A Universal Language for Associative Memory

Many variants of energy-based AM

- **Classical Hopfield Network** (J. J. Hopfield 1982; J. Hopfield 1984)
- **Dense Associative Memory** (Krotov and Hopfield 2016)

  binary and continuous states, polynomial and exponential activations…

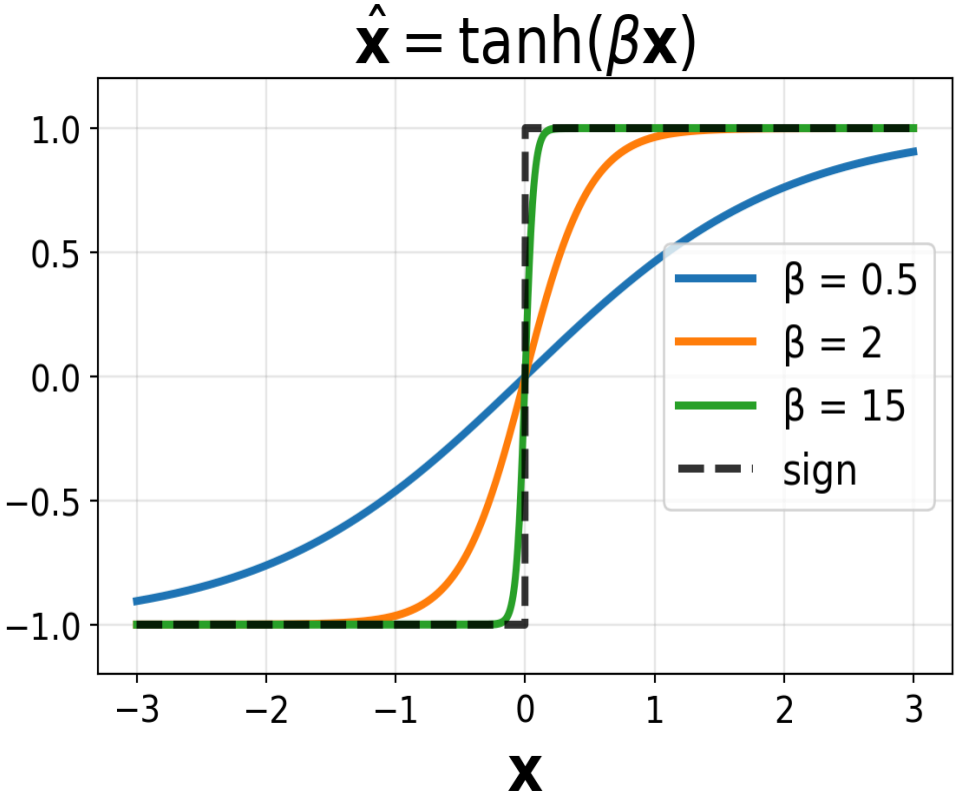
- **Hierarchical Associative Memory** (Krotov 2021)
- **Energy Transformer** (Hoover et al. 2023)
- **Neuron-Astrocyte Networks** (Kozachkov, Slotine, and Krotov 2023)

Problem 1

> Can we develop a language that works for both **continuous and binary** states?

From continuous to binary

Take limit as $\beta \to \infty \ldots$



Binary states use **energy flips**

$$\hat{x}_i^{(t+1)} = \underset{b \in \{-1,1\}}{\operatorname{argmin}} E(\hat{x}_i = b, \hat{x}$$

Continuous states use **gradient flow**

$$\tau \frac{dx_i}{dt} = -\frac{\partial E}{\partial \hat{x}_i}$$

Problem 2

> How to unify different non-linearities?

- $\hat{\mathbf{x}} = \tanh(\mathbf{x})$
- $\hat{\mathbf{x}} = \sigma(\mathbf{x})$
- $\hat{\mathbf{x}} = \mathrm{ReLU}(\mathbf{x})$
- $\hat{\mathbf{x}} = \mathrm{ELU}(\mathbf{x})$
- $\hat{\mathbf{x}} = \mathrm{softmax}(\mathbf{x})$
- $\hat{\mathbf{x}} = \mathrm{layernorm}(\mathbf{x})$
- $\ldots$

They are all monotonically increasing functions of $\mathbf{x}$

Monotonic functions are gradients of convex functions.

