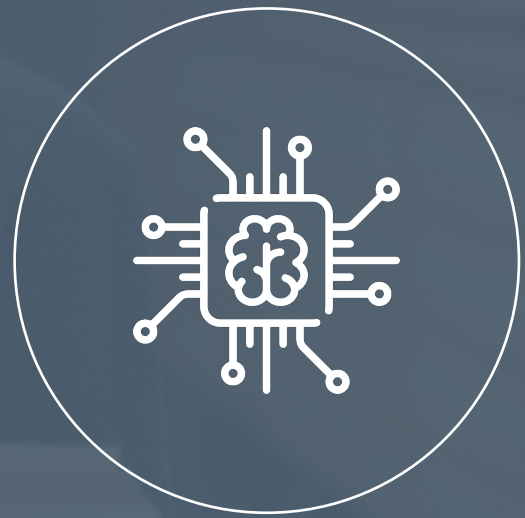
GATO

A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction



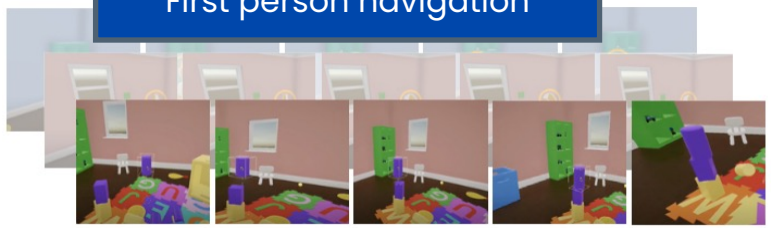
Token embedding and sequencing

Niklas Forsstroem
forsstroemniklas@gmail.com

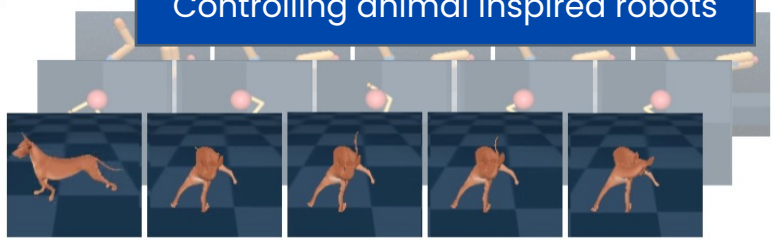
Outperform humans in Atari games

sample efficiency & generalization

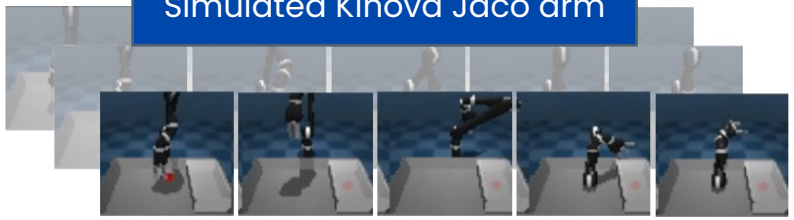
First person navigation



Controlling animal inspired robots



Simulated Kinova Jaco arm



Gato

Transformer sequence model with single set of weights

Maintaining dialogues

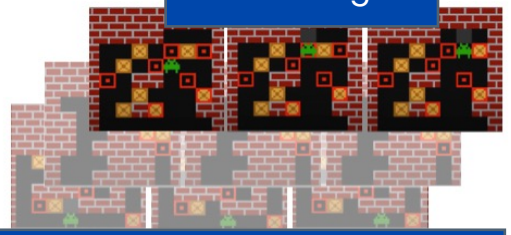
What is the capital of France?

Paris.

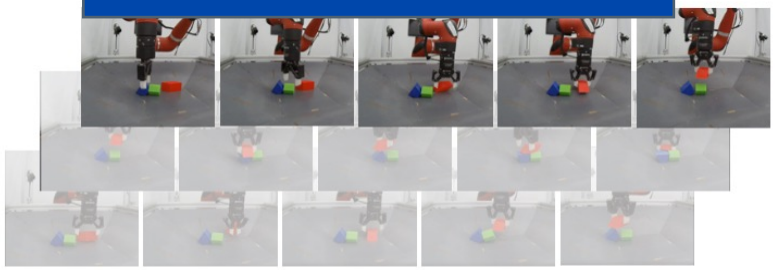
Can you write me a poem?

