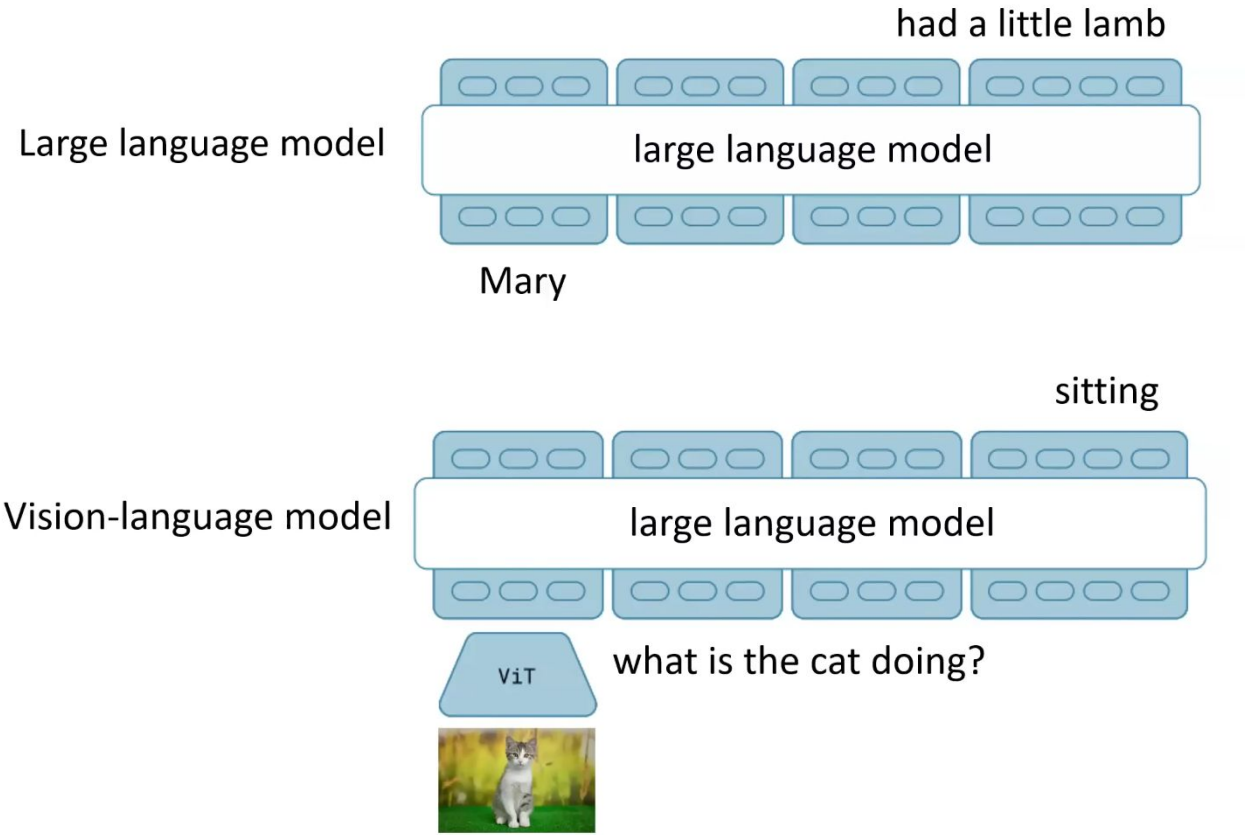
From LLM to Diffusion VLA

Diffusion VLA in reinforcement learning

Architecture details of Diffusion VLA

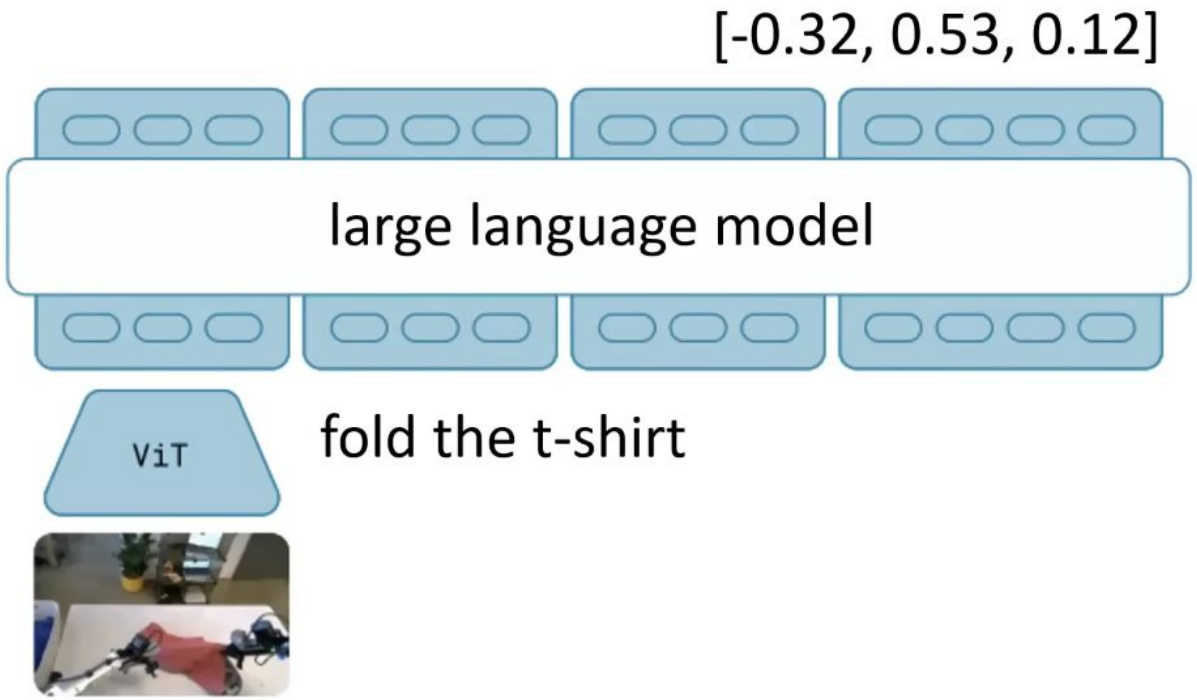
From LLM to Diffusion VLA

From LLM to VLM



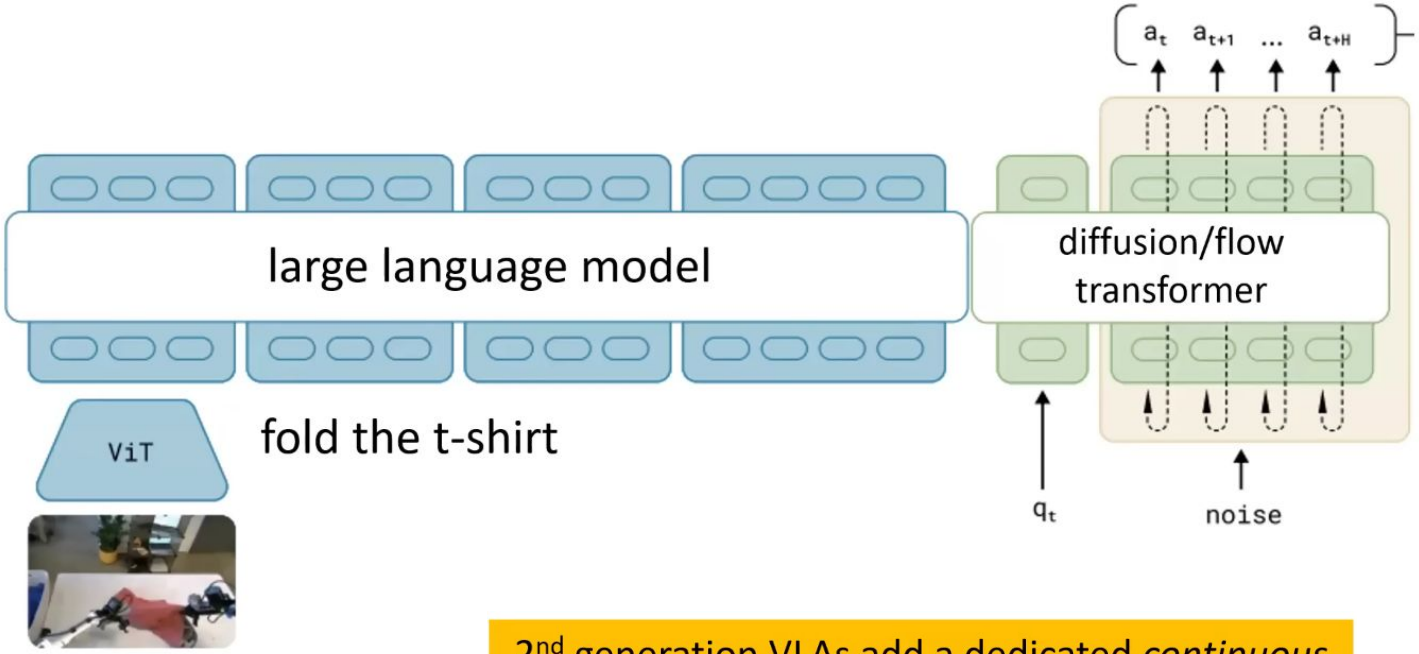
From VLM to VLA (OpenVLA, RT-2)

1st generation VLA



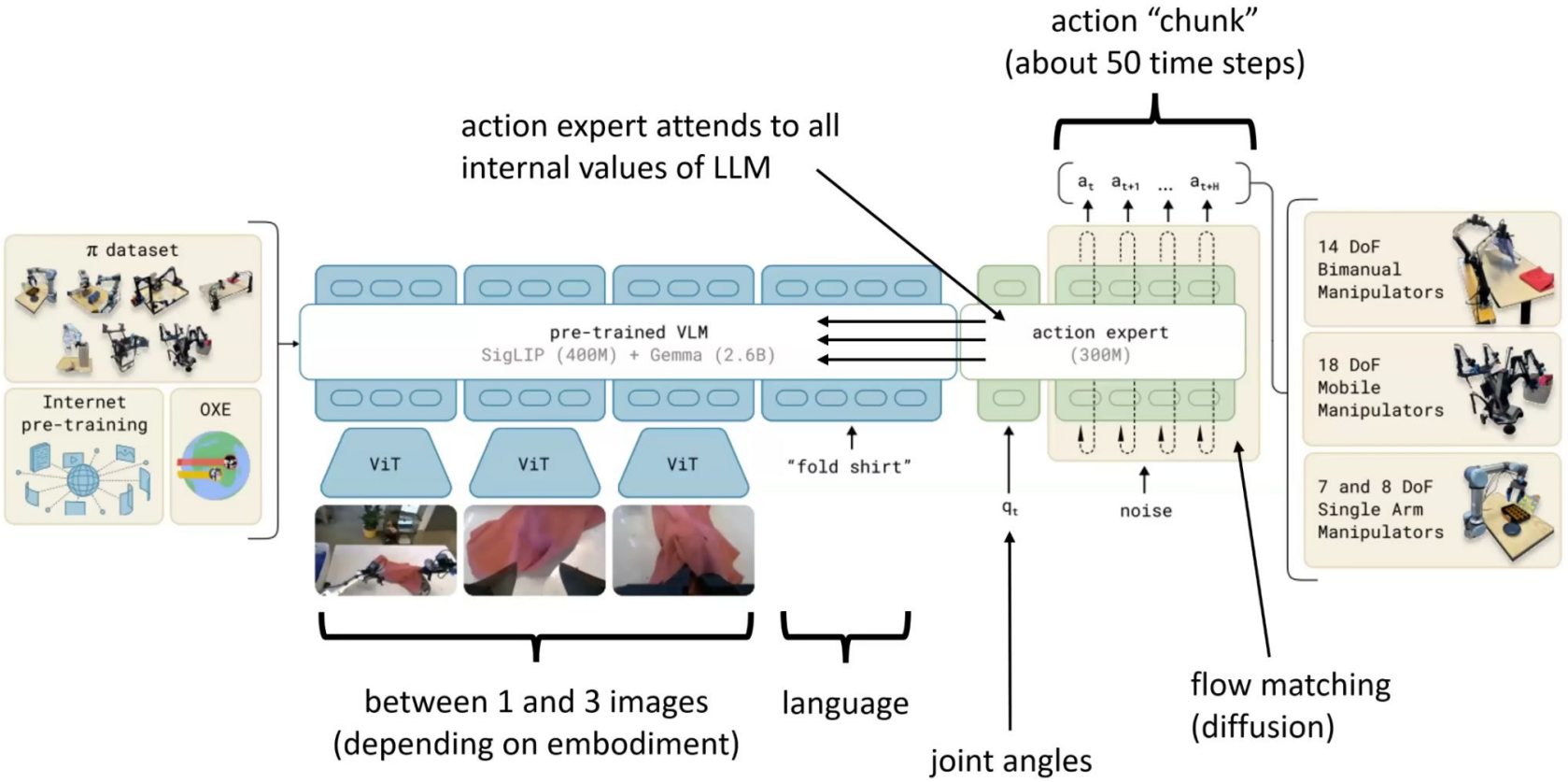
From VLM to VLA (Pi_0)

2nd generation VLA

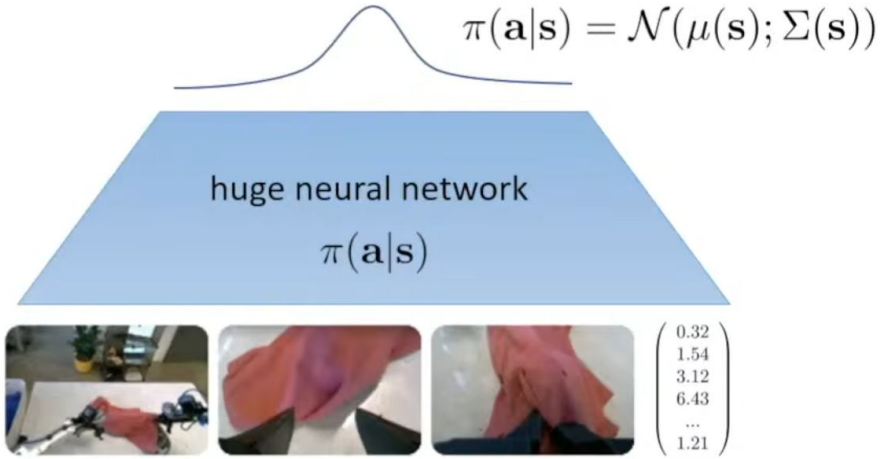


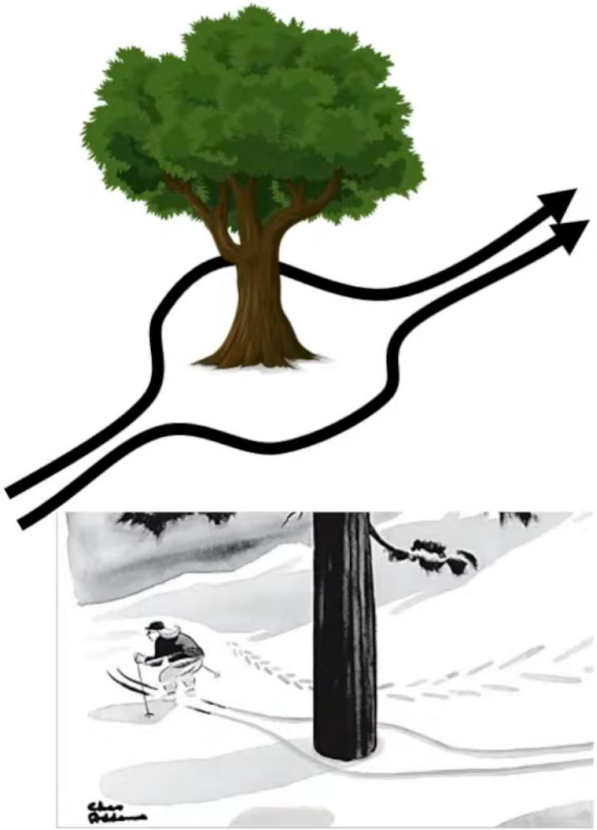
2nd generation VLAs add a dedicated *continuous* output mechanism that can specialize for motor control, like a “virtual motor cortex”

From VLM to VLA (Pi_0)



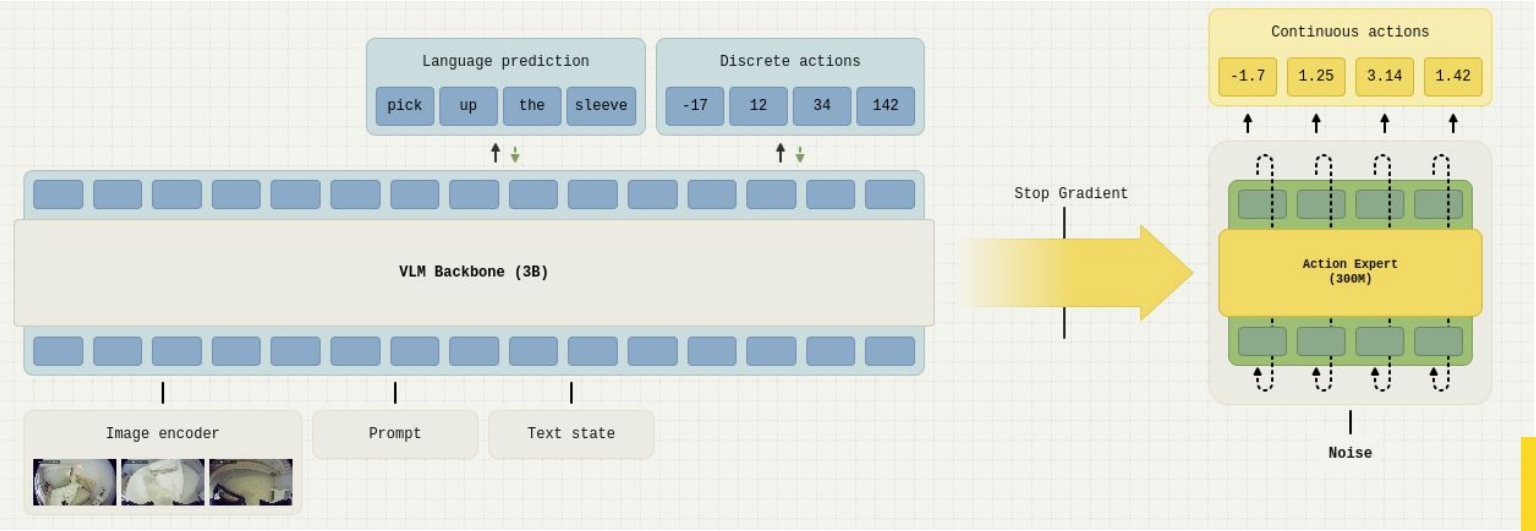
(pros) Diffusion as a policy representation

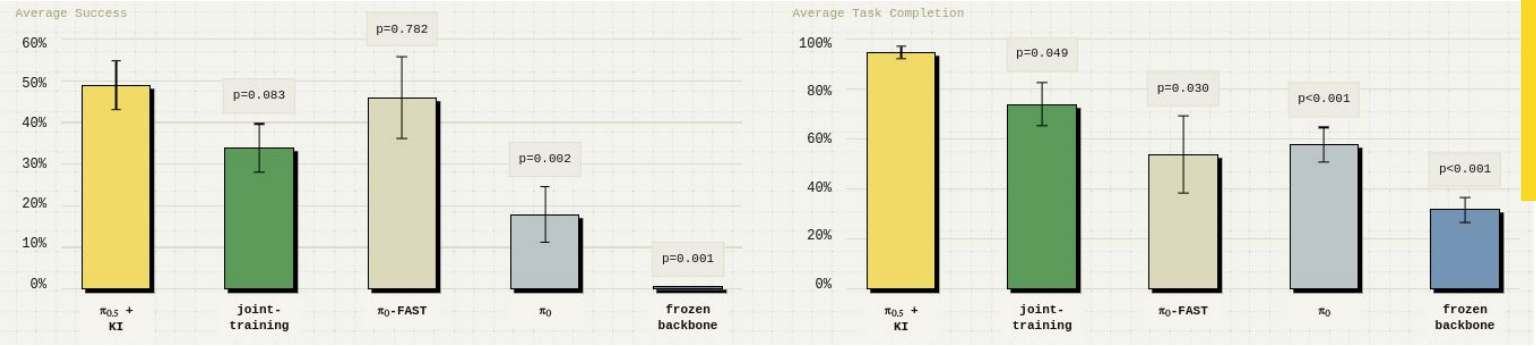




An optimal policy could be Gaussian, but you might not be able to discover it using the limited data