Let **Lagrangian** $\mathcal{L}(\mathbf{x})$ be the convex function whose gradient is our activation function i.e., $\nabla\mathcal{L}(\mathbf{x}) = \hat{\mathbf{x}}$.

We like convex functions.

$\mathbf{x}$ and $\hat{\mathbf{x}}$ are conjugate variables

$\hat{\mathbf{x}} := \nabla \mathcal{L}_x(\mathbf{x})$ describes the conjugate pair $(\mathbf{x}, \hat{\mathbf{x}})$ under the **Legendre Transform** of $\mathcal{L}_x$

$$\mathcal{T}[\mathcal{L}_x] = E_x = \langle \mathbf{x}, \hat{\mathbf{x}} \rangle - \mathcal{L}_x(\mathbf{x})$$

The transformed Lagrangian is the **energy** of an activation.

$\mathbf{x}$ and $\hat{\mathbf{x}}$ evolve together to minimize the energy E.

> ⓘ **Without stored patterns, this is not an interesting AM**
>
> Think of this energy as a *regularization term* applied during our dynamics.

Its effect is to add exponential decay to our dynamics.

$$\begin{aligned}\tau \frac{dx_i}{dt} &= -\frac{\partial E_x}{\partial \hat{x}_i} + \dots \\ &= -x_i + \dots\end{aligned}$$

Problem 3

> Obey all necessary properties of an AM

Necessary properties of an AM

Energy E is **continuous everywhere** and **bounded from below**

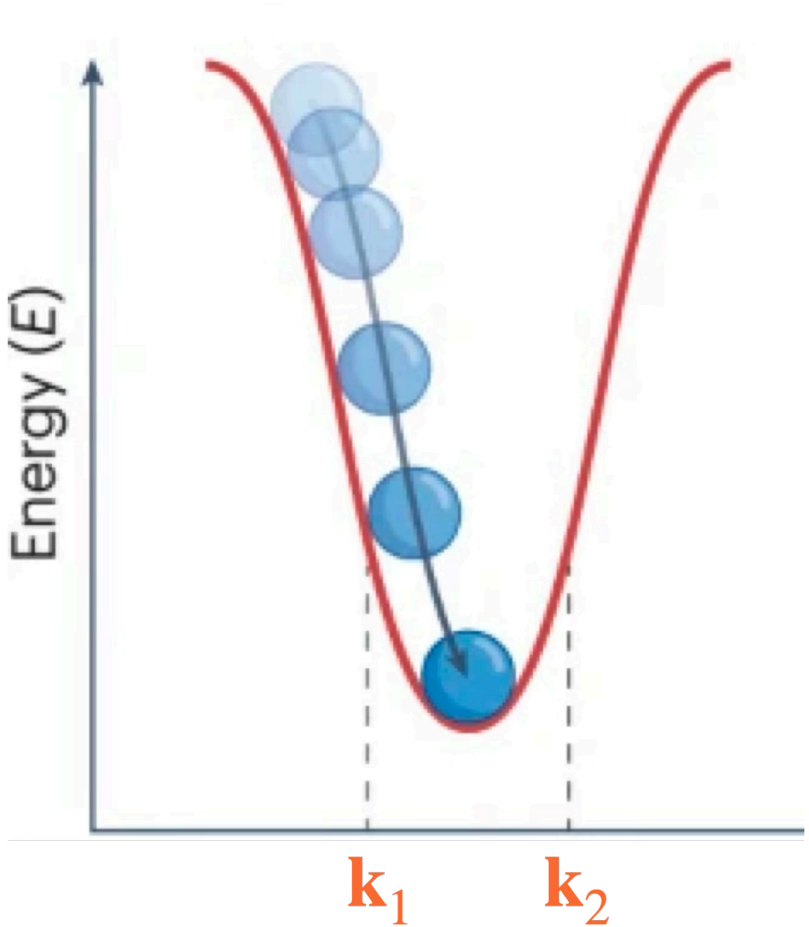
1. Dynamics via gradient flow **never increase energy**