I don't know exactly what to write. There's just so much to answer.

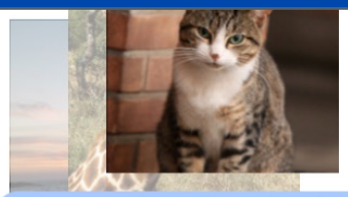
Planning



Stack blocks with real robot arm

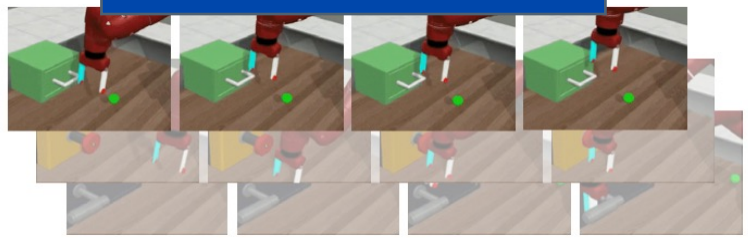


Generate image captions



A cat that is sitting next to a brick wall and green grass some bushes as the sun sets

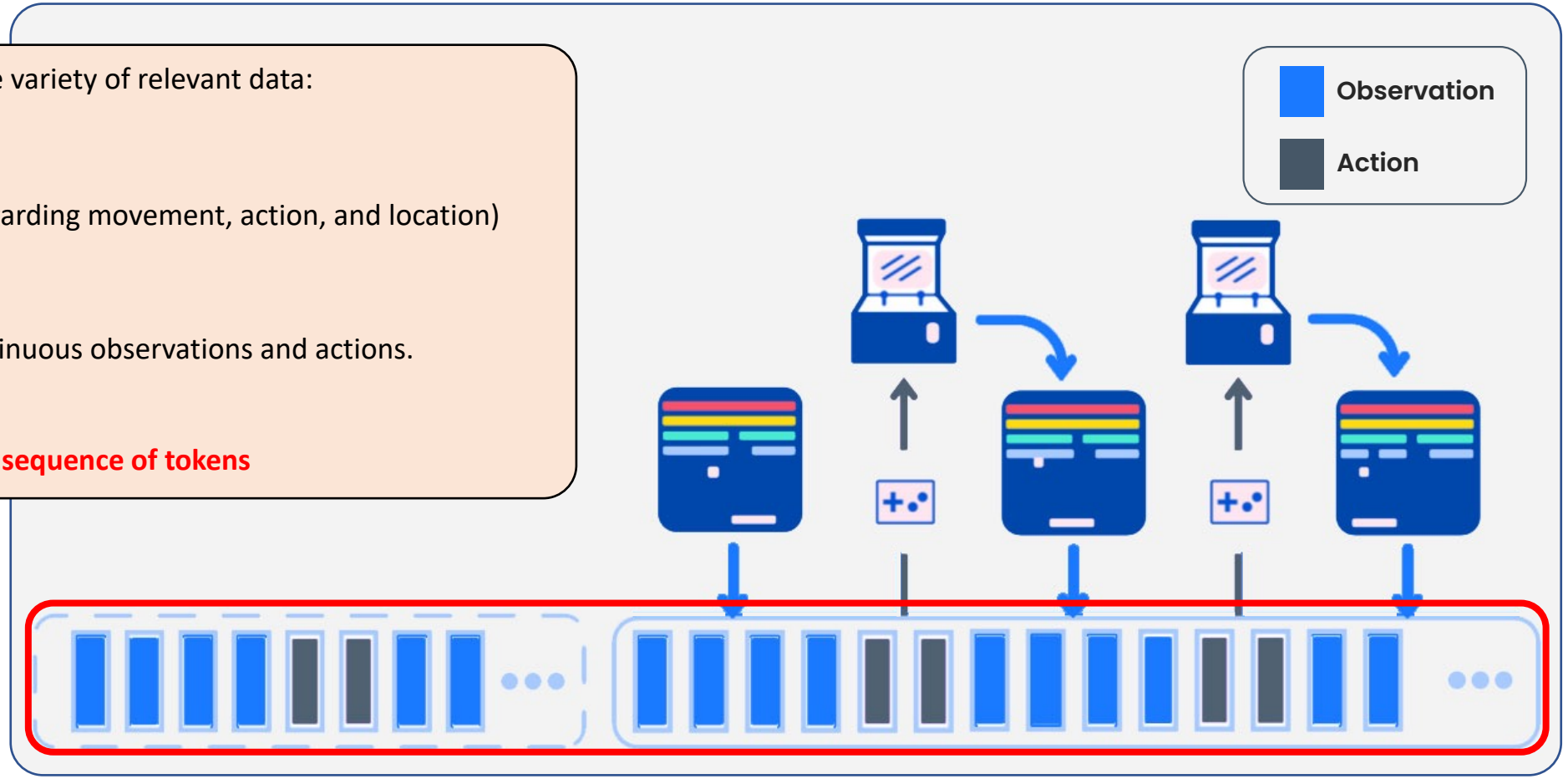
Multi task learning



Model application

General approach

- Trained on the widest possible variety of relevant data:
  - Images
  - Text
  - Proprioception (info regarding movement, action, and location)
  - Joint torques
  - Button presses
  - Other discrete and continuous observations and actions.
- All data is serialized into a flat **sequence of tokens**



High-level tokenization

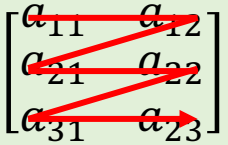
Sequencing of agent data

- **Episodes** are presented to the agent in order of time.
- **Each timestep** is presented in the order: **Observations** ($[y_{1:k}, x_{1:m}, z_{1:n}]$), **Separator** ('|'), **Actions** ($a_{1:A}$)
- A sequence of tokens is the concatenation of data from T timesteps:

$$s_{1:L} = \left[\left[y^1_{1:k}, x^1_{1:m}, z^1_{1:n}, \ '|', a^1_{1:A}\right], \ldots, \left[y^T_{1:k}, x^T_{1:m}, z^T_{1:n}, \ '|', a^T_{1:A}\right]\right]$$

- **Observations** ($[y_{1:k}, x_{1:m}, z_{1:n}]$) are ordered according to a key, each item is sequenced as follows:
  - **Text tokens** ($y_{1:k}$) are in the same order as the raw input text (with SentencPiece encoding).