(cons) Knowledge Insulation





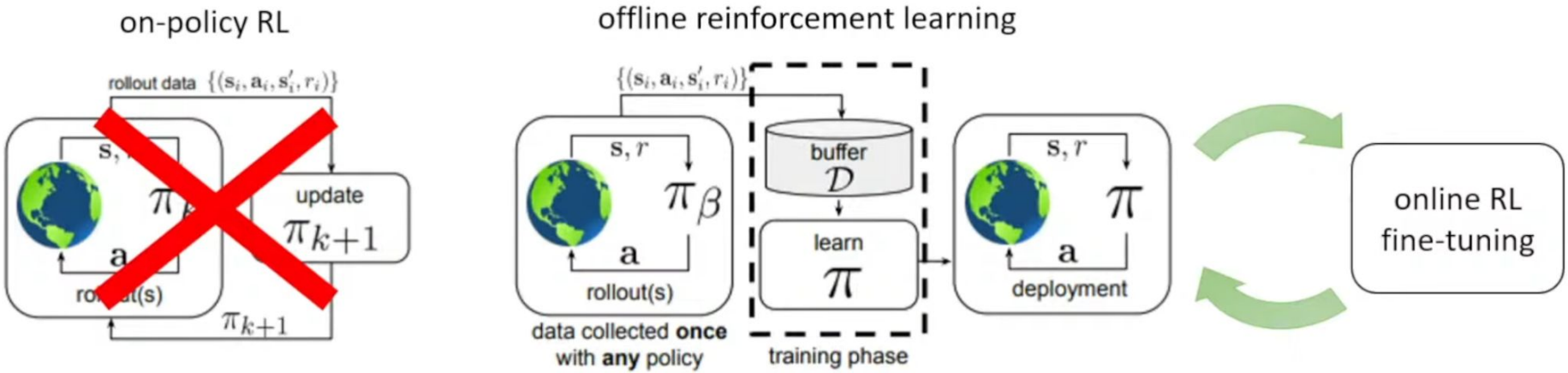
Diffusion loss is 5-10x times slower to train

Action expert designs can interfere with the pre-trained VLM backbone

Knowledge insulation stops the gradient from the action expert & trains the backbone with FAST actions & web data

Diffusion in reinforcement learning

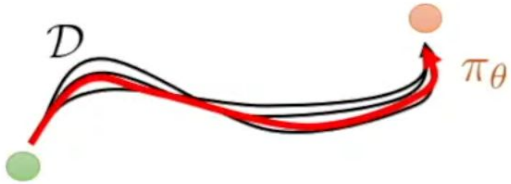
Offline reinforcement learning



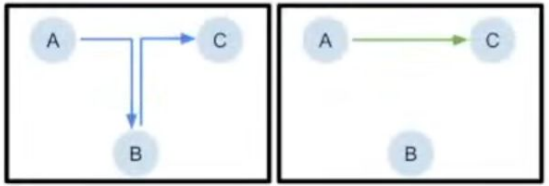
You wanna start from the best you've got and then make it even better!

What does offline RL do

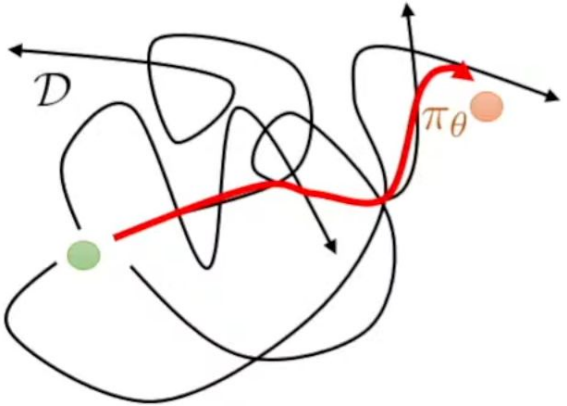
Bad intuition: it's like imitation learning



Better intuition: get order from chaos



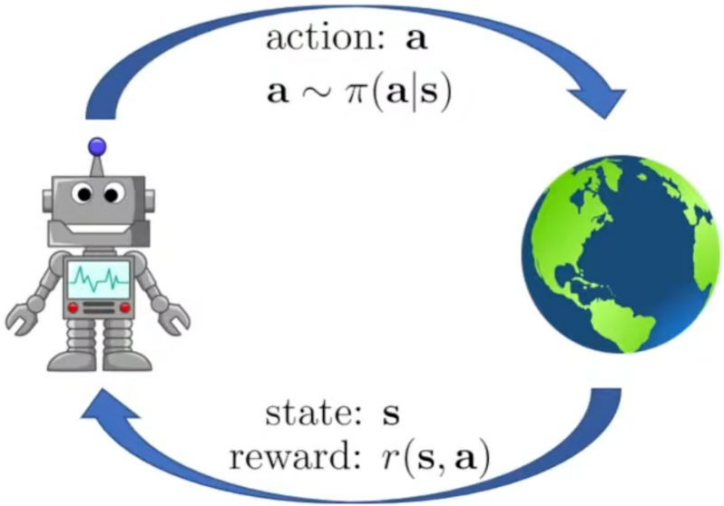
"Macro-scale" stitching



But this is just the clearest example!

Ideally, we "pre-train" with offline RL, and then run online RL!

A quick primer: RL & Q-functions



$$\text{RL objective: } \max_{\pi} \sum_{t=1}^{T} E_{\mathbf{s}_t, \mathbf{a}_t \sim \pi}[r(\mathbf{s}_t, \mathbf{a}_t)]$$

$$\text{Q-function: } Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\mathbf{s}_{t'}, \mathbf{a}_{t'} \sim \pi}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

$$\pi(\mathbf{a}|\mathbf{s}) = 1 \text{ if } \mathbf{a} = \arg\max_{\mathbf{a}} Q^{\pi}(\mathbf{s}, \mathbf{a})$$ policy iteration

$$Q^{\star}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q^{\star}(\mathbf{s}', \mathbf{a}')$$ Bellman optimality equation

enforce this equation at all states!

$$\text{minimize } \textstyle\sum_i (Q(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \max_{\mathbf{a}'_i} Q(\mathbf{s}'_i, \mathbf{a}'_i)])^2$$

You can do it Off-policy and Offline and then get more online data

Distributional shift in value-based RL

Q-learning:

Bellman optimality equation

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$

actor-critic:

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + E_{\mathbf{a}' \sim \pi}[Q(\mathbf{s}', \mathbf{a}')]$$

$$\pi = \arg\max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]$$

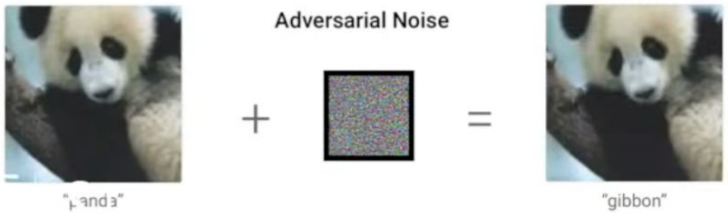
what is the objective?

$$\min_{Q} E_{(\mathbf{s},\mathbf{a}) \sim \pi_\beta(\mathbf{s},\mathbf{a})} \left[(Q(\mathbf{s}, \mathbf{a}) - y(\mathbf{s}, \mathbf{a}))^2 \right]$$

behavior policy

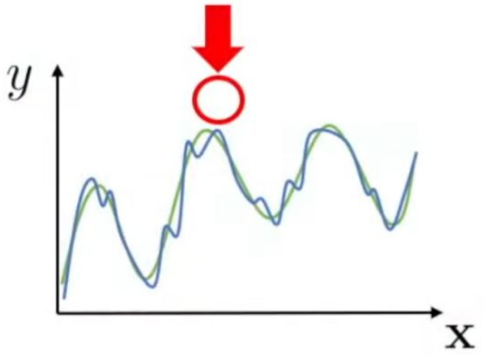
target value

we have *distributional shift* when $\pi_\beta(\mathbf{a}|\mathbf{s}) \neq \pi(\mathbf{a}|\mathbf{s})$



This is **just like** what we saw before in the simple case, just at **every time step** when learning the Q-function!

Conservative Q-learning & pessimism



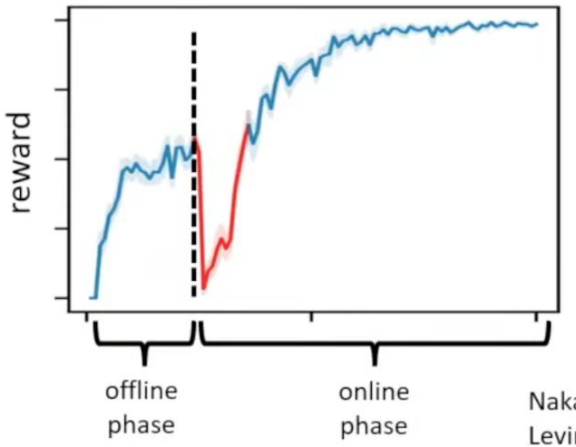
$$\hat{Q}^{\pi} = \arg\min_{Q} \max_{\pi} \alpha E_{\mathbf{s}\sim D, \mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{\mathbf{s},\mathbf{a}\sim D}[Q(\mathbf{s},\mathbf{a})]$$

term to push down big Q-values

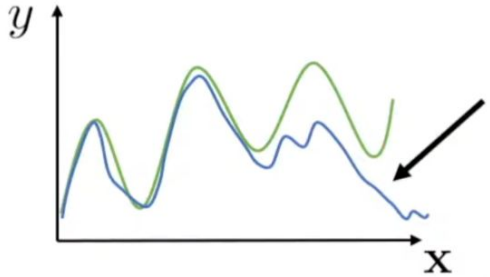
regular objective $+E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_{\pi}[Q(\mathbf{s}',\mathbf{a}')])\right)^2\right]$

If you don’t know the Q-value of some unknown action, just assume that it’s bad

Adding online exploration



why does this happen?

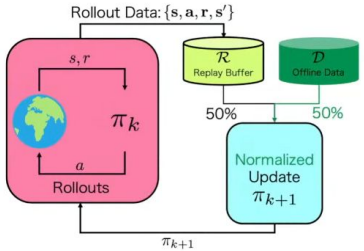


Nakamoto, Zhai, Singh, Mark, Ma, Finn, Kumar, Levine. **Cal-QL: Calibrated Offline RL Pre-Training for Efficient Online Fine-Tuning.** 2023

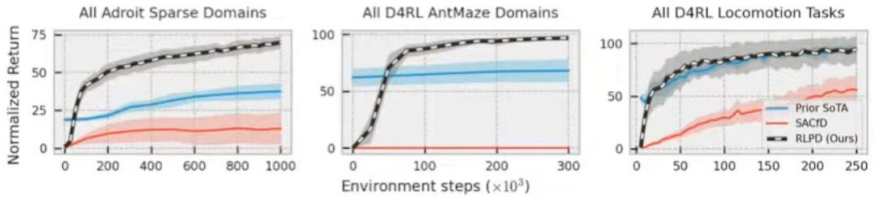
It's very hard to figure out a method that trains value functions **both offline and online**

conservative value functions have high OOD error

when we start collecting online data, these points might no longer be OOD!



Simple but strong baseline (RLPD) to that problem



Another class of offline RL methods

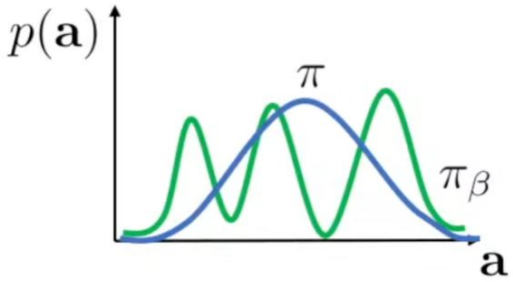
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + E_{\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}')]$$

$$\pi = \arg\max_\pi E_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \;\text{ s.t. }\; D(\pi,\pi_\beta) \leq \epsilon$$

"policy constraint method"

$$\min_Q E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[(Q(\mathbf{s},\mathbf{a}) - y(\mathbf{s},\mathbf{a}))^2\right]$$

Problem: it can be very hard to make this work while avoiding "adversarial" actions



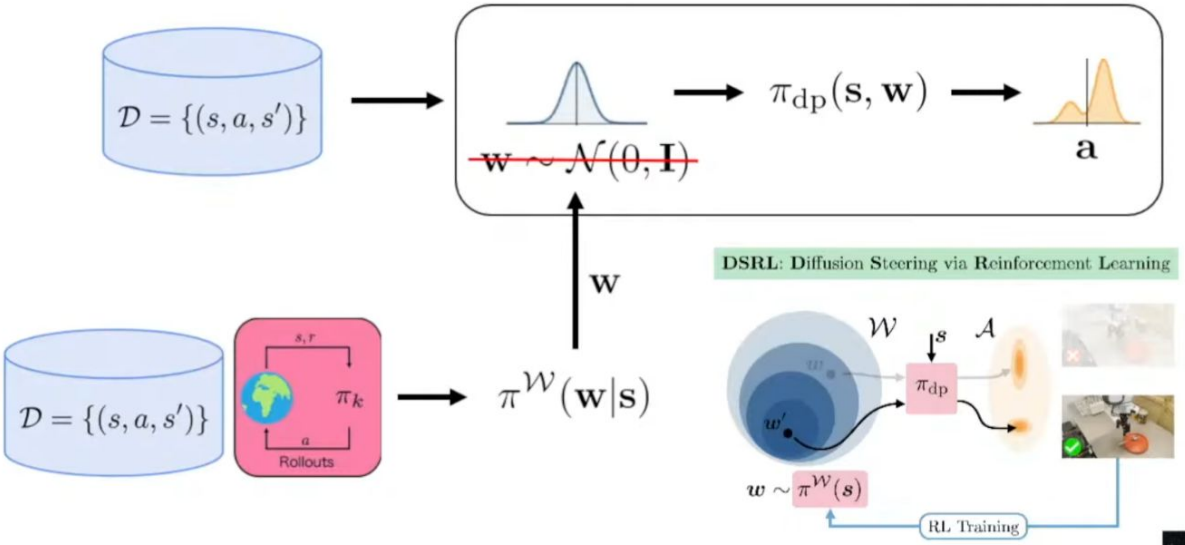
(surprising) key idea: use a very powerful policy class (diffusion, flow matching)

problem: RL with diffusion model actors is difficult
because it's hard to get gradients w.r.t. model likelihoods

If we can train very expressive actors (e.g., diffusion), policy constraints can work well offline and online

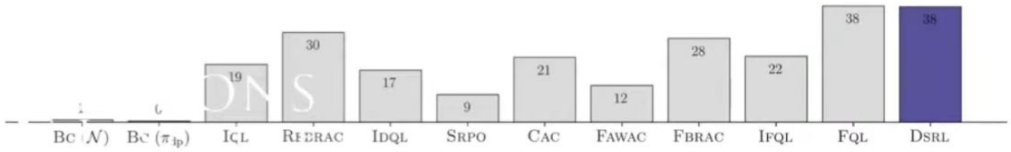
Diffusion steering

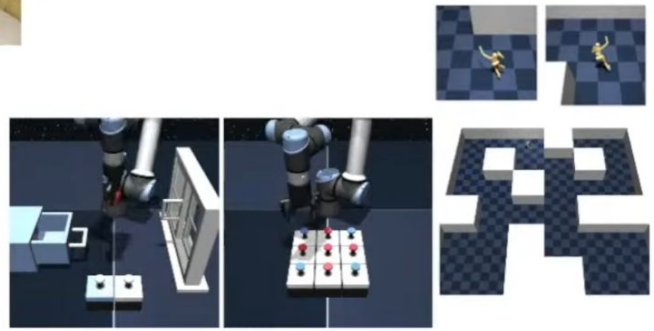
start by training a diffusion policy on **offline data** with **imitation learning**









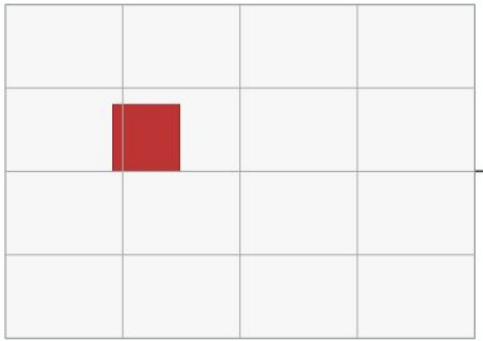


Architecture details of VLA

A quick primer: Transformer Tokens

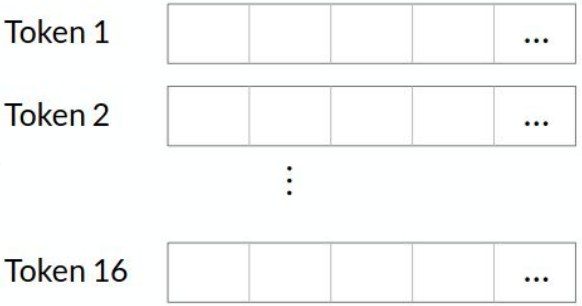
A token is represented by a vector of features.

E.g., 4 × 4 image patches



Flatten each patch
+ learned projection

One vector per patch



16 tokens × d features

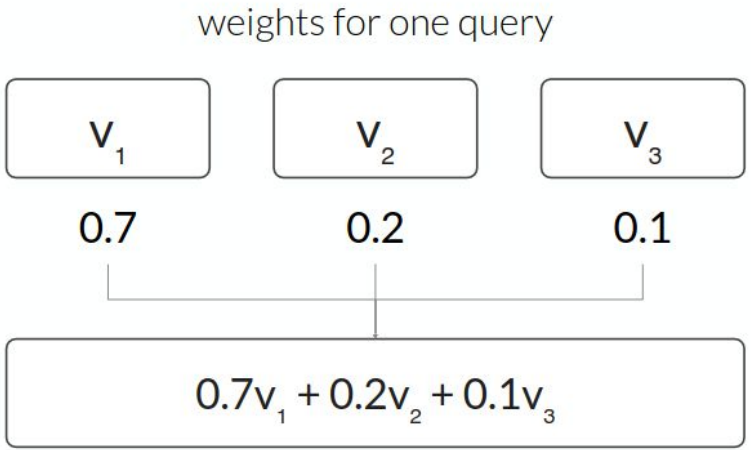
- Add position information so the model knows where each patch came from.
- Text tokens and joint-state vectors can also become token representations.

A quick primer: Transformer Attention

Each token combines information from other tokens, with input-dependent weights.

Query–key similarities → softmax weights → weighted sum of values

	Role
Query **q**	What information would help?
Key **k**	Describes this token for matching
Value **v**	What information gets combined?

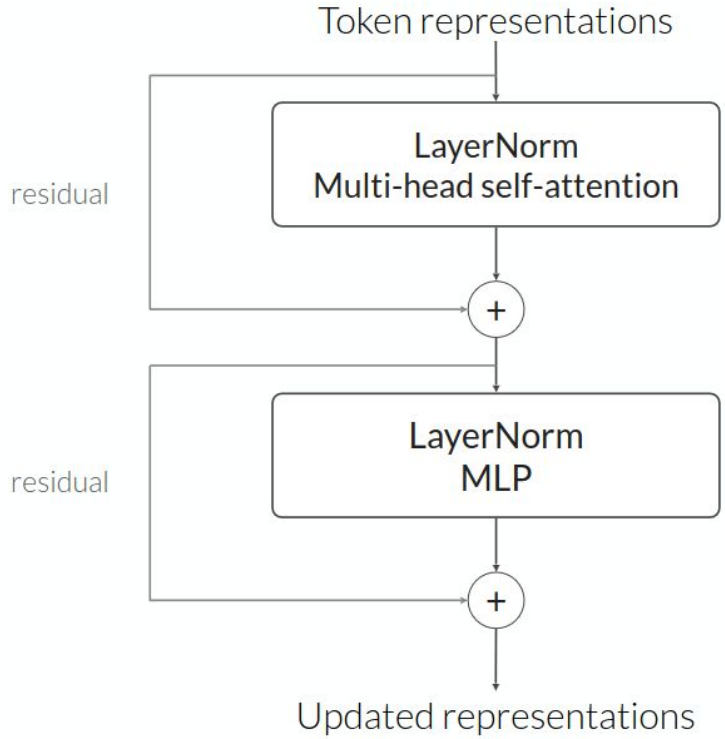


Queries, keys, and values come from learned linear projections.