$$\tau \frac{d\mathbf{x}}{dt} = -\nabla E \implies \frac{dE}{dt} \leq 0$$

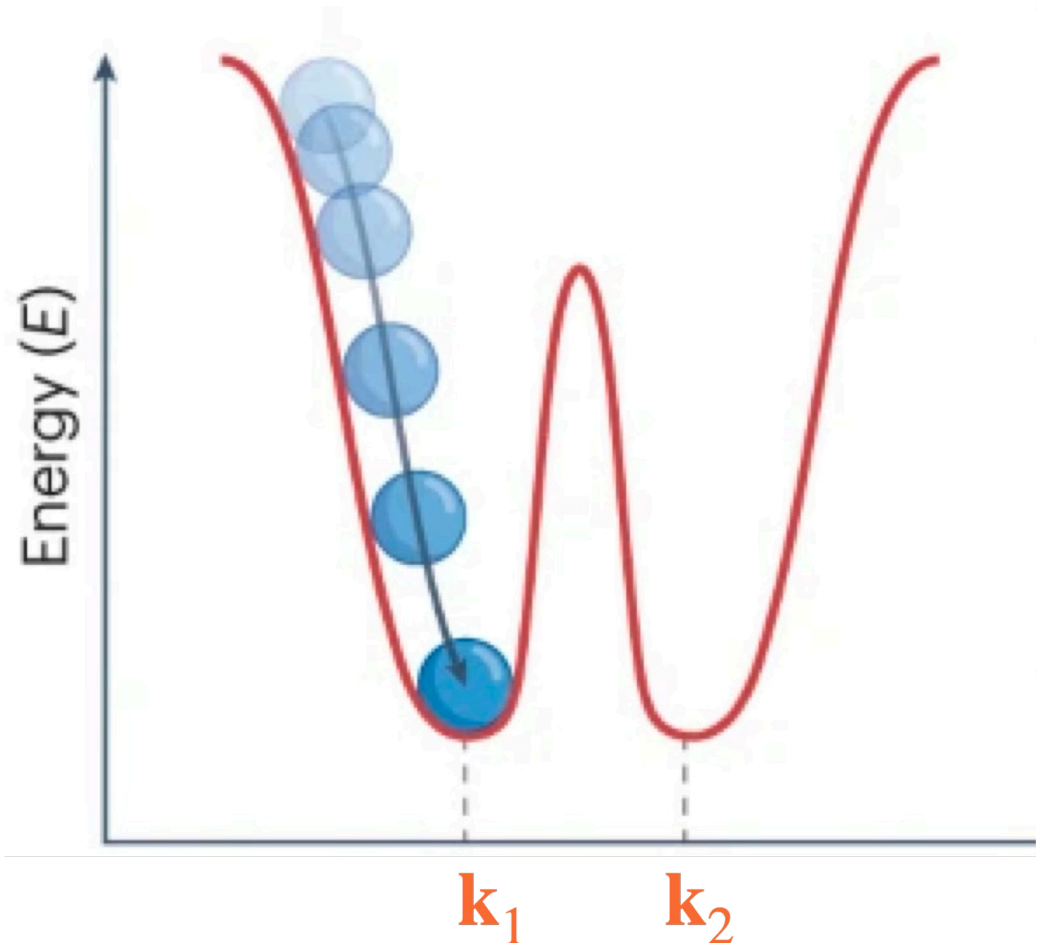
2. Bounded energy means **guaranteed convergence**

$$\nabla E = 0 \implies \frac{d\mathbf{x}^*}{dt} = 0 \implies \mathbf{x}^* \text{ is a } \mathbf{memory}$$