  - **Image patch tokens** ($x_{1:m}$) are in **raster order.**
  - **Tensors** ($z_{1:n}$) (such as discrete and continuous observations) are in **row-major order.**
- **Separator** ('|'); a designated separator token is provided after observations.
- **Actions** ($a_{1:A}$) are tokenized as discrete or continuous values and in **row-major order.**

Row-major / raster order



Summarized state representations

Tokenization and embedding

Proprioception

$[0.3, 1.5, 0.25, -0.4]$

Action

$[-0.7]$

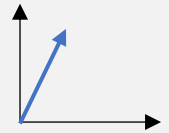
$\begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}, \begin{bmatrix}7.6\\5.3\\\vdots\end{bmatrix}, \begin{bmatrix}4.6\\3.4\\\vdots\end{bmatrix}$ $\quad \begin{bmatrix}9.0\\3.3\\\vdots\end{bmatrix}$

Action tokens always get same position encoding

Full sequence

$\ldots \begin{bmatrix}2.3\\8.3\\\vdots\end{bmatrix}, \begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}, \begin{bmatrix}7.6\\5.3\\\vdots\end{bmatrix}, \begin{bmatrix}4.6\\3.4\\\vdots\end{bmatrix}, \begin{bmatrix}9.0\\3.3\\\vdots\end{bmatrix}, \begin{bmatrix}2.3\\0.9\\\vdots\end{bmatrix} \ldots$

Mu-law encoding & vector embedding

$f(1.5) =$ 

Uses lookup table

Local position encoding

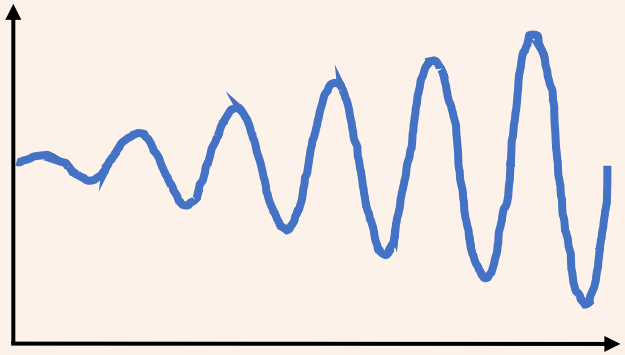
$$h\left(\begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}\right) = \begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix} + \boldsymbol{p_0}, \quad \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix} + \boldsymbol{p_1}$$