Attention weights can change while the trained model parameters stay fixed.

A quick primer: Transformer Block

One common arrangement (pre-norm)



Multi-head attention

Several attention computations exchange information across tokens.

MLP

Transforms each token's features using shared weights.

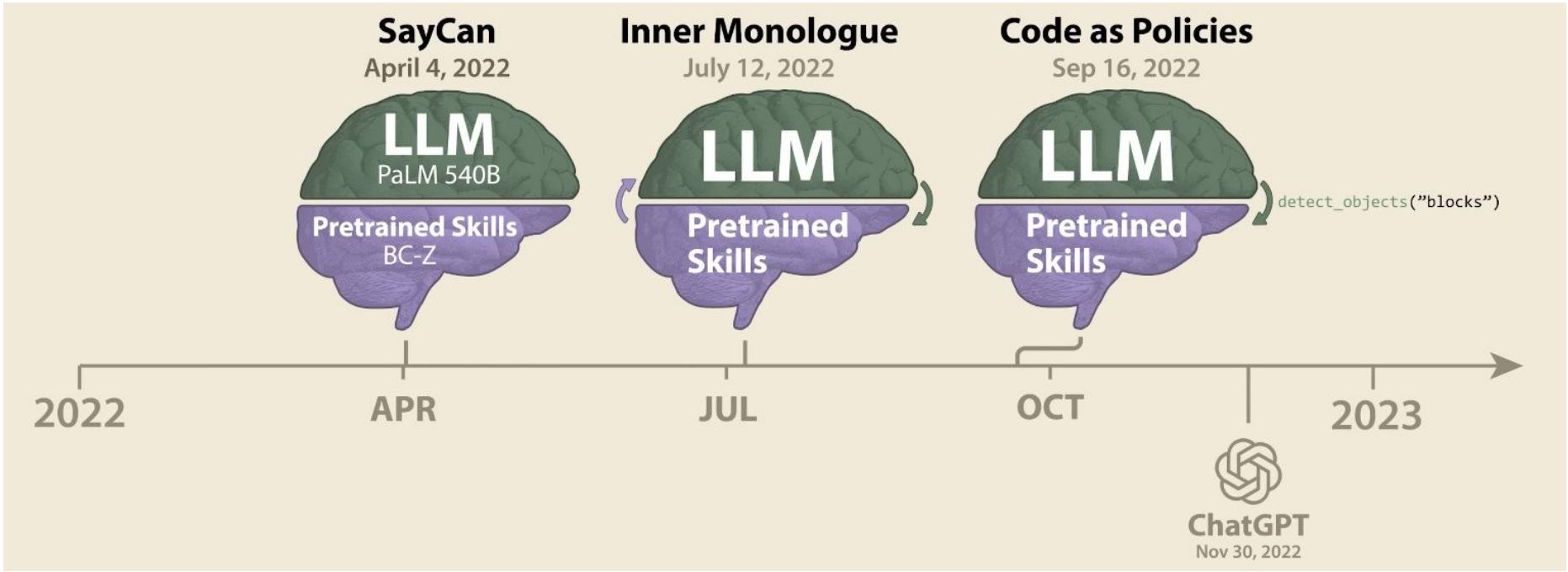
Residual connections

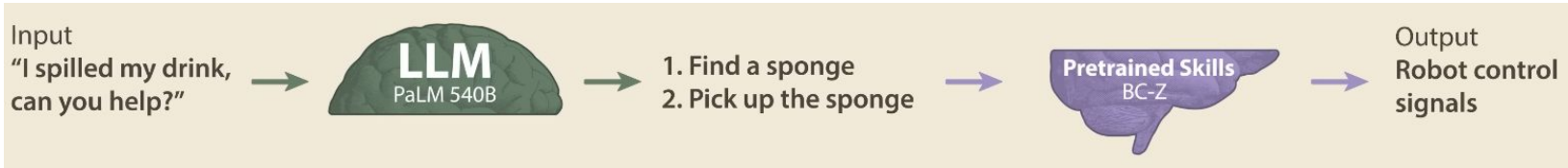
Add the incoming representation back.

LayerNorm

Normalizes features within each token.

Why transformer model as a policy?





Why transformer model as a policy?

Input
"I spilled my drink, can you help?"

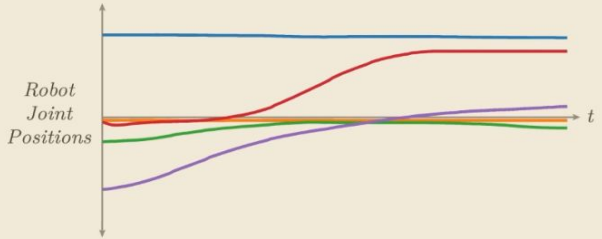
LLM
PaLM 540B

1. Find a sponge
2. Pick up the sponge
3. Go to the spill
4. Put down the sponge

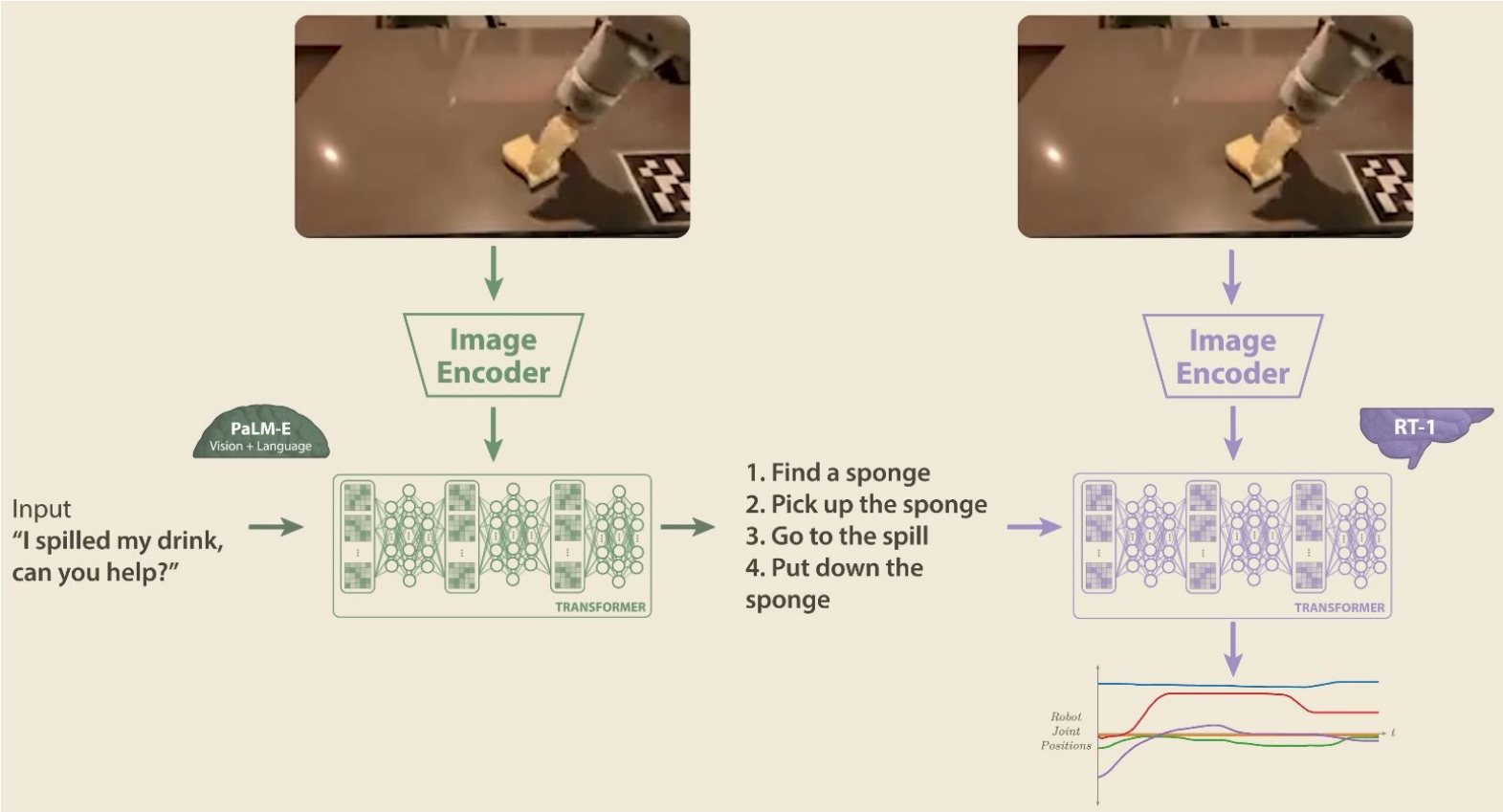


RT-1

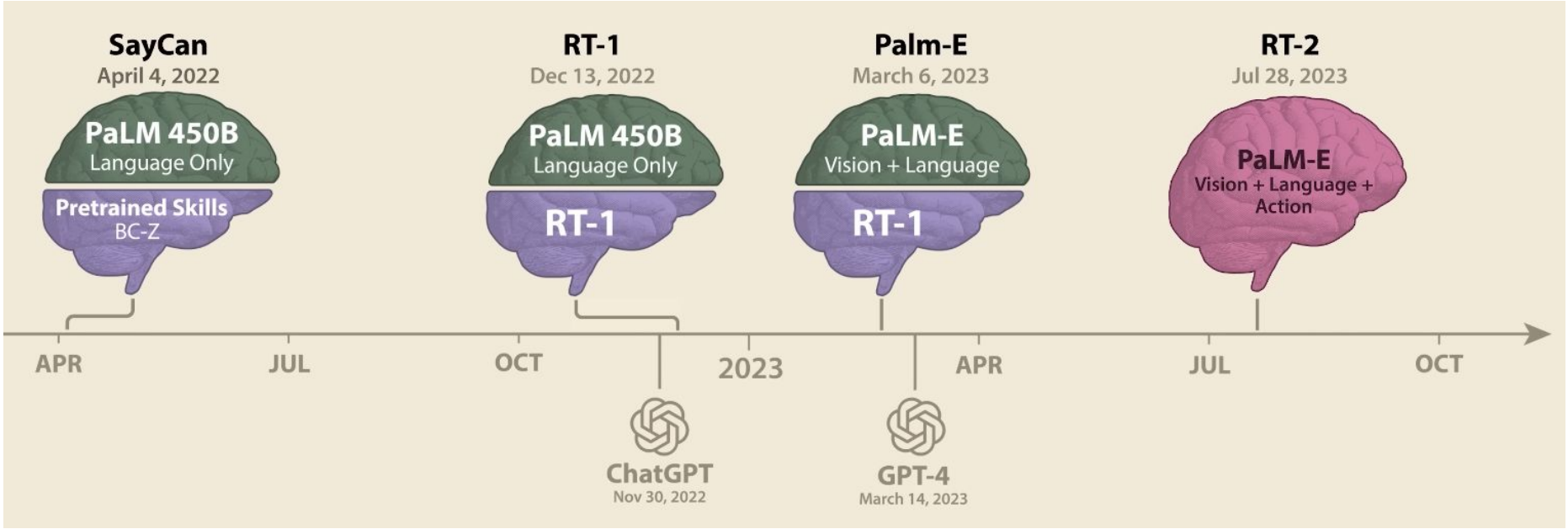
Output
Robot control signals



Why transformer model as a policy?



Why transformer model as a policy?



How to optimize transformer models for a robotic task?

RT-2



July 28, 2023

robotics-transformer2.github.io

- Vision Langauge Action (VLA) Model
- 5-55 Billion Parameters
- Multi-TPU Cluster
- 200-1000 ms inference time

π_0

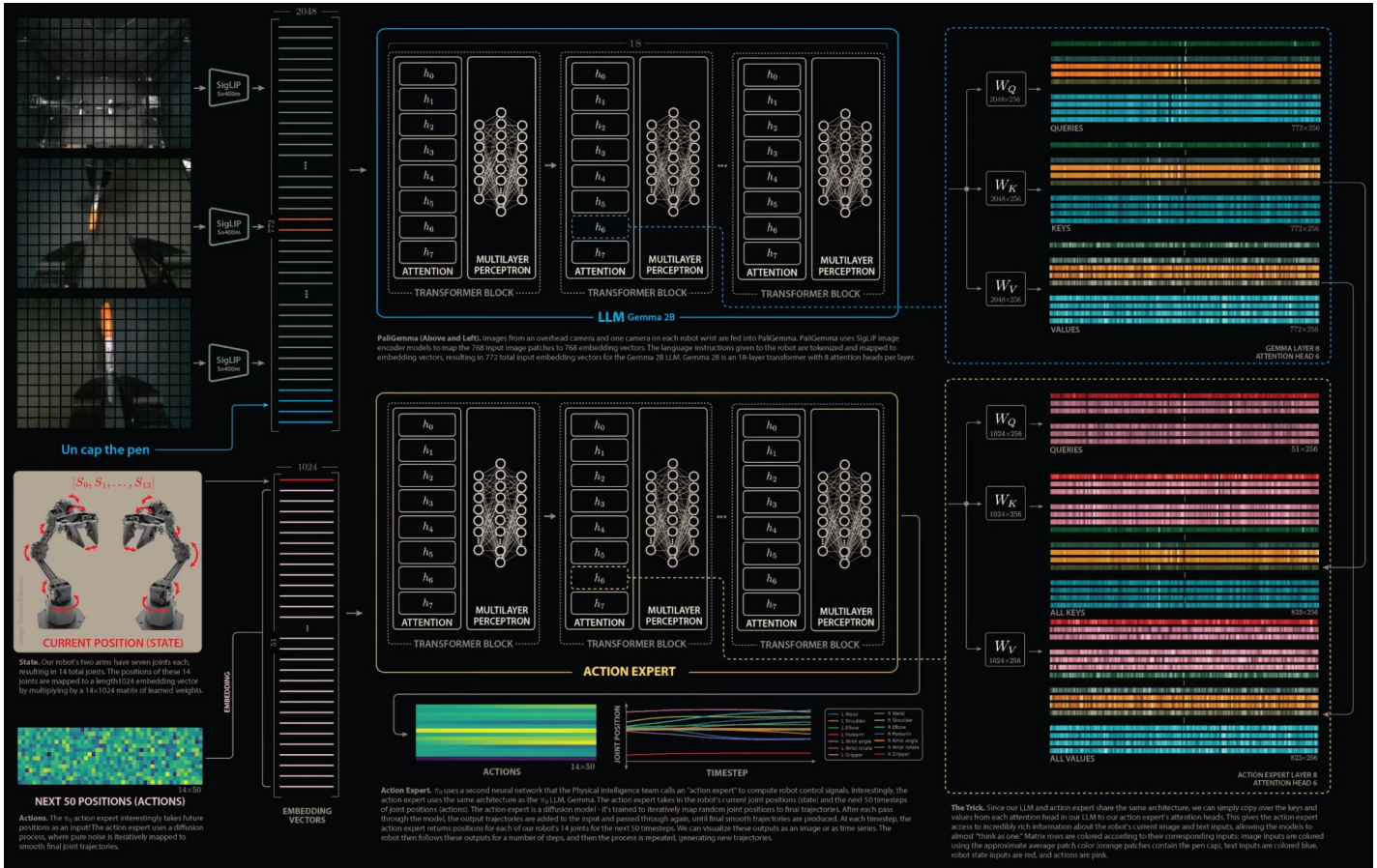


October 31, 2024

pi.website/blog/pi0

- Vision Langauge Action (VLA) Model
- 3.3 Billion Parameters
- Nvidia RTX 4090 GPU
- 73 ms inference time

Overview of Pi0 Architecture



State. Our robot's two arms have seven joints each, resulting in 14 total joints. The positions of these 14 joints are mapped to a length1024 embedding vector by multiplying by a 14×1024 matrix of learned weights.

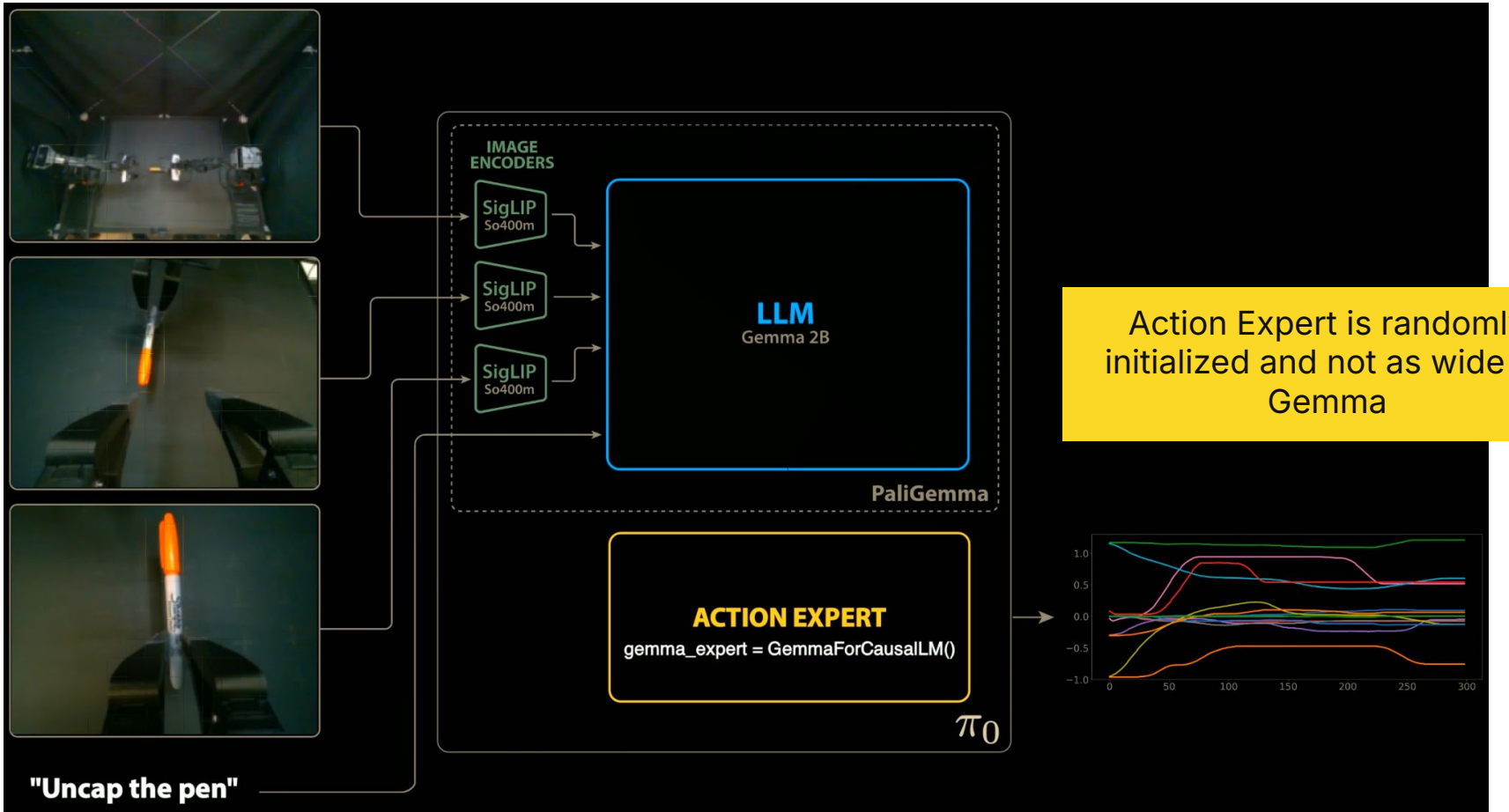
Actions. The π_0 action expert interestingly takes future positions as an input! The action expert uses a diffusion process, where pure noise is iteratively mapped to smooth final joint trajectories.