Classical Hopfield



Dense AM



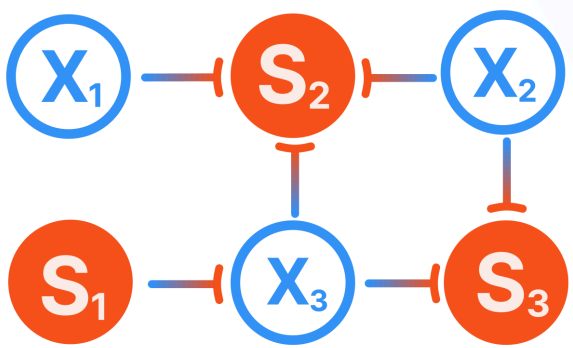
Problem 4: Adding “hierarchy”

- We want **multiple nonlinearities**
- We want **arbitrarily complex** parameterizations
- We want to **scale**

Associative Memory

*A hypergraph of **neurons** communicating via **synapses***

One total energy
Local computation
Guaranteed convergence



$$E_{\text{total}} = \sum_{\text{neurons}} E + \sum_{\text{synapses}} E$$

$$\frac{d\mathbf{x}_\ell}{dt} = -\frac{\partial E_{\text{total}}}{\partial \hat{\mathbf{x}}_\ell}$$

Neuron Layer

Simplify non-linear dynamics using Lagrangians

Legendre Transform of the Lagrangian defines both **activations** and layer's **energy**

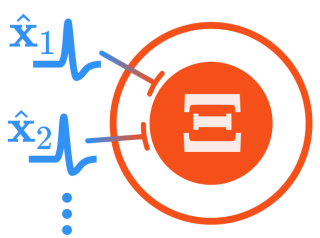
$$\hat{\mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \qquad E = \langle \mathbf{x}, \hat{\mathbf{x}} \rangle - \mathcal{L}$$

Dynamic **states** minimize energy of activations

$$\frac{d\mathbf{x}}{dt} = -\frac{\partial E}{\partial \hat{\mathbf{x}}}$$

Hypersynapse

Learn to align activations of connected neurons



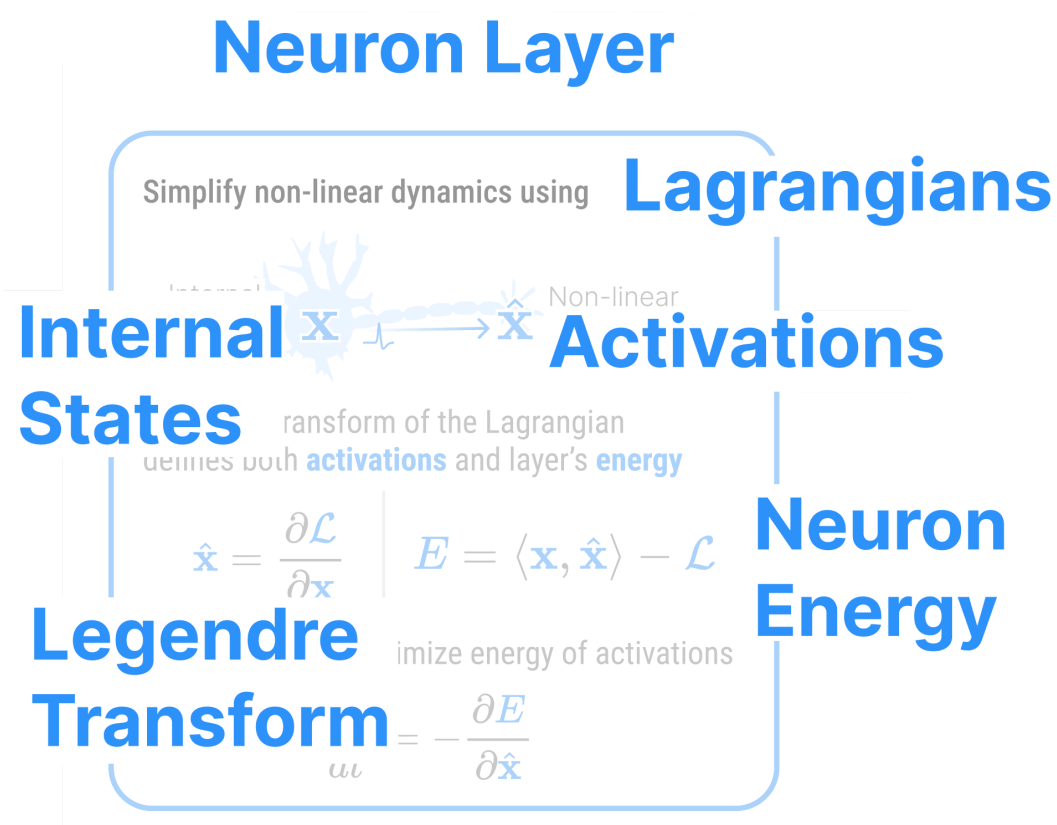
Energy encodes activation patterns in **synaptic weights** Ξ