$e.g. \boldsymbol{p_0} = \begin{bmatrix}3.0\\5.8\\\vdots\end{bmatrix}$

Mu-law companding

Reducing dynamic range

Input signal



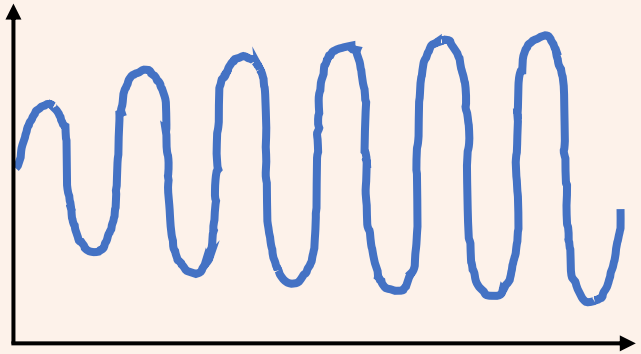
$\mu = 100$
$M = 256$

Preserves sign

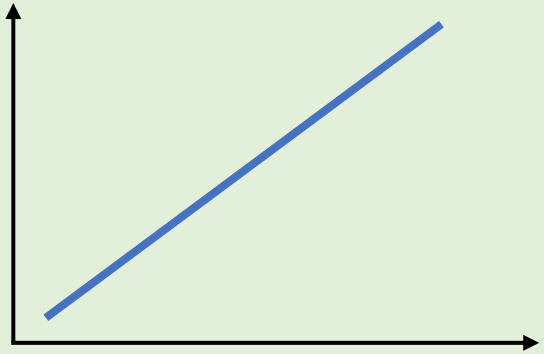
$$F(x) = \text{sgn}(x)\frac{\log(|x|\mu + 1.0)}{\log(M\mu + 1.0)}$$