PaliGemma (Above and Left). Images from an overhead camera and one camera on each robot wrist are fed into PaliGemma. PaliGemma uses SigLIP image encoder models to map the 768 input image patches to 768 embedding vectors. The language instructions given to the robot are tokenized and mapped to embedding vectors, resulting in 772 total input embedding vectors for the Gemma 2B LLM. Gemma 2B is an 18-layer transformer with 8 attention heads per layer.

Action Expert. π_0 uses a second neural network that the Physical Intelligence team calls an "action expert" to compute robot control signals. Interestingly, the action expert uses the same architecture as the π_0 LLM, Gemma. The action expert takes in the robot's current joint positions (state) and the next 50 timesteps of joint positions (actions). The action expert is a diffusion model - it's trained to iteratively map random joint positions to final trajectories. After each pass through the model, the output trajectories are added to the input and passed through again, until final smooth trajectories are produced. At each timestep, the action expert returns positions for each of our robot's 14 joints for the next 50 timesteps. We can visualize these outputs as an image or as time series. The robot then follows these outputs for a number of steps, and then the process is repeated, generating new trajectories.

The Trick. Since our LLM and action expert share the same architecture, we can simply copy over the keys and values from each attention head in our LLM to our action expert's attention heads. This gives the action expert access to incredibly rich information about the robot's current image and text inputs, allowing the models to almost "think as one." Matrix rows are colored according to their corresponding inputs: image inputs are colored using the approximate average patch color (orange patches contain the pen cap), text inputs are colored blue, robot state inputs are red, and actions are pink.

Overview of Pi0 Architecture



Action Expert is randomly initialized and not as wide as Gemma

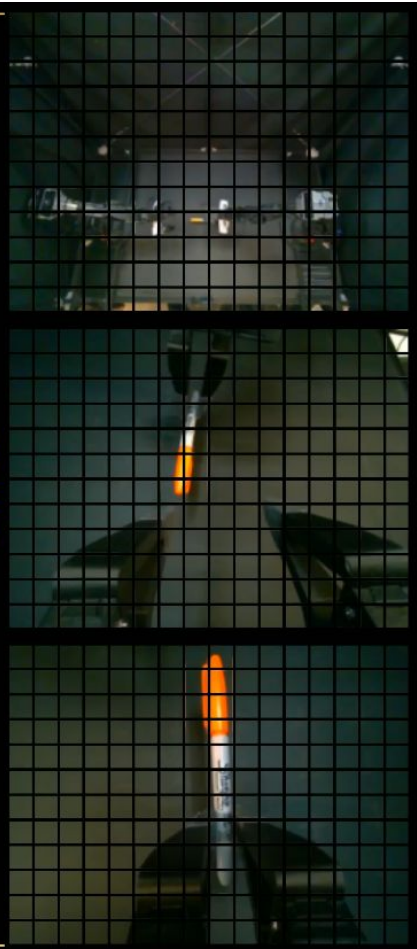


IMAGE ENCODERS

SigLIP So400m

SigLIP So400m

SigLIP So400m

LLM
Gemma 2B

"Uncap the pen"

* The 768 patches include some black "letterboxing" patches above & below, not shown here

π_0

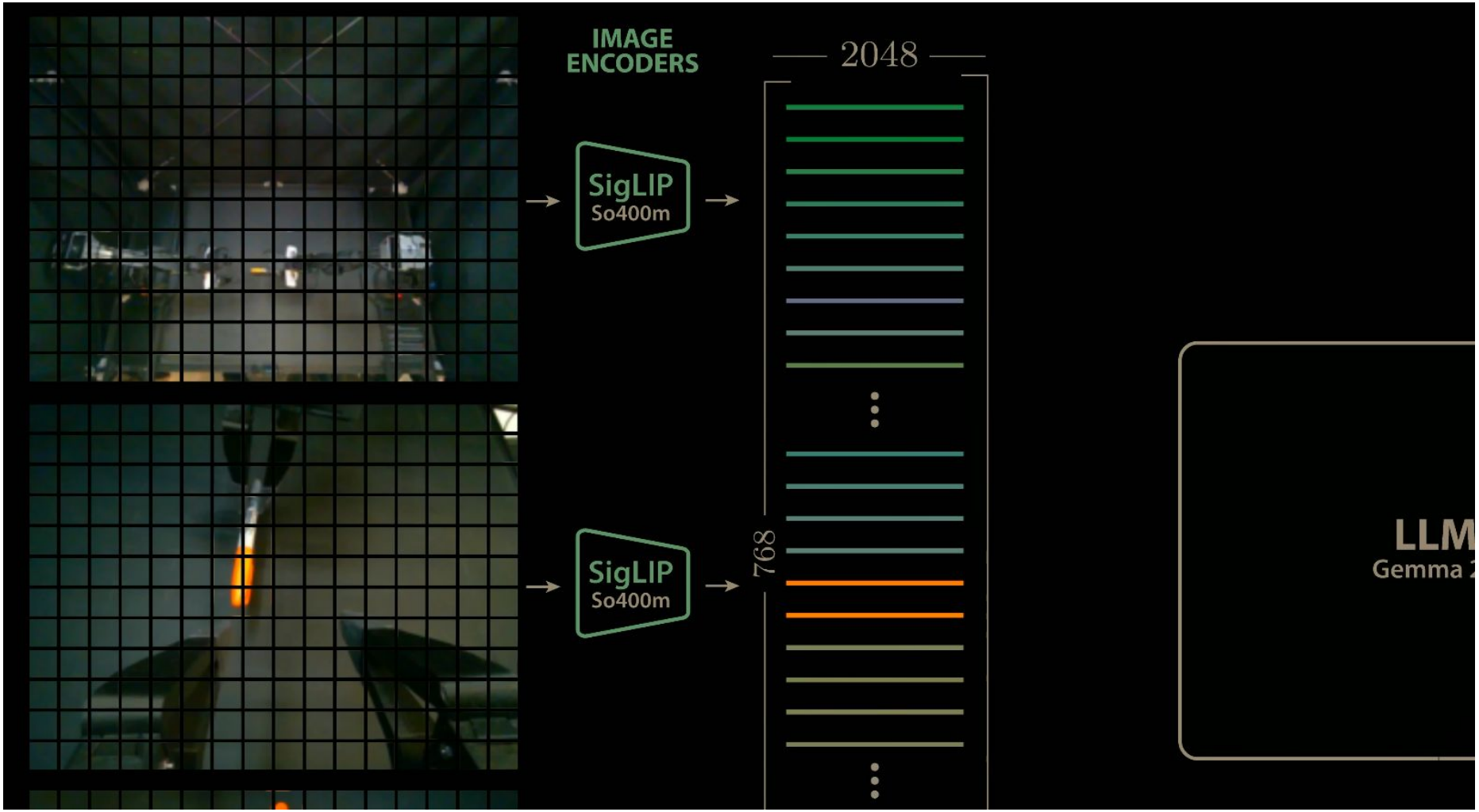


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
768
LLM
Gemma

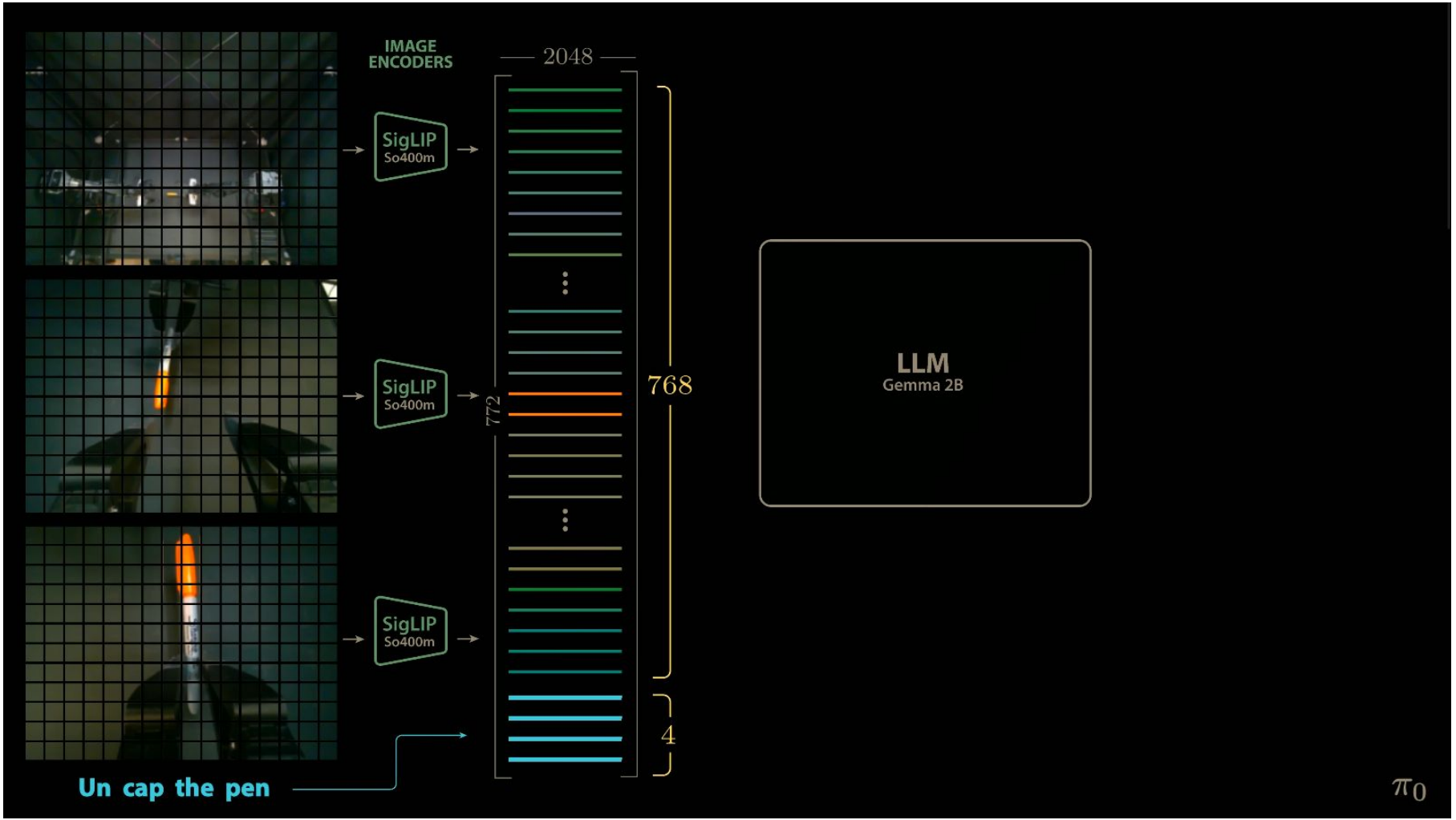
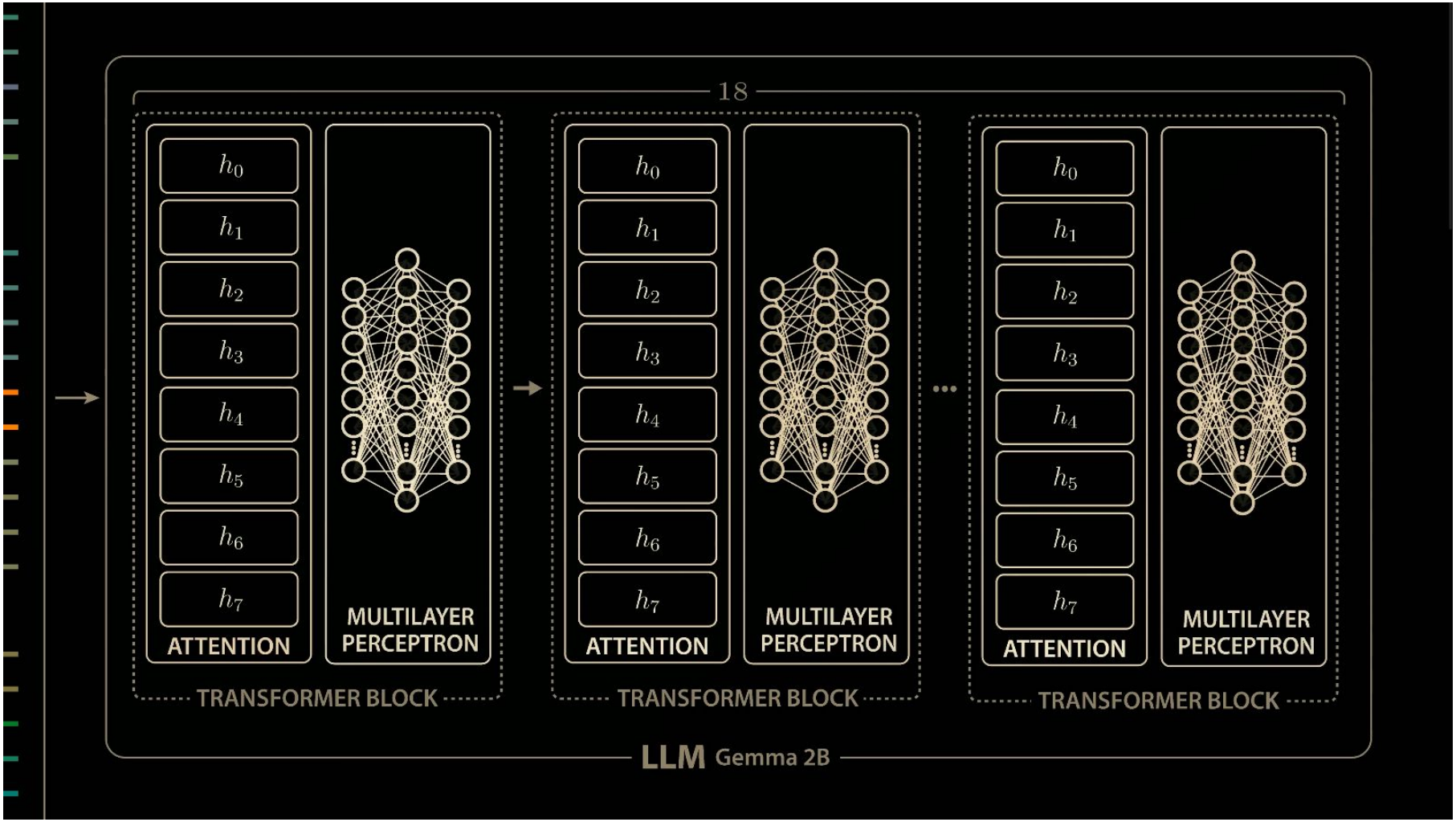


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
SigLIP So400m
772
768
4
LLM
Gemma 2B
Un cap the pen
π_0


18
h_0
h_1
h_2
h_3
h_4
h_5
h_6
h_7
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
LLM Gemma 2B

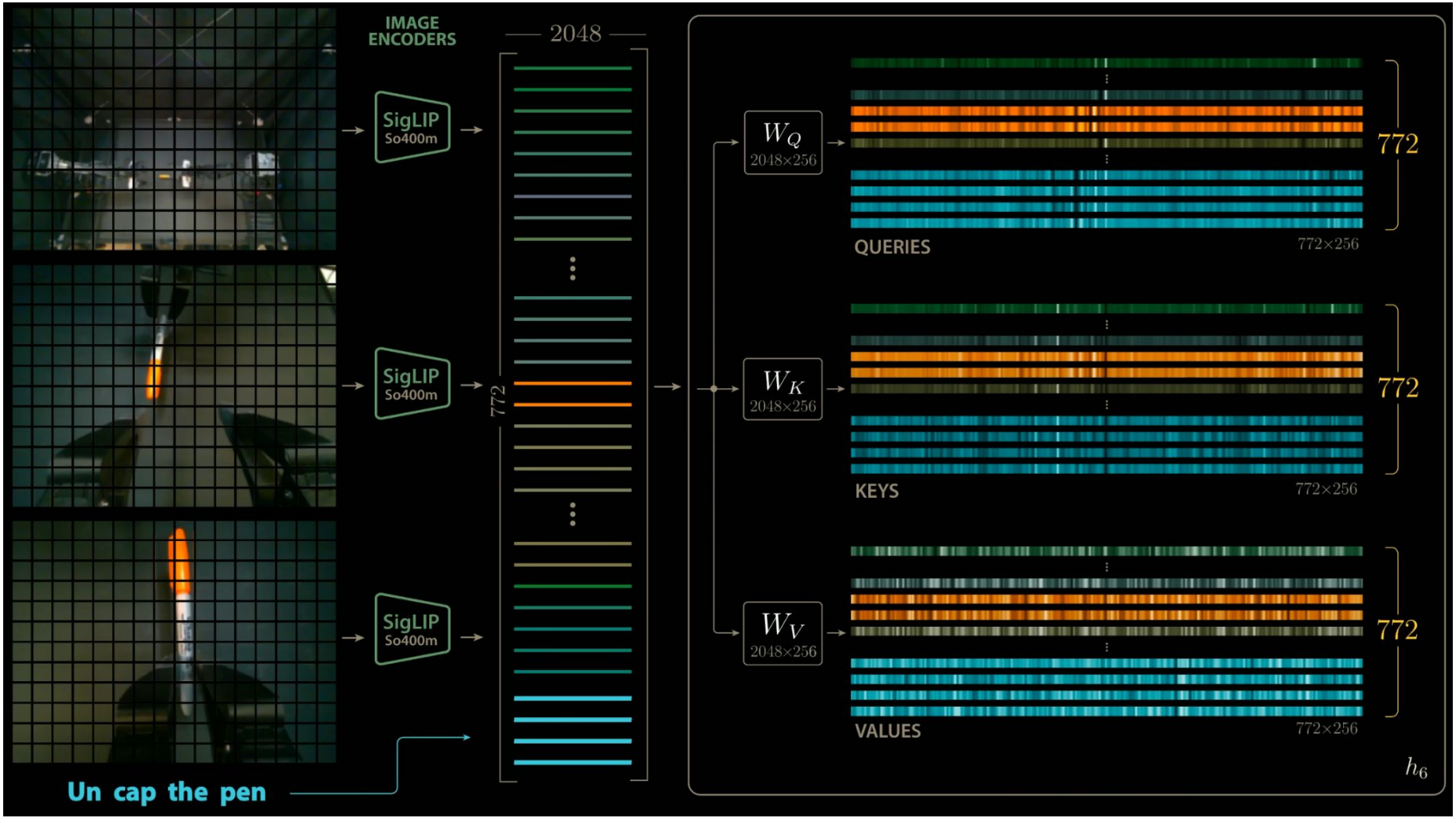


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
SigLIP So400m
772
W_Q 2048×256
W_K 2048×256
W_V 2048×256
QUERIES
772×256
772
KEYS
772×256
772
VALUES
772×256
772
h_6
Un cap the pen