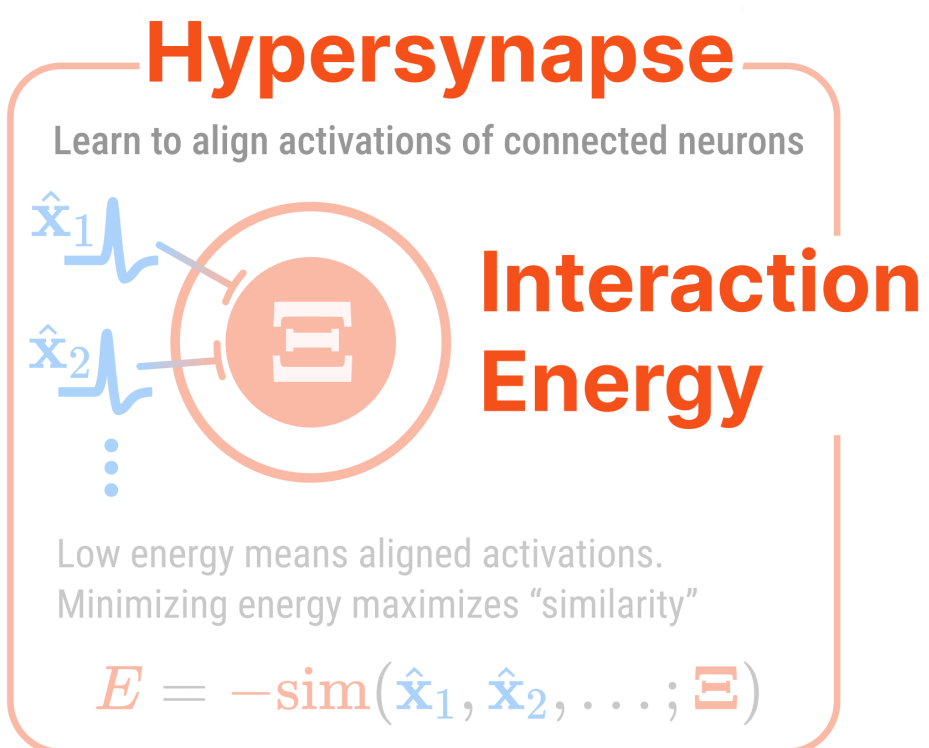
Low energy means aligned activations.
Minimizing energy maximizes "similarity"

$$E = -\text{sim}(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots; \Xi)$$

Local computation minimizes a global energy.

We call this framework **HAMUX**.





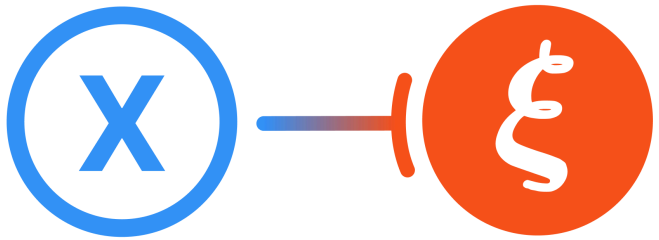
Patterns are stored in the **interaction energies** of **hypersynapses**.