Constant denominator

Reduced dynamic range



After mu-law companding



- Capping to [-1, 1]
- Discretization into 1024 bins

Processed signal

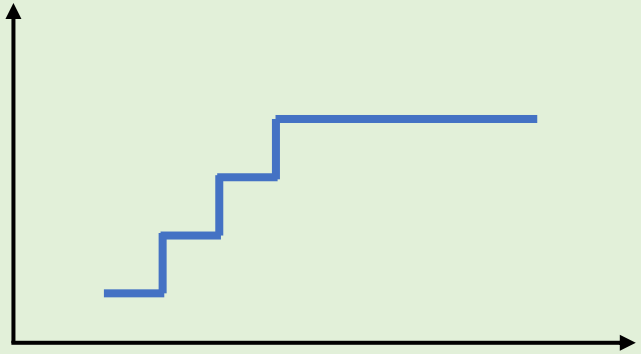
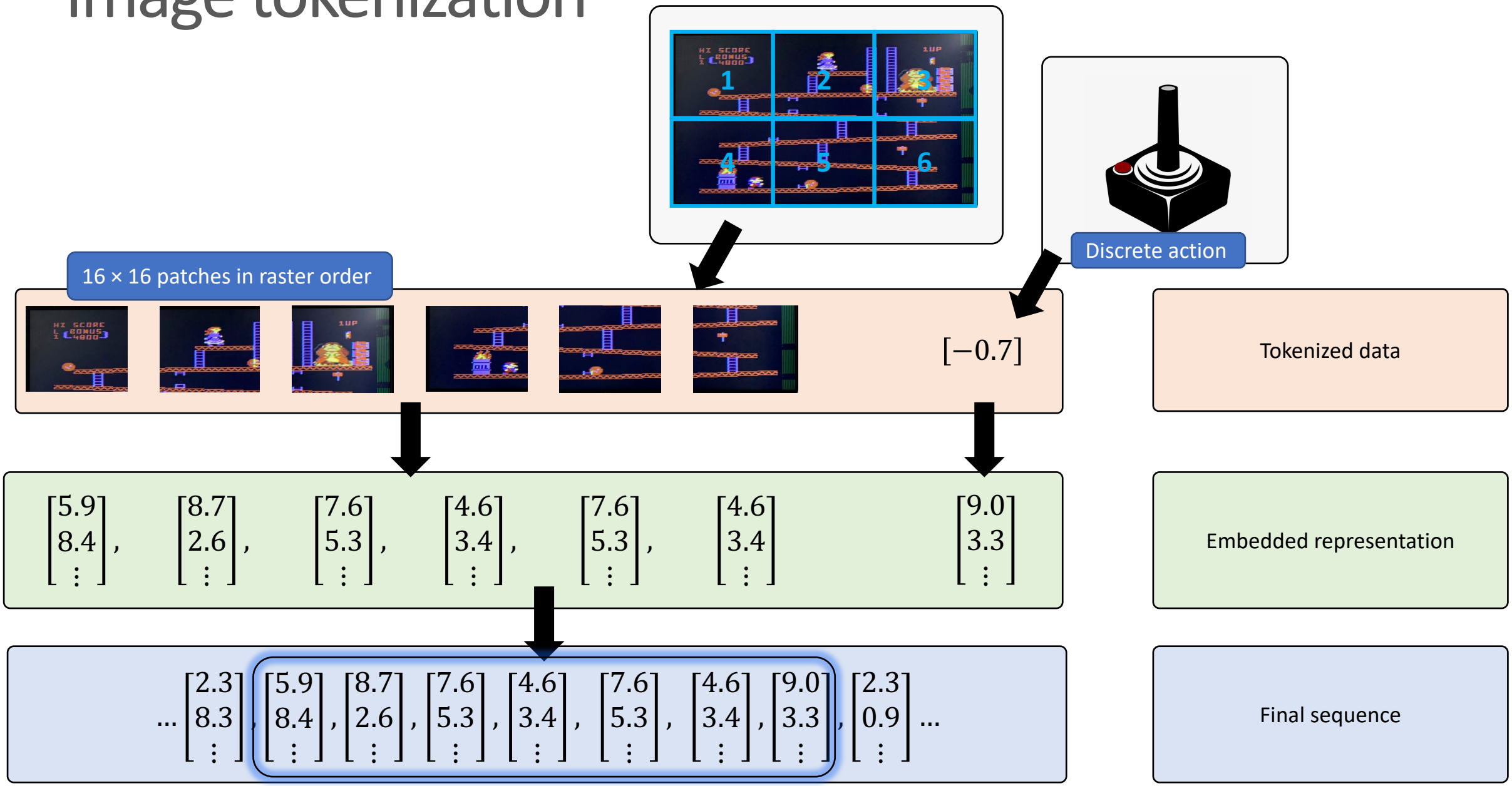
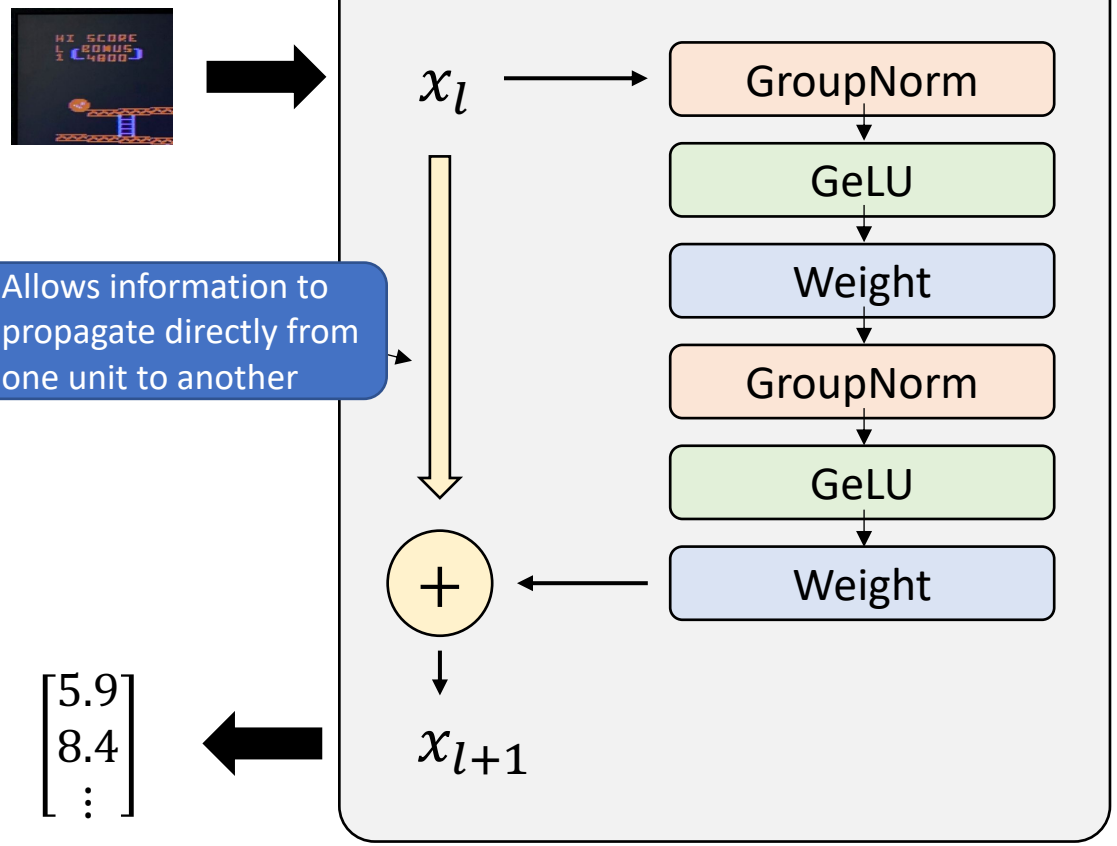