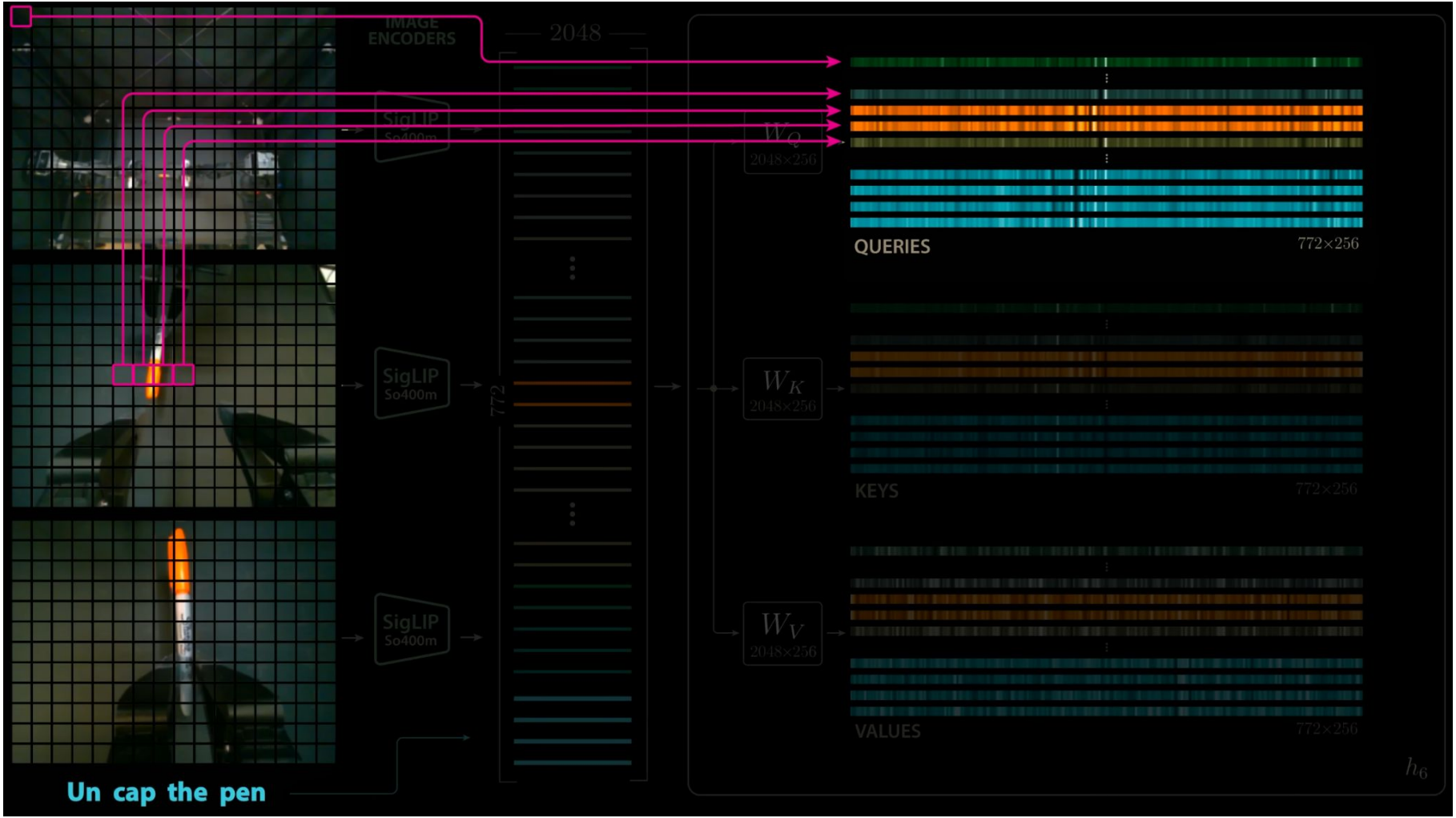
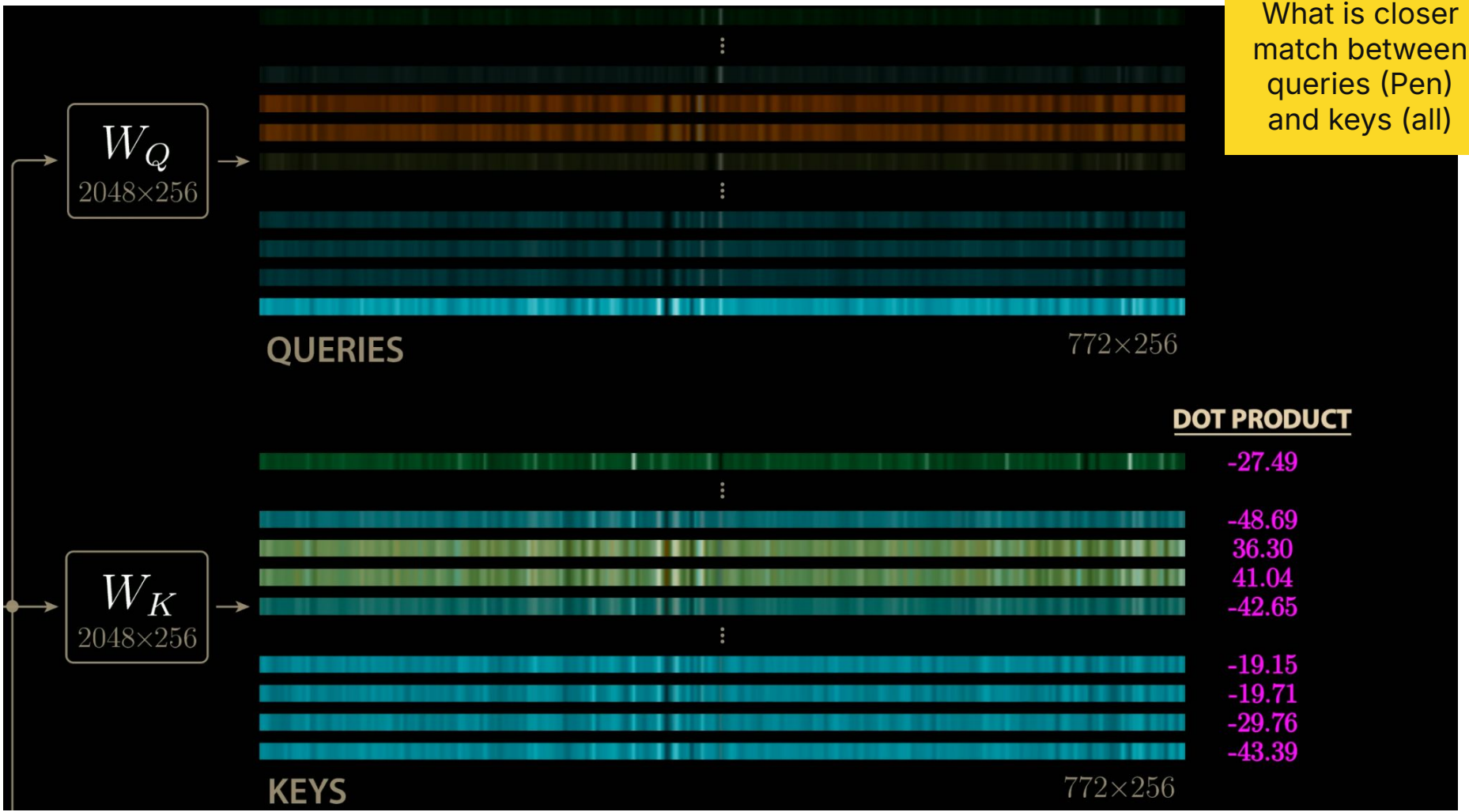
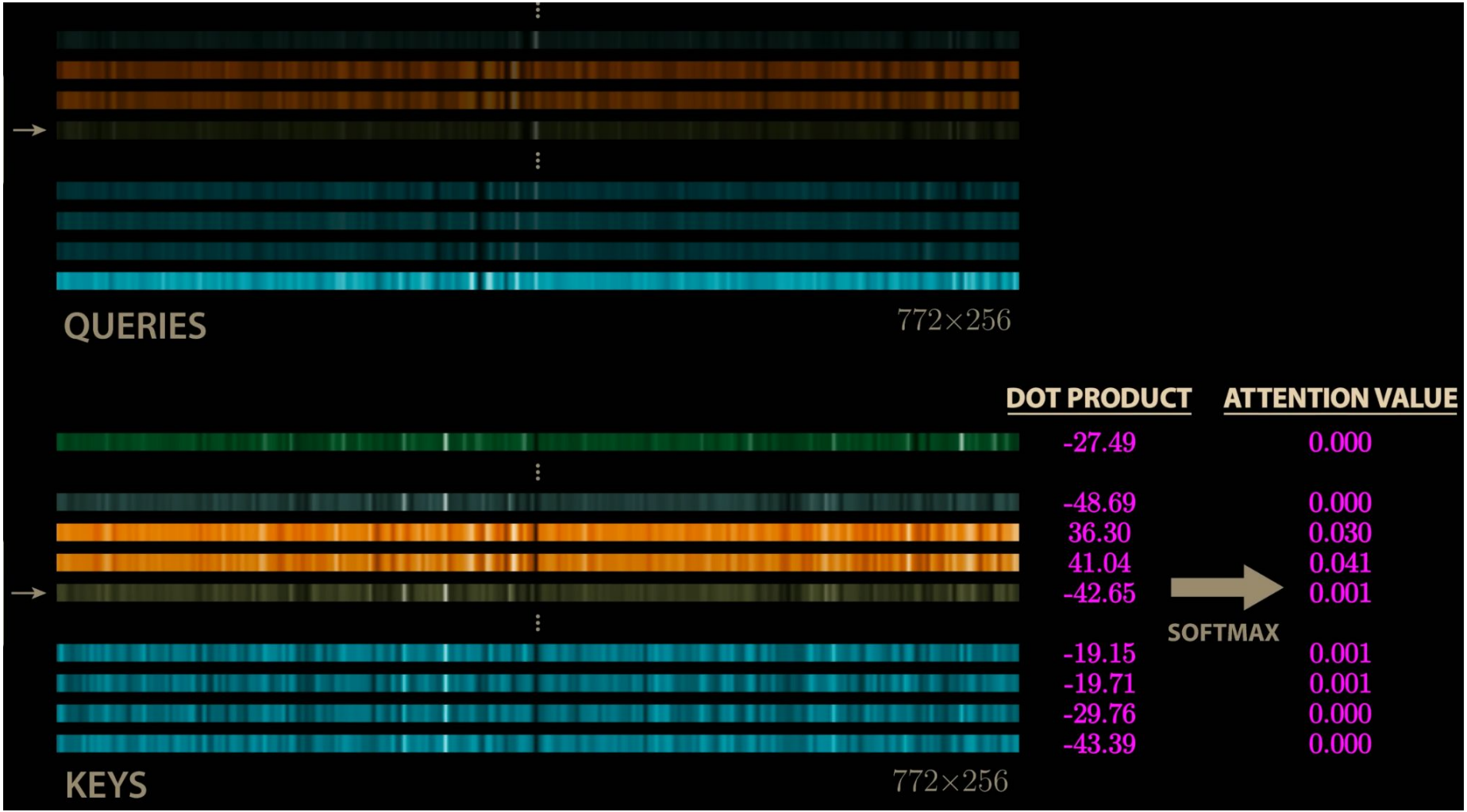


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
SigLIP So400m
772
W_Q 2048×256
W_K 2048×256
W_V 2048×256
QUERIES
772×256
KEYS
772×256
VALUES
772×256
h_6
Un cap the pen


What is closer match between queries (Pen) and keys (all)
W_Q
2048×256
QUERIES
772×256
DOT PRODUCT
-27.49
-48.69
36.30
41.04
-42.65
-19.15
-19.71
-29.76
-43.39
W_K
2048×256
KEYS
772×256


QUERIES
772×256
DOT PRODUCT
ATTENTION VALUE
-27.49
0.000
-48.69
0.000
36.30
0.030
41.04
0.041
-42.65
0.001
SOFTMAX
-19.15
0.001
-19.71
0.001
-29.76
0.000
-43.39
0.000
KEYS
772×256

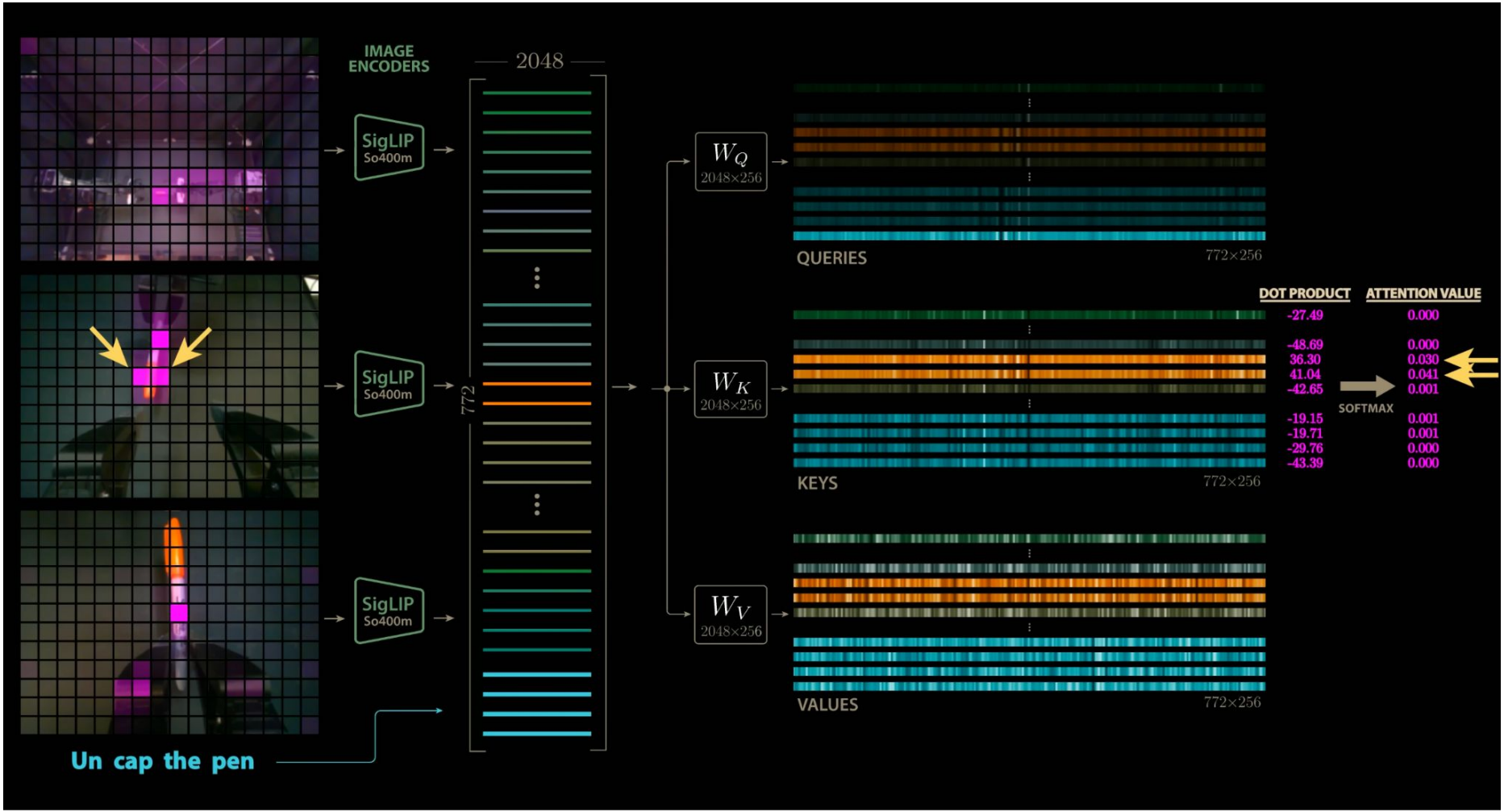
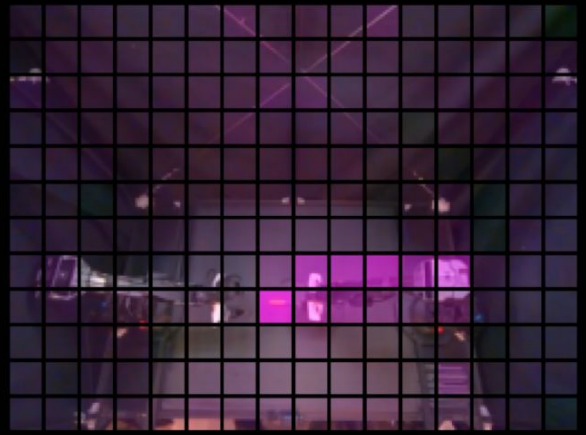
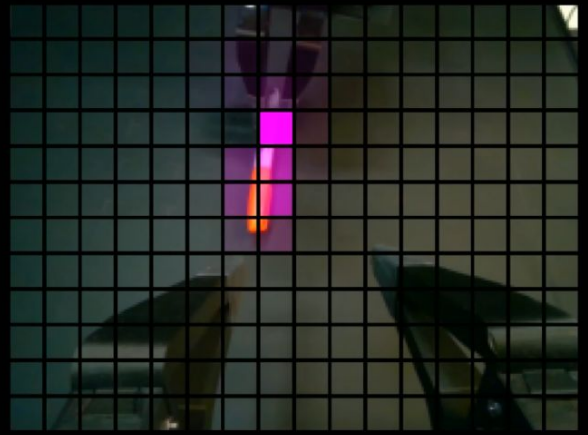


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
SigLIP So400m
772
W_Q 2048×256
W_K 2048×256
W_V 2048×256
QUERIES
772×256
KEYS
772×256
VALUES
772×256
DOT PRODUCT
ATTENTION VALUE
-27.49
0.000
-48.69
0.000
36.30
0.030
41.04
0.041
-42.65
0.001
SOFTMAX
-19.15
0.001
-19.71
0.001
-29.76
0.000
-43.39
0.000
Un cap the pen

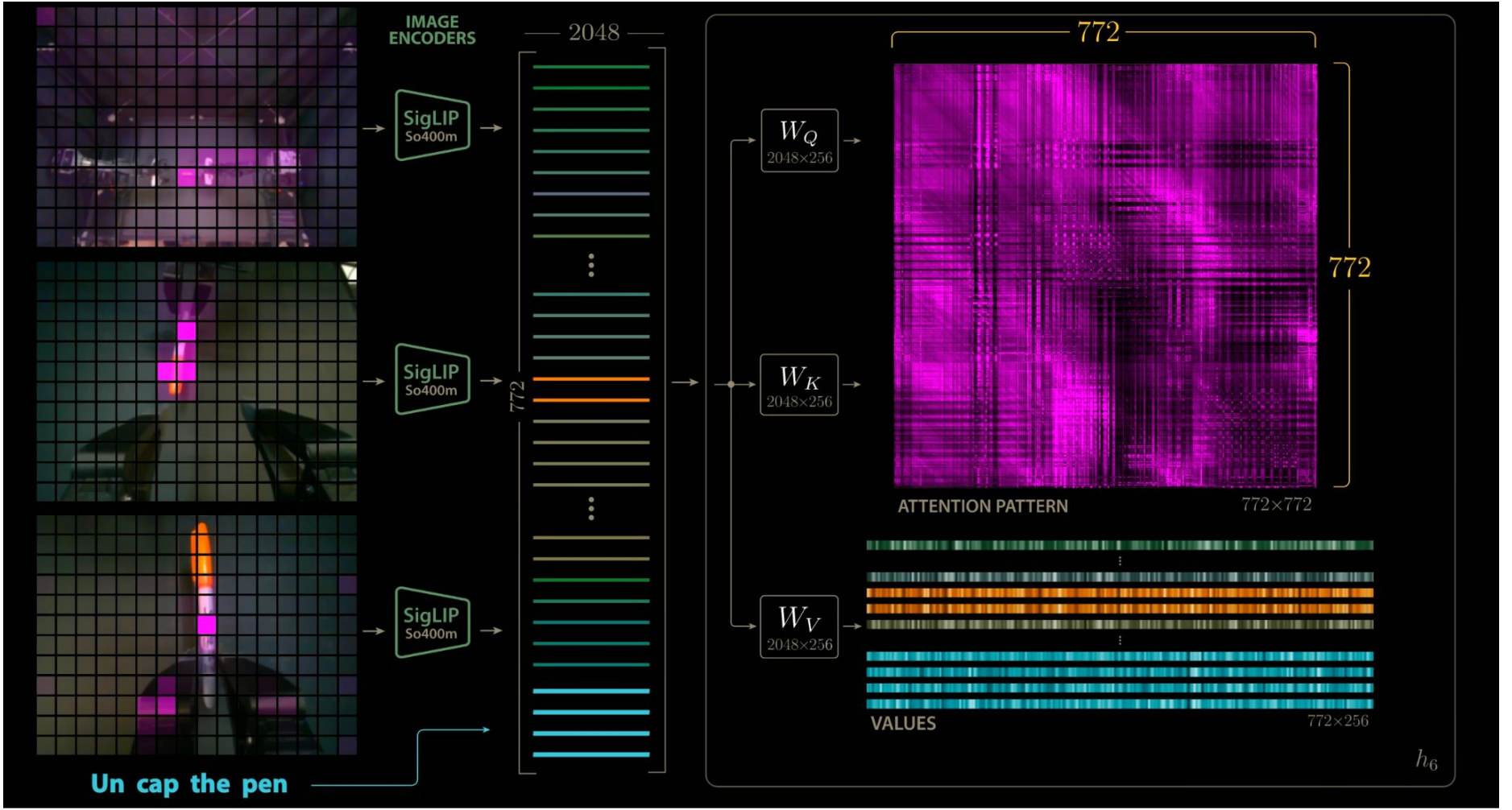



IMAGE ENCODERS
SigLIP So400m
SigLIP So400m
SigLIP So400m
2048
772
W_Q 2048×256
W_K 2048×256
W_V 2048×256
772
772
ATTENTION PATTERN
772×772
VALUES
772×256
h_6
Un cap the pen