Building AMs with HAMUX

Continuous Hopfield Nets in HAMUX (J. Hopfield 1984)

One dynamic state vector

One matrix of stored patterns



E_x E_{syn}

$$E = E_x + E_{\text{syn}}$$

Given $\mu = 1 \dots K$ stored patterns of dim $i = 1 \dots D$

Decide activations and interactions:

$$\hat{x}_i = \tanh(\beta x_i)$$

$$E_{\text{syn}} = -\frac{1}{2} \sum_{\mu} \left(\sum_{i} \xi_i^{\mu} \, \hat{x}_i \right)^2$$

We get $E_x = \mathcal{T}[\mathcal{L}_x]$ *for free.*

The CHN total energy becomes

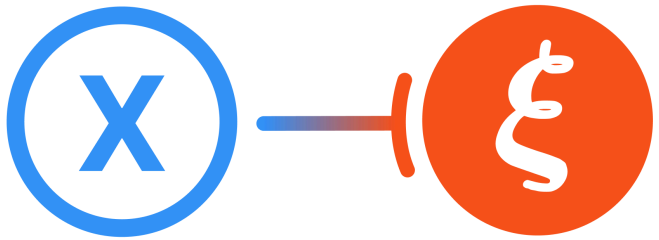
$$E = E_x + E_{\text{syn}}$$

The neg. gradient of E is the

Dense AM (Krotov and Hopfield 2016)

One dynamic state vector

One matrix of stored patterns



E_x E_{syn}

$$E = E_x + E_{\text{syn}}$$

Given $\mu = 1 \dots K$ stored patterns of dim $i = 1 \dots D$

Activations are the same as CHN.

Define new interaction energy

$$\hat{x}_i = \tanh(\beta x_i)$$

$$E_{\text{syn}} = -\frac{1}{n} \sum_{\mu} \text{ReLU}\left(\sum_i \xi_i^\mu \, \hat{x}_i \right)^n$$

The DenseAM total energy is

$$E = E_x + E_{\text{syn}}$$

The neg. gradient of E is the DenseAM update rule

$$\frac{dx_i}{dt} = -\frac{\partial E}{\partial \hat{x}_i}$$

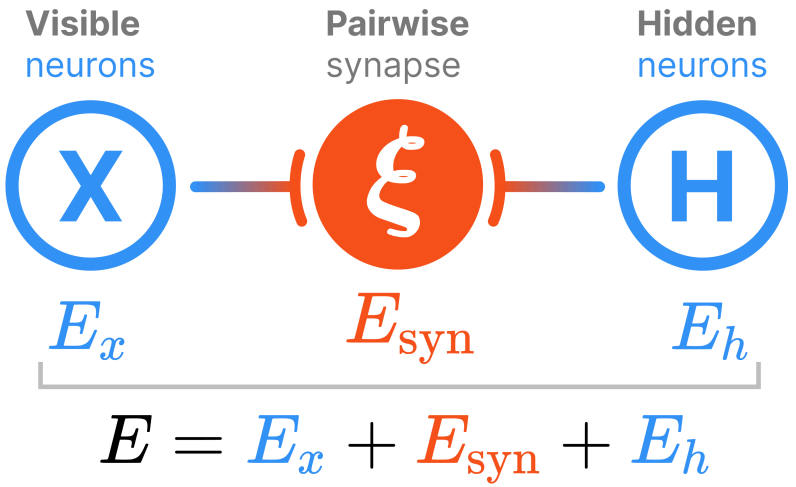
Biologically Plausible DenseAM (Krotov and Hopfield 2021)

$$E_{\text{oldsyn}} = -\frac{1}{n}\sum_{\mu} \text{ReLU}\left(\xi_i^{\mu}\ \hat{x}_i\right)^{n}$$

Problem. $n > 2$ describes non-pairwise synapses. *Not biologically implementable.*

💡 **Solution**. Absorb the polynomial into new activations.

Biologically Plausible DenseAM (Krotov and Hopfield 2021)



Introduce hidden neurons h_μ!