Image tokenization



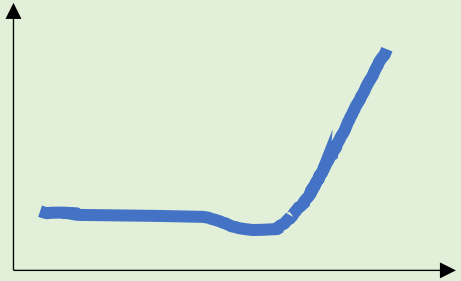
Converting tokenized image to embedding

ResNet architecture



- **GroupNorm** normalizes data to appropriate scales, similar to LayerNorm and InstanceNorm
- Good for distributed systems

- **GeLU** is an activation function.
- Provides well defined gradient in negative regime
- Prevents neurons from dying while bounding influence of negative regime activations



- The **weight** layers are configurable with hyper parameters
- Common to use small convolutional layers

- Multiple layers of these units can be stacked to increase complexity

2.2 Embedding input tokens and setting output targets

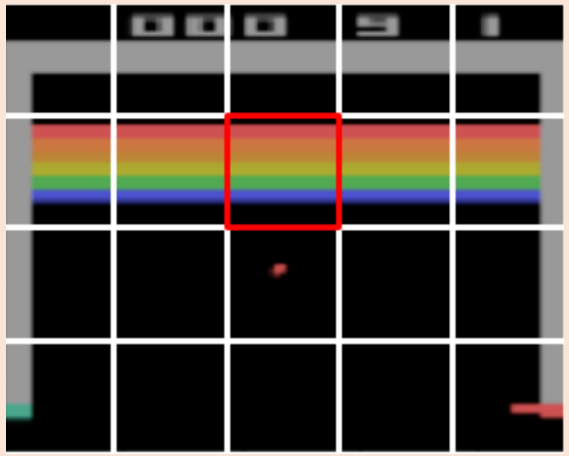
Patch position encodings: Input from image patches

- For image patch token embeddings, between the embedded image and mapping onto the final sequence, a learnable within-image position encoding vector is also added.
- Conveys information about a patch's global position within the image

Image sectioning

Width normalized to range [0, 128]

Height normalized to range [0, 128]



Finding coordinate ranges

$x \in [40\%, 60\%]$
$= [51.2, 76.8]$

$y \in [25\%, 50\%]$
$= [32, 64]$

Selecting encoding point

During training, the center point is used: $x = 64, y = 48$

During evaluation, a random point in the range is used