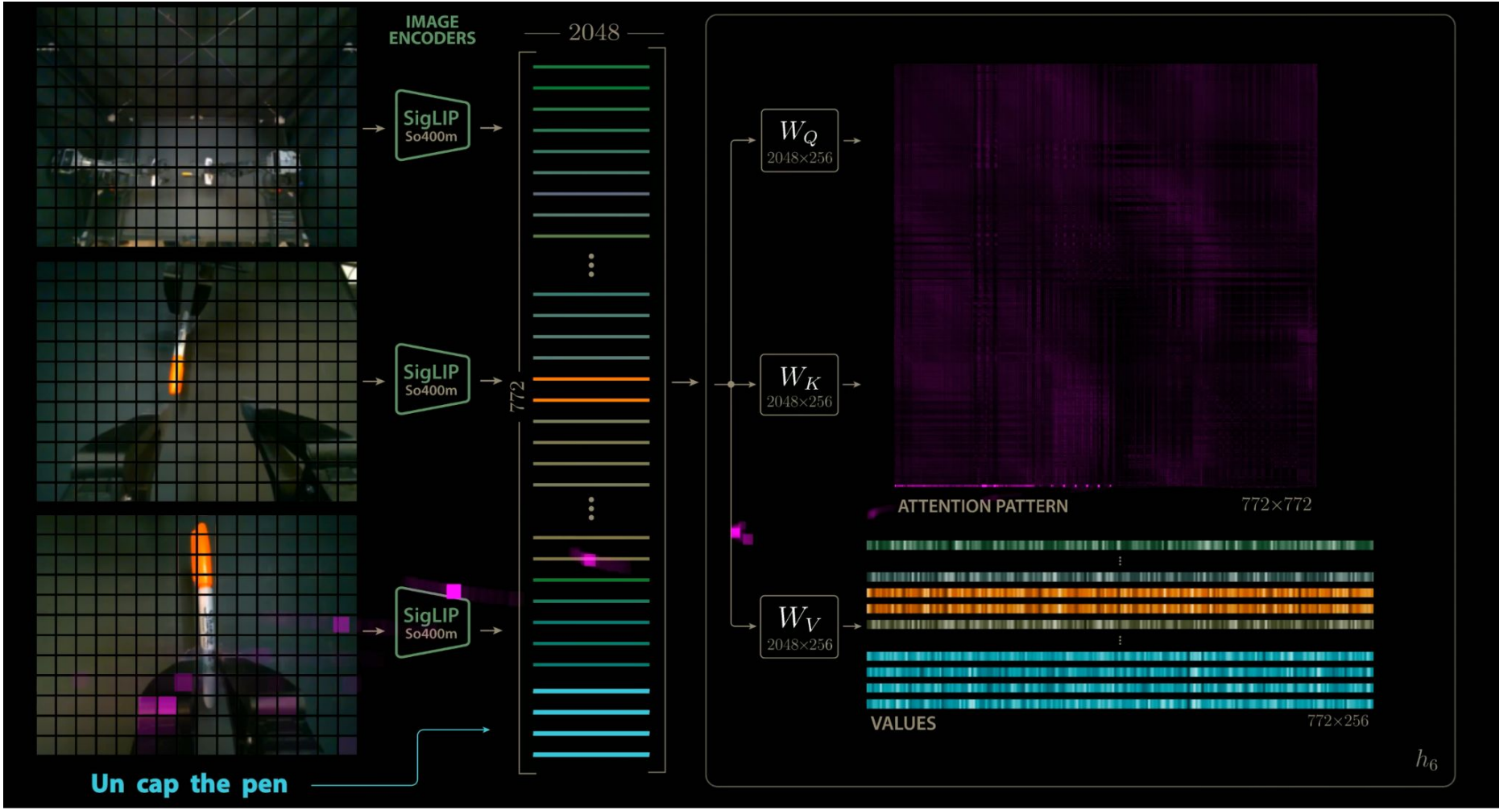
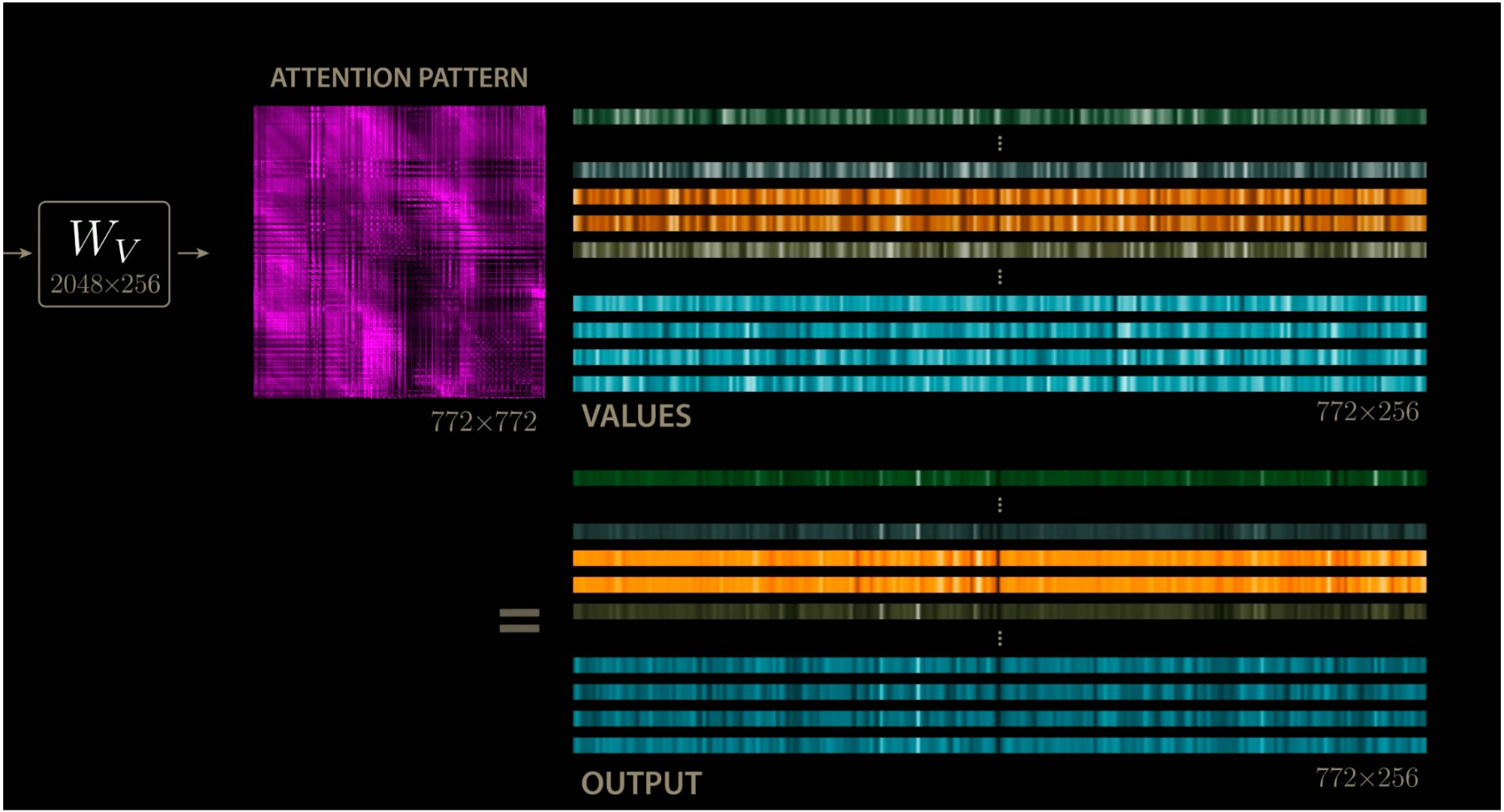
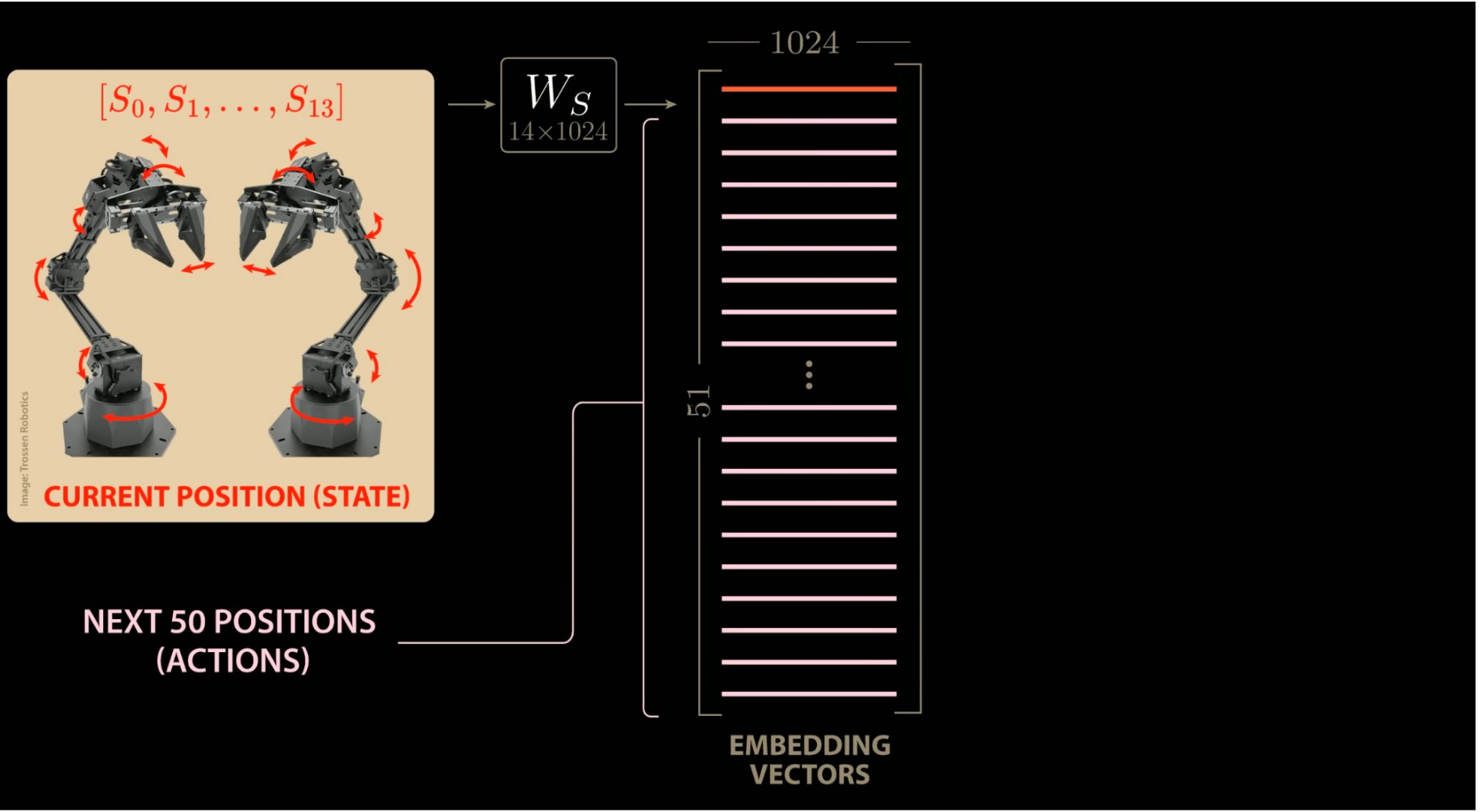


IMAGE ENCODERS
2048
SigLIP So400m
SigLIP So400m
SigLIP So400m
772
W_Q 2048×256
W_K 2048×256
W_V 2048×256
ATTENTION PATTERN
772×772
VALUES
772×256
h_6
Un cap the pen


ATTENTION PATTERN
W_V
2048×256
772×772
VALUES
772×256
=
OUTPUT
772×256


$[S_0, S_1, \ldots, S_{13}]$
Image: Trossen Robotics
CURRENT POSITION (STATE)
W_S
14×1024
1024
51
NEXT 50 POSITIONS
(ACTIONS)
EMBEDDING
VECTORS