Decide activations for new neurons

$$\hat{h}_\mu = \text{ReLU}(h_\mu)^{n-1}$$

Use pairwise interaction energy

$$E_{\text{syn}} = -\sum_{i,\mu} \hat{h}_\mu \xi_i^\mu \hat{x}_i$$

We get $E_h = \mathcal{T}[\mathcal{L}_h]$ *for free.*

Total energy is now

$$E = E_x + E_{\text{syn}} + E_h$$

Two states evolve together to

minimize the same total energy!

$$\begin{cases} \dfrac{dx_i}{dt} = -\dfrac{\partial E}{\partial \hat{x}_i} \\ \dfrac{dh_\mu}{dt} = -\dfrac{\partial E}{\partial \hat{h}_\mu} \end{cases}$$

Update rule becomes

$$\begin{cases} \dfrac{dx_i}{dt} = -x_i + \displaystyle\sum_\mu \xi_i^\mu \hat{h}_\mu \\ \dfrac{dh_\mu}{dt} = -h_\mu + \displaystyle\sum_i \xi_i^\mu \hat{x}_i \end{cases}$$

Exercise

Prove that these systems are equivalent

$$\begin{cases} \dfrac{dx_i}{dt} = -x_i + \sum_\mu \xi_i^\mu \hat{h}_\mu \\ \dfrac{dh_\mu}{dt} = \sum_i \xi_i^\mu \hat{x}_i - h_\mu \end{cases}$$

$$= \begin{cases} \dfrac{dx_i}{dt} = -x_i + \sum_\mu \xi_i^\mu \mathrm{ReLU}(\sum_j \xi_j^\mu \hat{x}_j)^{n-1} \end{cases}$$

Solution

Integrate out h_μ to find fixed point h_μ^* as a function of $\hat{x}_i$.