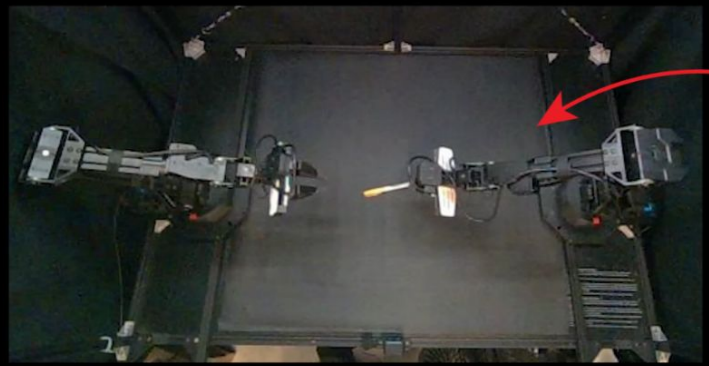

1
10
20
30
40
50
60
70
80
90
100

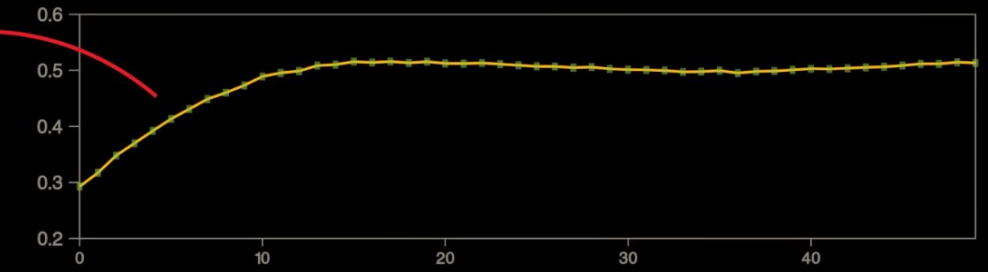
ACTION EXPERT

JOINT

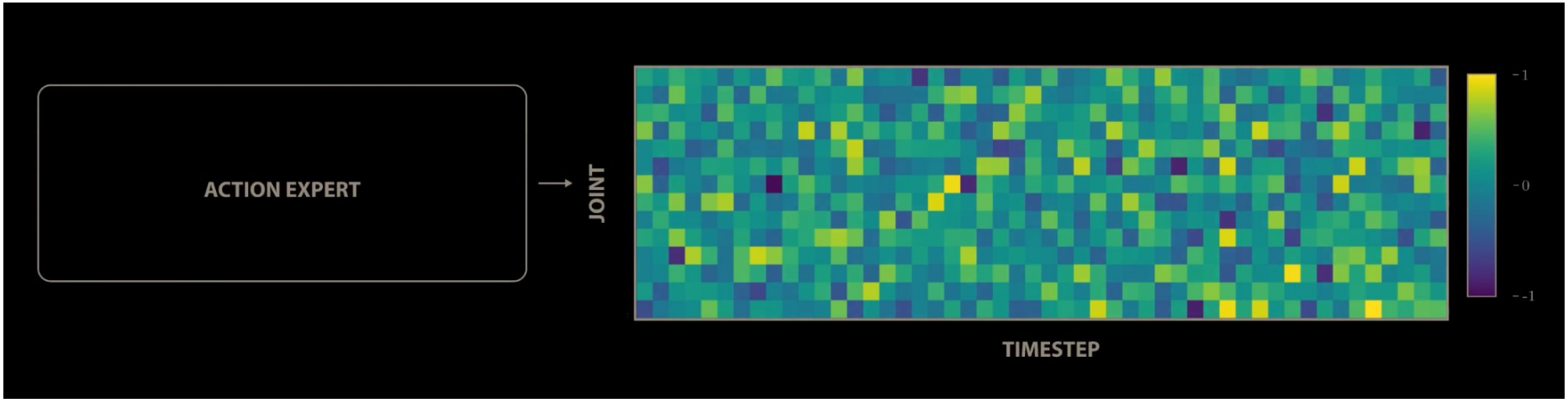


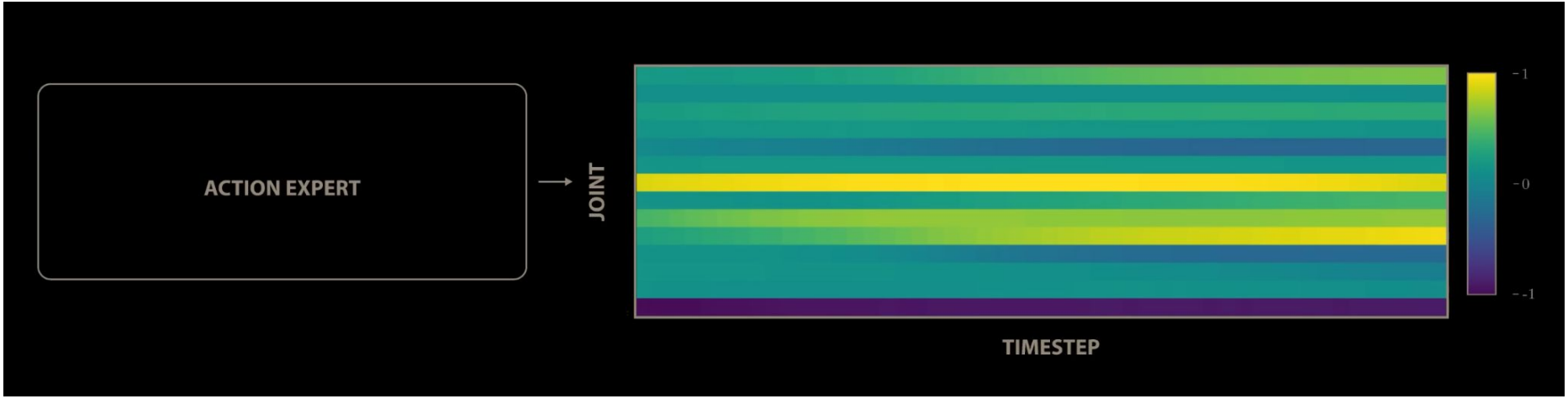
TIMESTEP

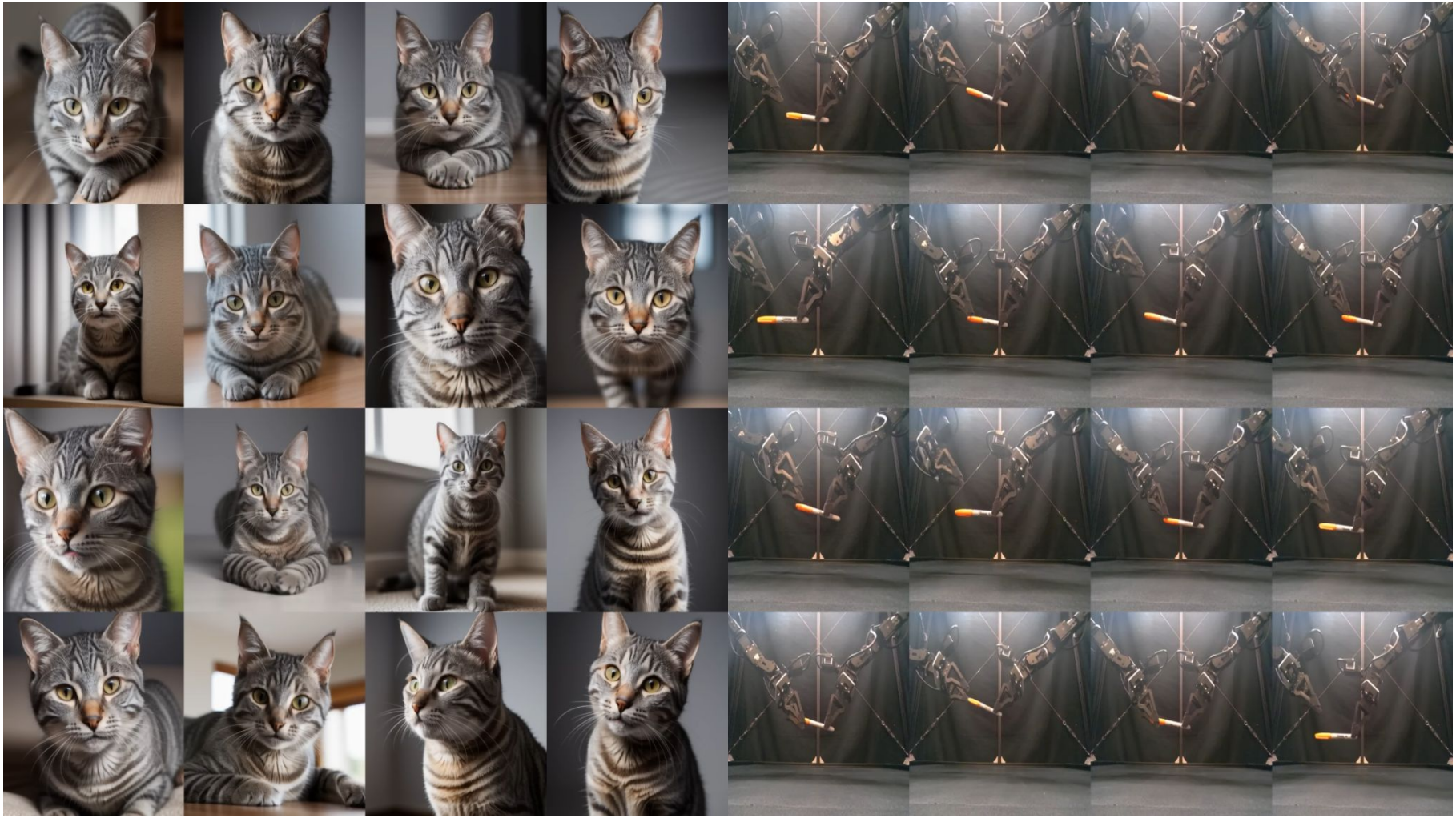


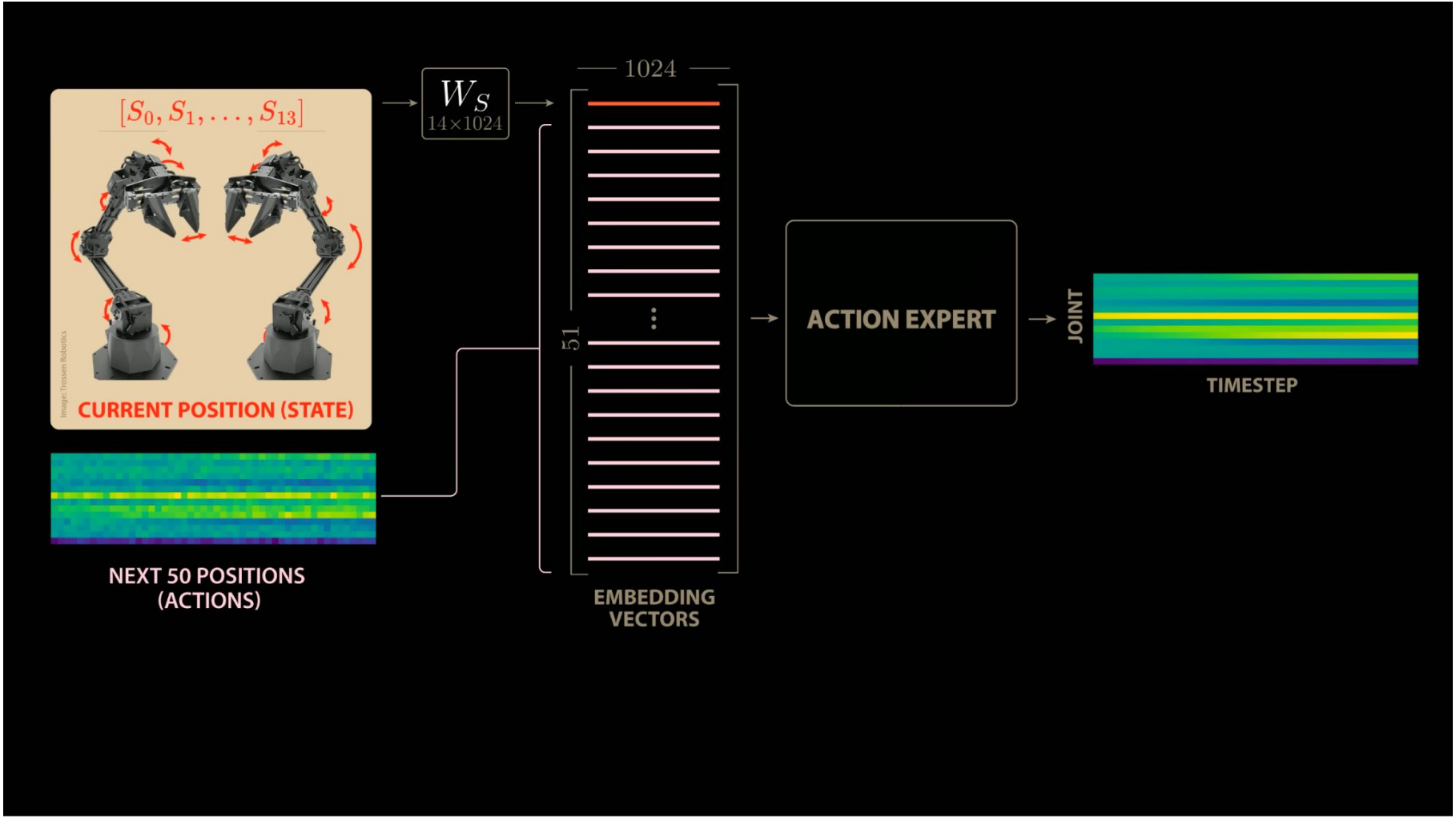


TIMESTEP


ACTION EXPERT
JOINT
TIMESTEP
1
0
-1


ACTION EXPERT
JOINT
TIMESTEP
1
0
-1




$[S_0, S_1, \ldots, S_{13}]$
Image: Trossen Robotics
CURRENT POSITION (STATE)
W_S
14×1024
1024
51
NEXT 50 POSITIONS
(ACTIONS)
EMBEDDING
VECTORS
ACTION EXPERT
JOINT
TIMESTEP

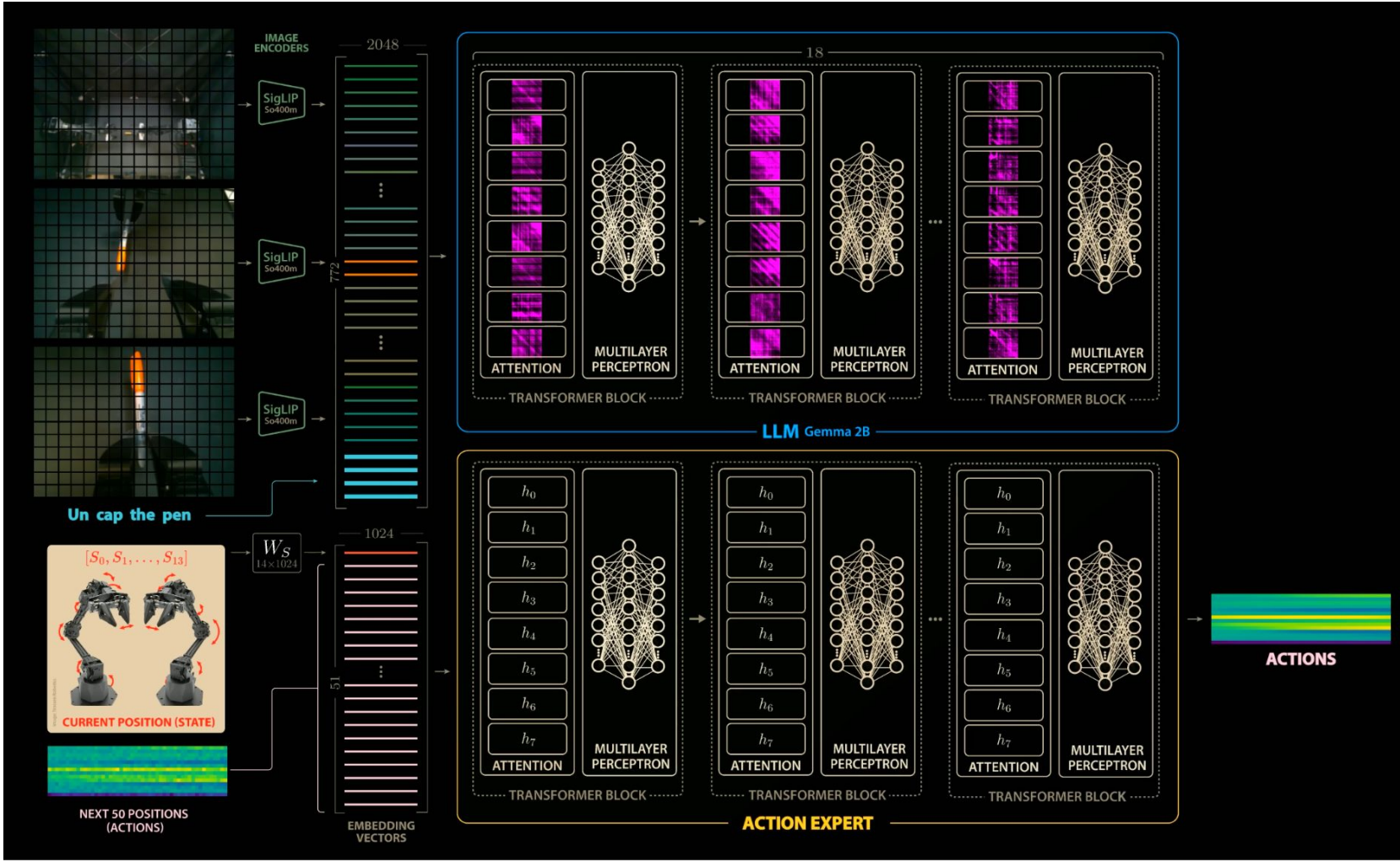
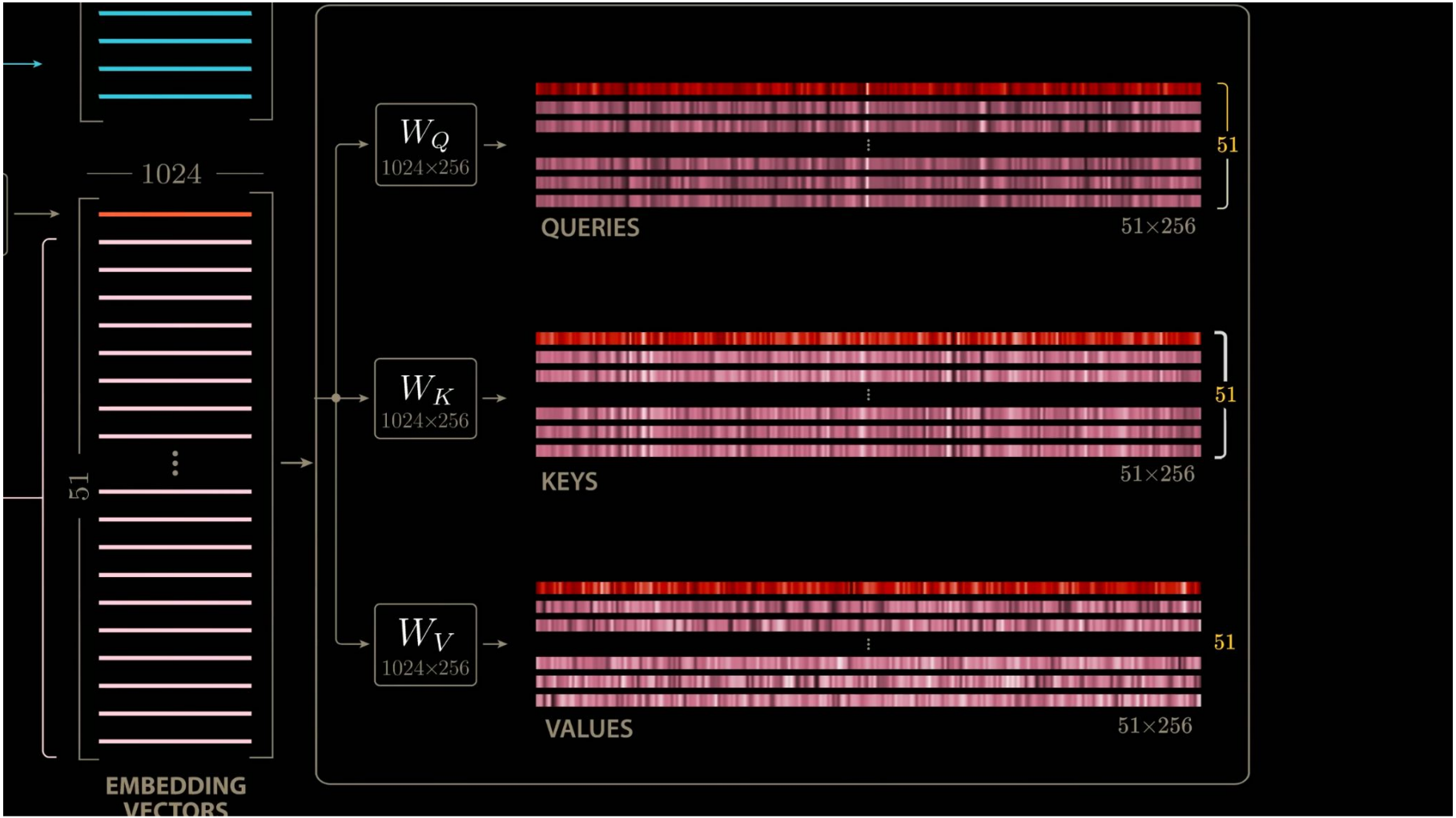
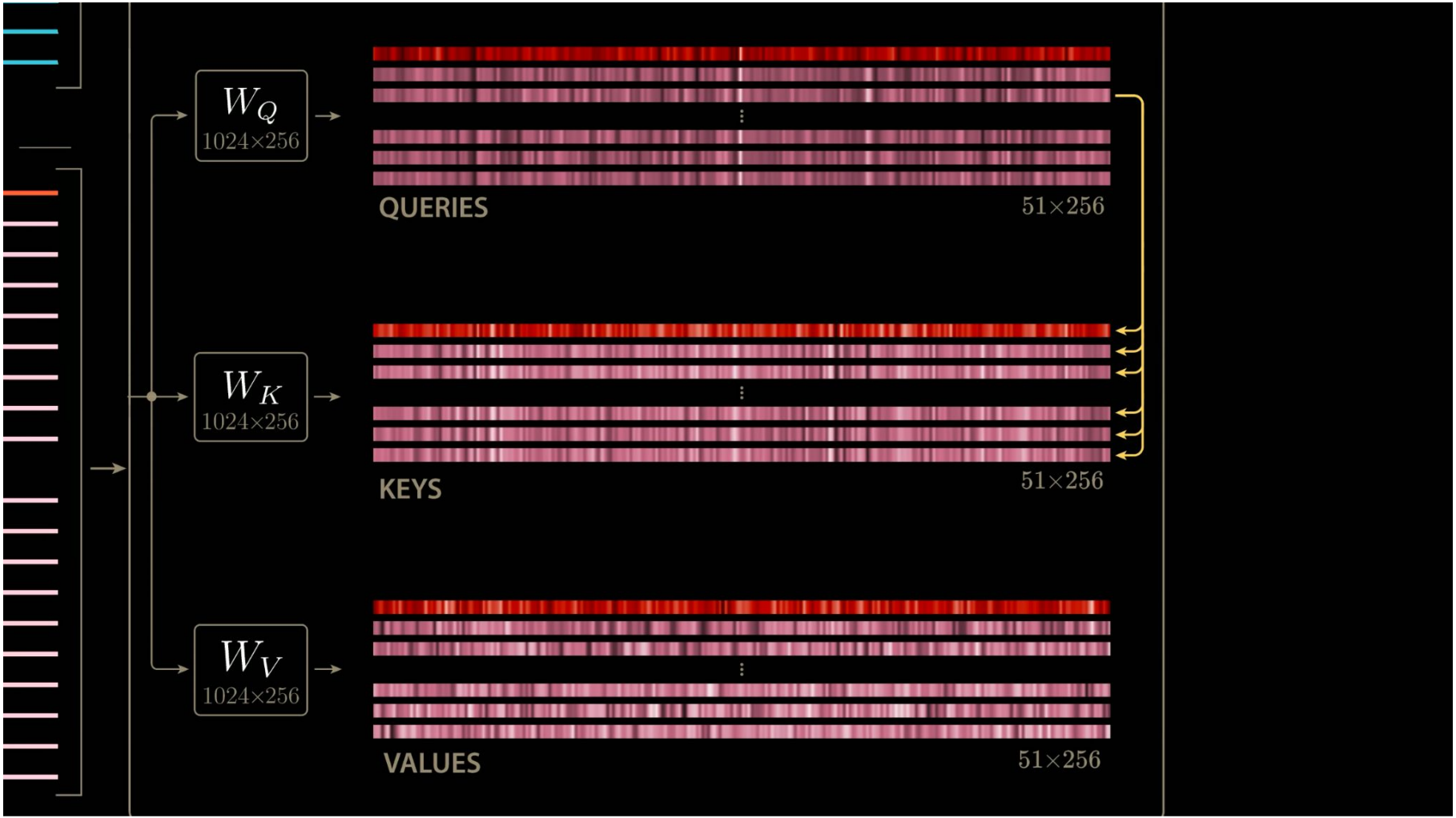


IMAGE ENCODERS
SigLIP So400m
SigLIP So400m
SigLIP So400m
2048
772
18
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
LLM Gemma 2B
Un cap the pen
$[S_0, S_1, \ldots, S_{13}]$
CURRENT POSITION (STATE)
W_S 14×1024
1024
51
NEXT 50 POSITIONS (ACTIONS)
EMBEDDING VECTORS
h_0 h_1 h_2 h_3 h_4 h_5 h_6 h_7
ACTION EXPERT
ACTIONS


1024
51
EMBEDDING VECTORS
W_Q
1024×256
W_K
1024×256
W_V
1024×256
QUERIES
51×256
51
KEYS
51×256
51
VALUES
51×256
51


W_Q
1024×256
W_K
1024×256
W_V
1024×256
QUERIES
51×256
KEYS
51×256
VALUES
51×256

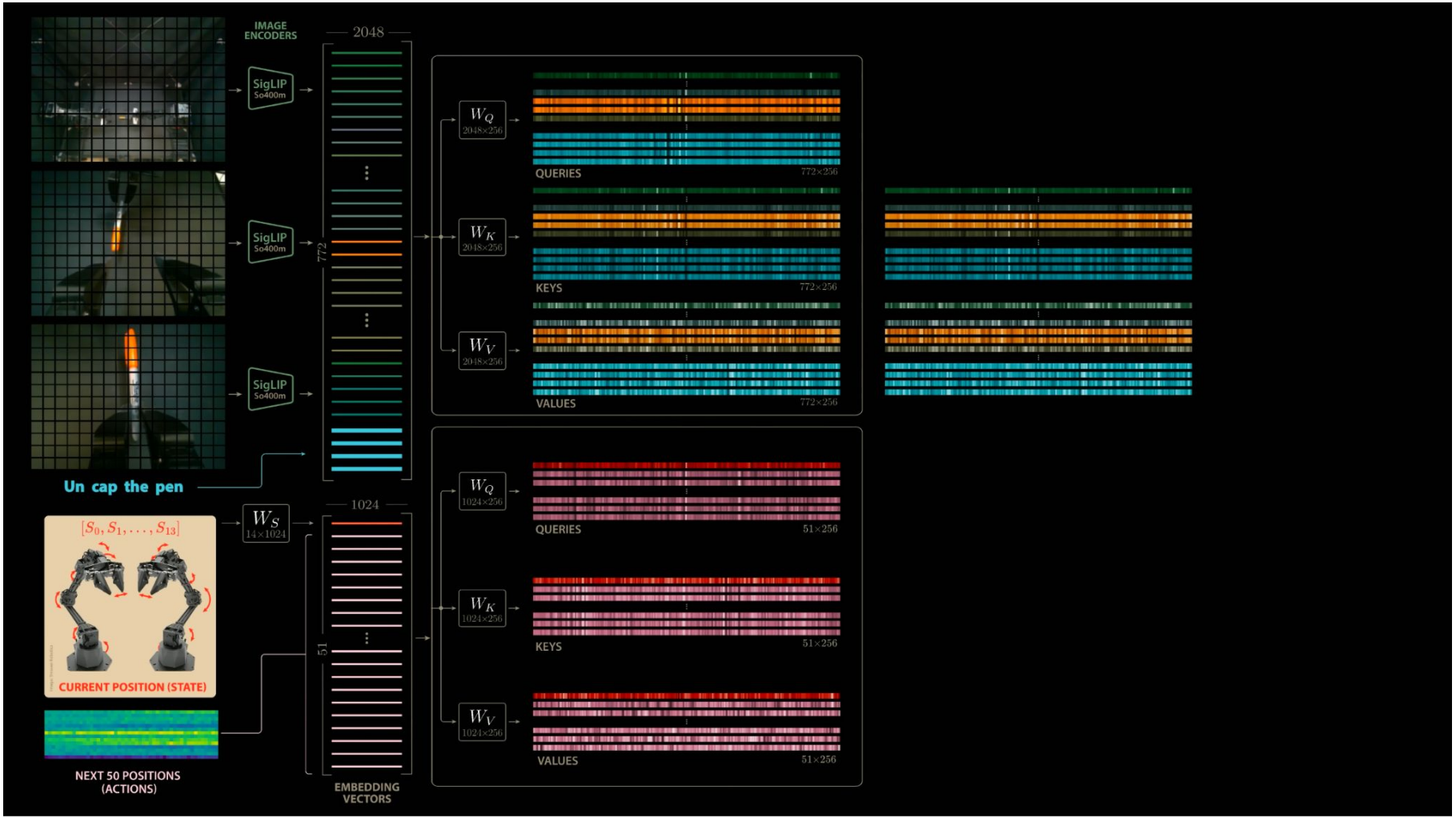
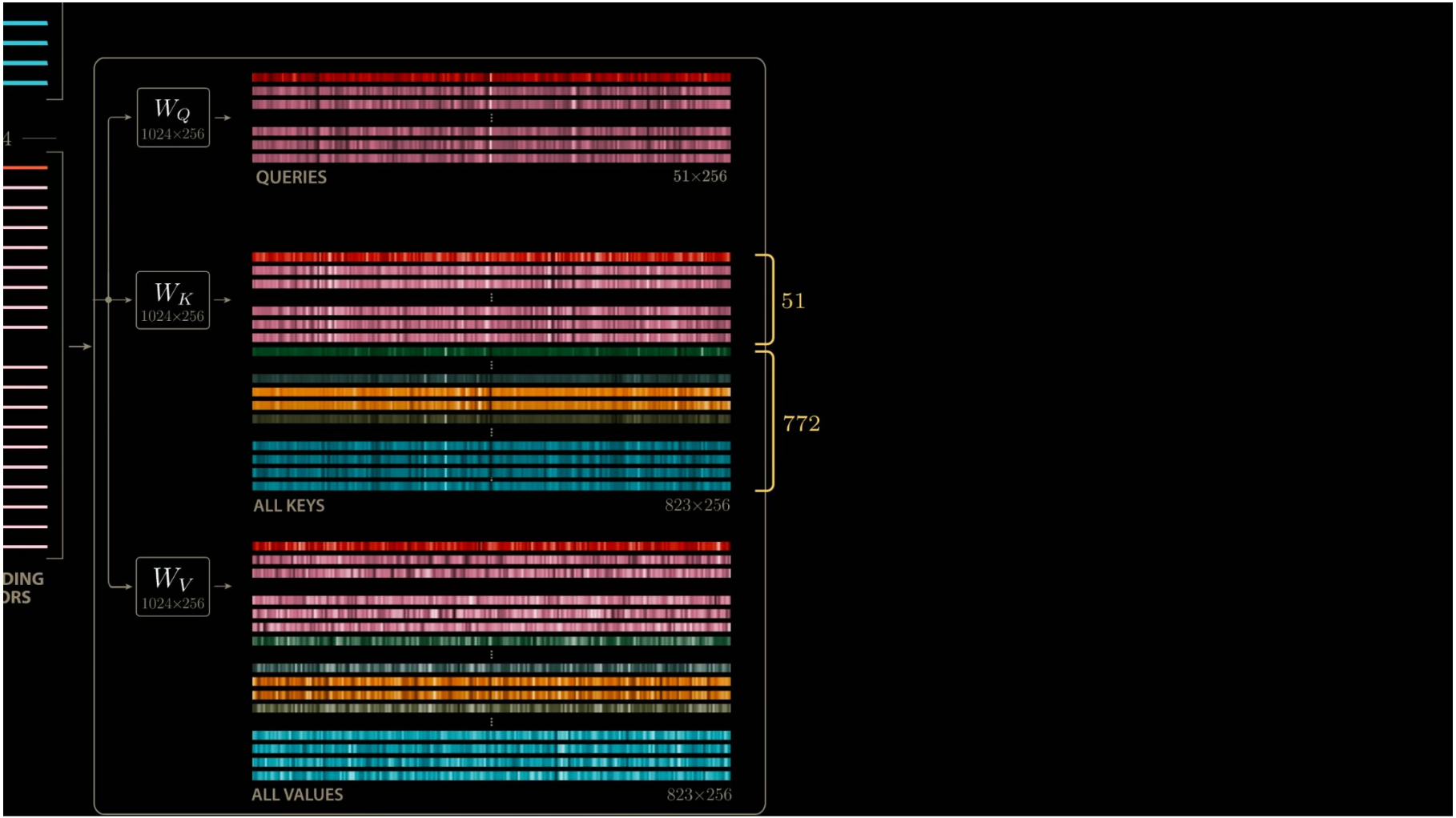
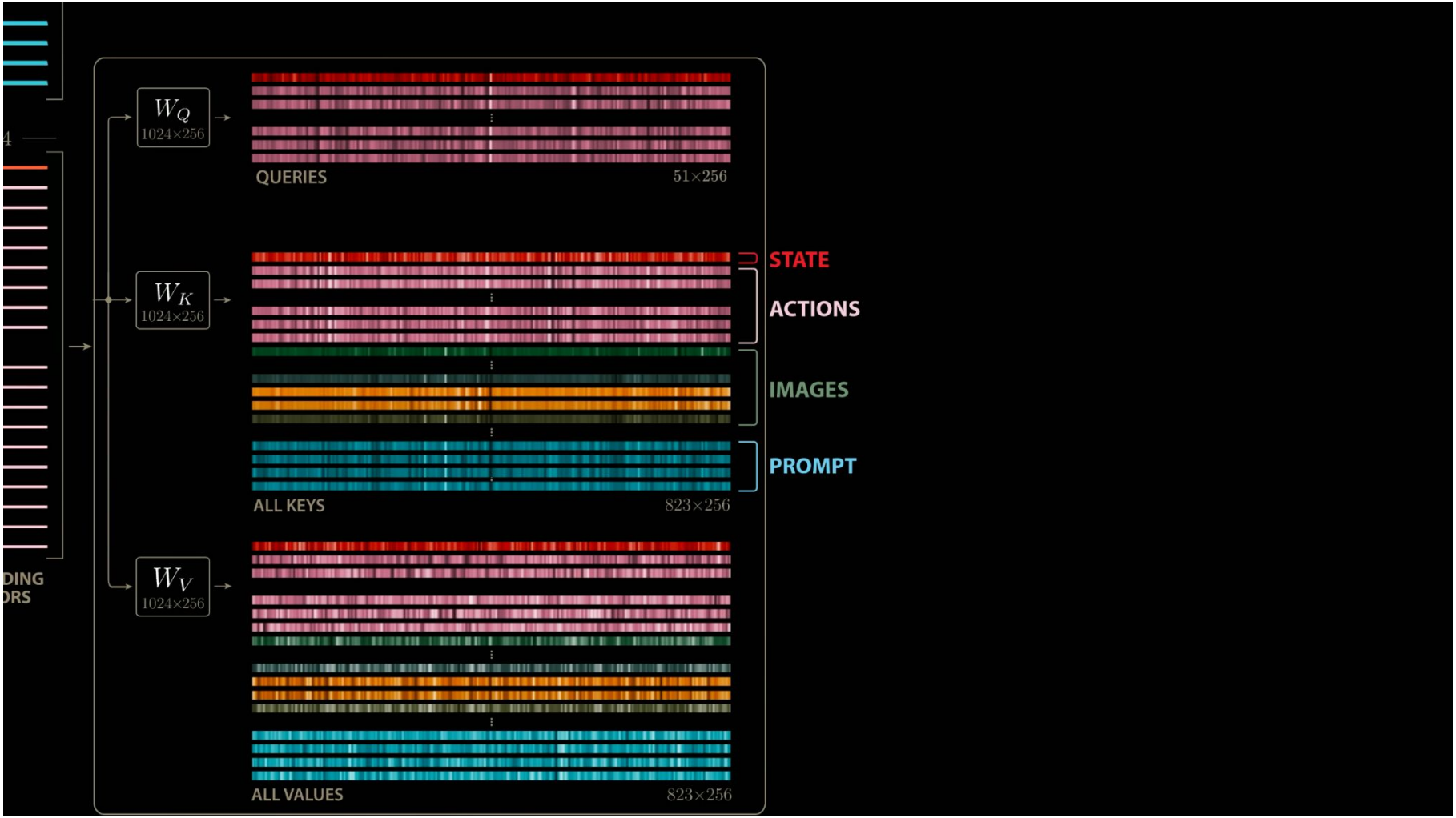
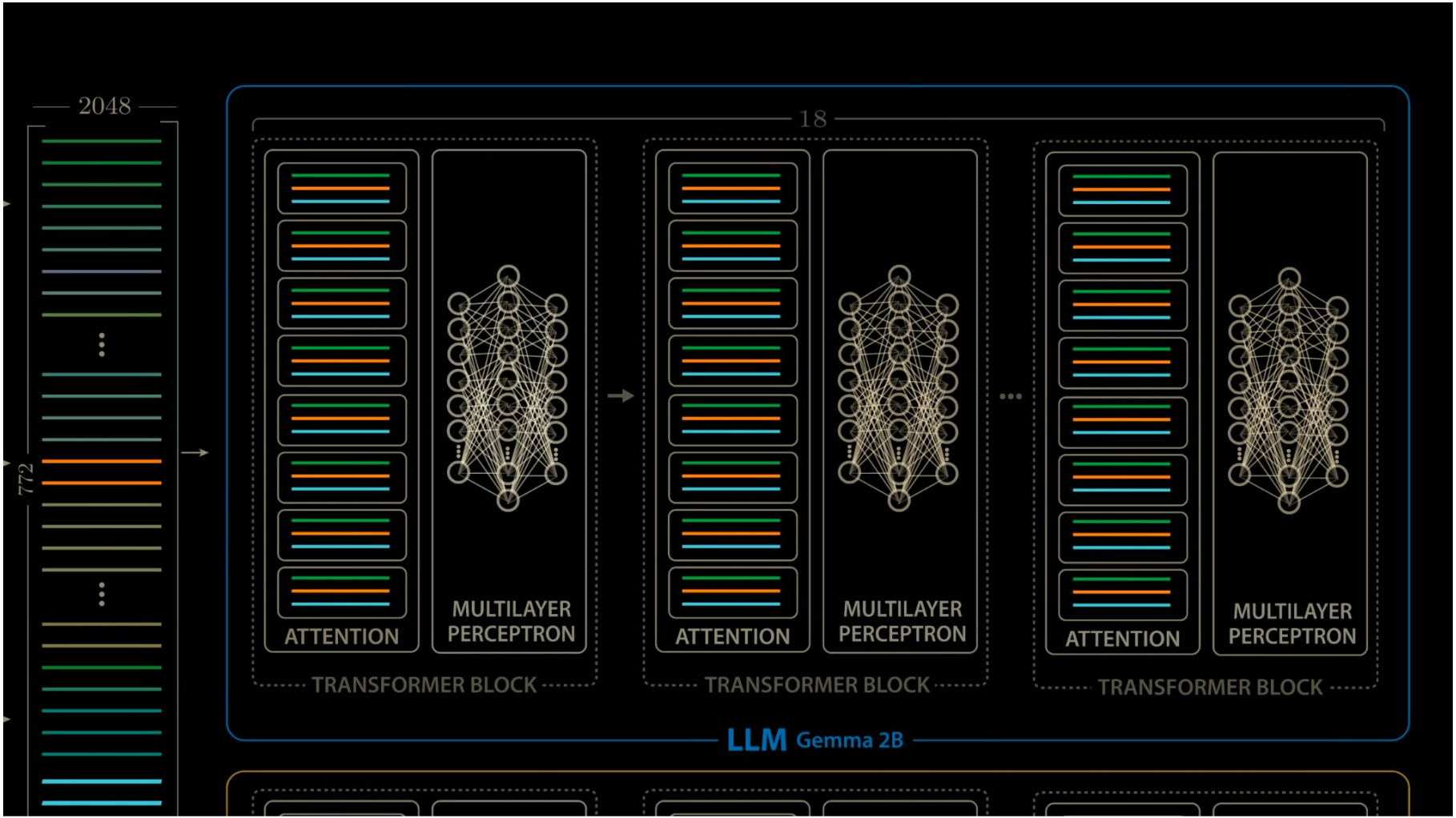
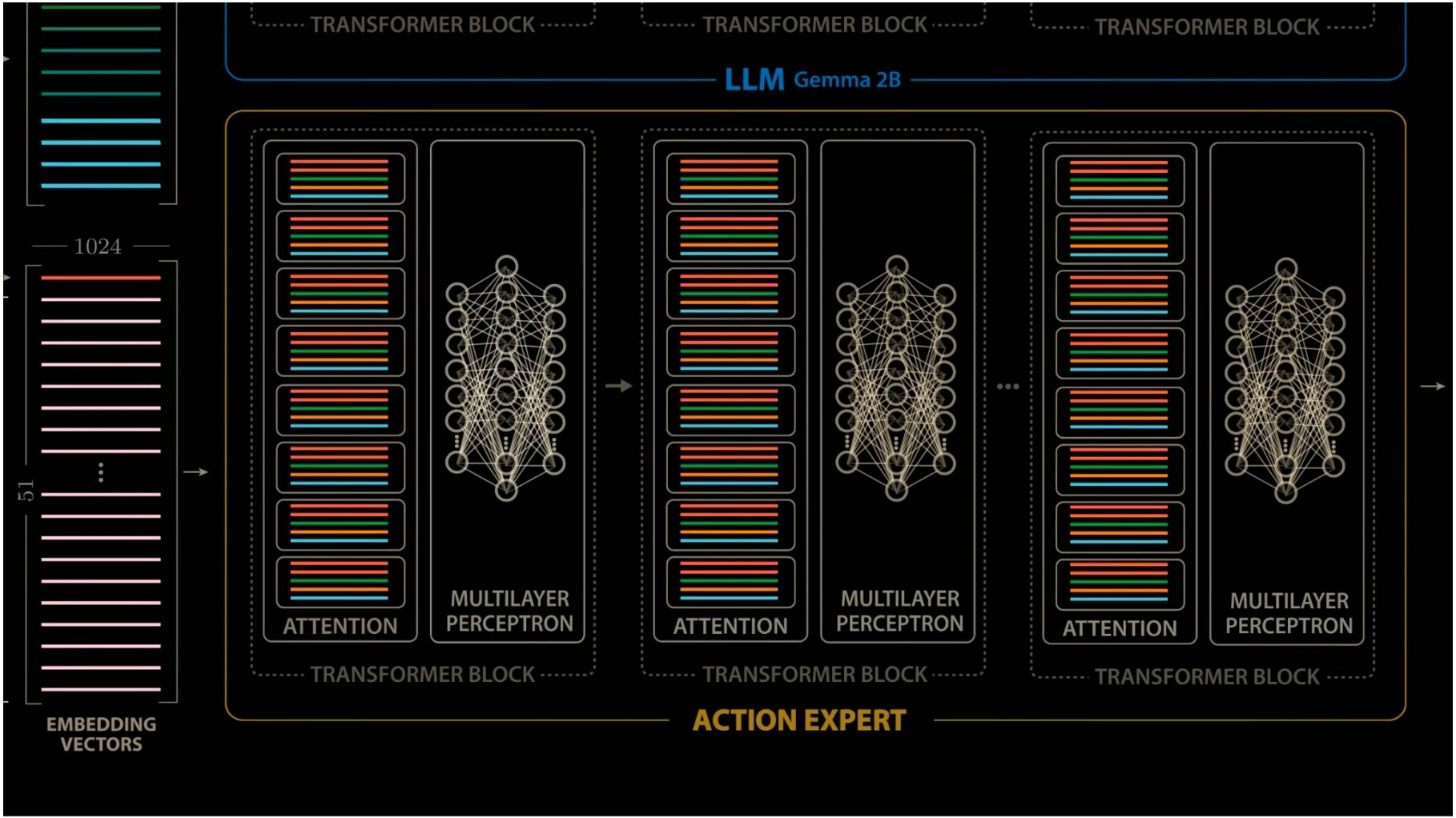


IMAGE ENCODERS
SigLIP So400m
SigLIP So400m
SigLIP So400m
2048
772
Un cap the pen
W_S 14×1024
$[S_0, S_1, \ldots, S_{13}]$
CURRENT POSITION (STATE)
NEXT 50 POSITIONS (ACTIONS)
1024
51
EMBEDDING VECTORS
W_Q 2048×256
W_K 2048×256
W_V 2048×256
QUERIES 772×256
KEYS 772×256
VALUES 772×256
W_Q 1024×256
W_K 1024×256
W_V 1024×256
QUERIES 51×256
KEYS 51×256
VALUES 51×256


W_Q
1024×256
QUERIES
51×256
W_K
1024×256
51
772
ALL KEYS
823×256
W_V
1024×256
ALL VALUES
823×256


W_Q
1024×256
QUERIES
51×256
W_K
1024×256
STATE
ACTIONS
IMAGES
PROMPT
ALL KEYS
823×256
W_V
1024×256
ALL VALUES
823×256


2048
772
18
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
LLM Gemma 2B


TRANSFORMER BLOCK
TRANSFORMER BLOCK
TRANSFORMER BLOCK
LLM Gemma 2B
1024
51
EMBEDDING VECTORS
ATTENTION
MULTILAYER PERCEPTRON
ATTENTION
MULTILAYER PERCEPTRON
ATTENTION
MULTILAYER PERCEPTRON
TRANSFORMER BLOCK
TRANSFORMER BLOCK
TRANSFORMER BLOCK
ACTION EXPERT