$$\frac{dh_\mu^*}{dt} = 0 = \sum_i \xi_i^\mu \hat{x}_i - h_\mu^*$$

Thus

$$h_\mu^* = \sum_i \xi_i^\mu \hat{x}_i$$

$$\hat{h}_\mu^* = \mathrm{ReLU}(h_\mu^*)^{n-1}$$

The polynomial is now captured in activations of integrated-out hidden neurons

$$\frac{dx_i}{dt} = \sum_\mu \xi_i^\mu \hat{h}_\mu^* - x_i$$

$$= \sum_\mu \xi_i^\mu \mathrm{ReLU}(\sum_j \xi_j^\mu \hat{x}_j)^{n-1} - x_i$$

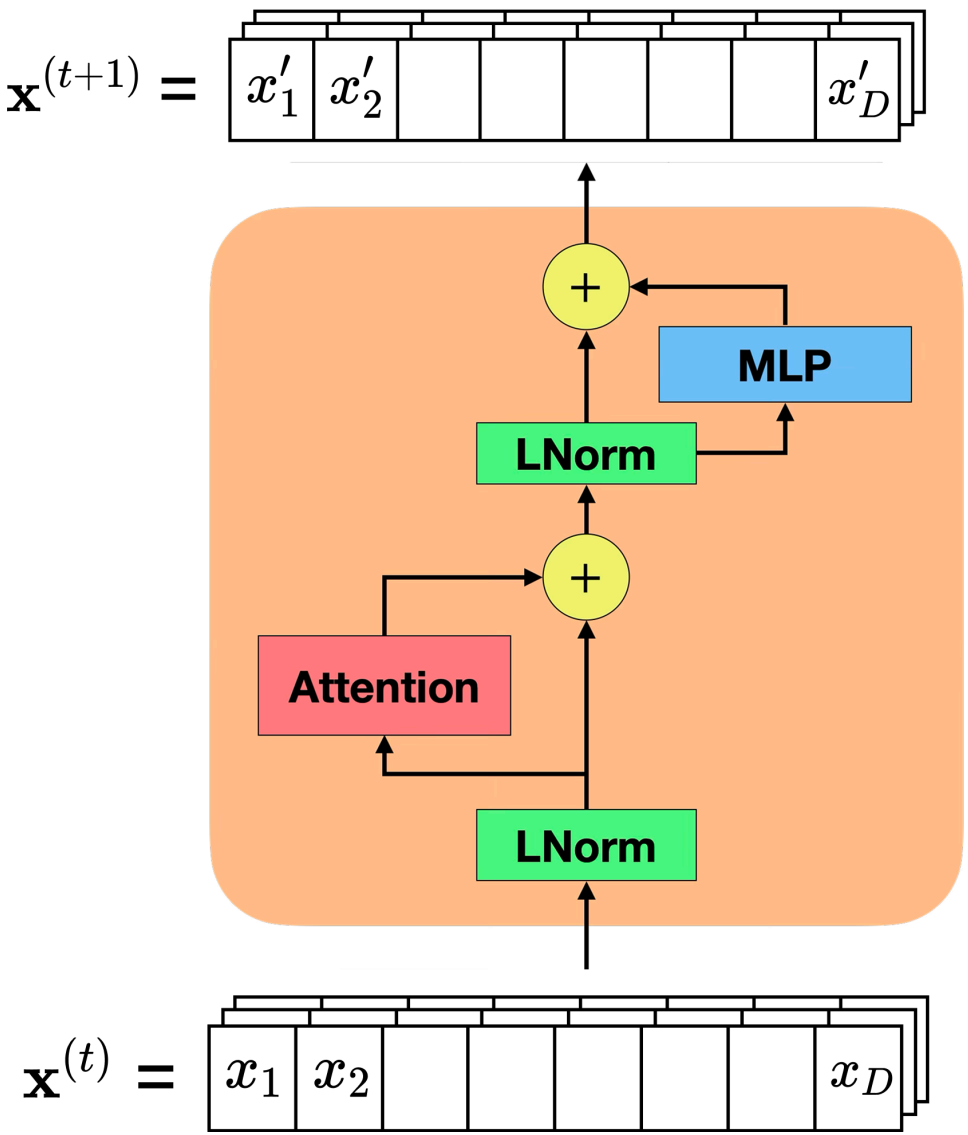
This language for AMs helped us build

- Original Hopfield Network
- Dense Associative Memory
- Biologically plausible DenseAM

But can it help us invent something new?

Energy Transformer

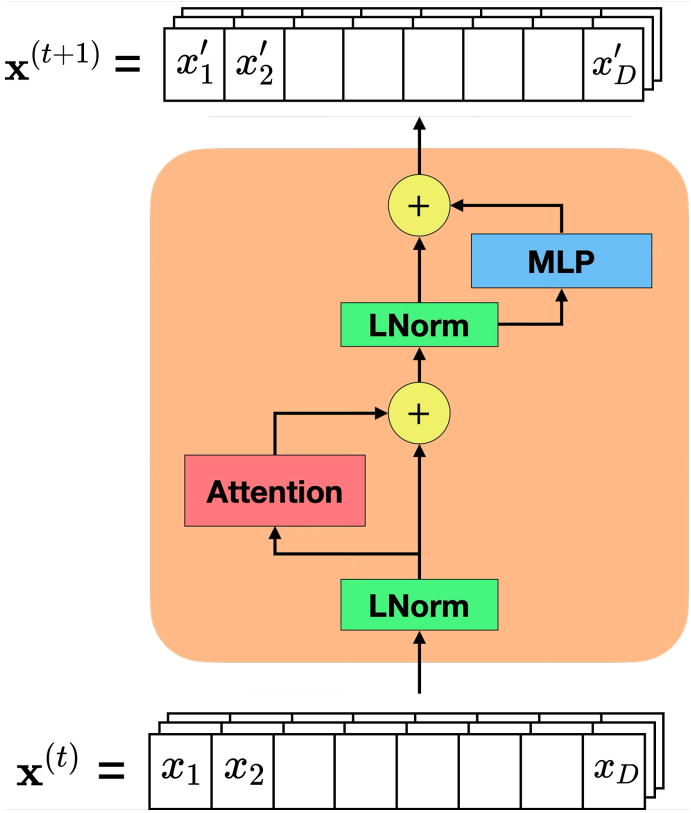
Deriving the Transformer as AM



Four components

1. LayerNorm
2. Attention
3. Skip Connections
4. MLP

HAMUX-ify the Transformer Block



What is our dynamic state?

- Tokens $\mathbf{x}$

Does it have a non-linear activation?

- $\hat{\mathbf{x}}$ = LayerNorm($\mathbf{x}$) ✅

Is the activation monotonic?

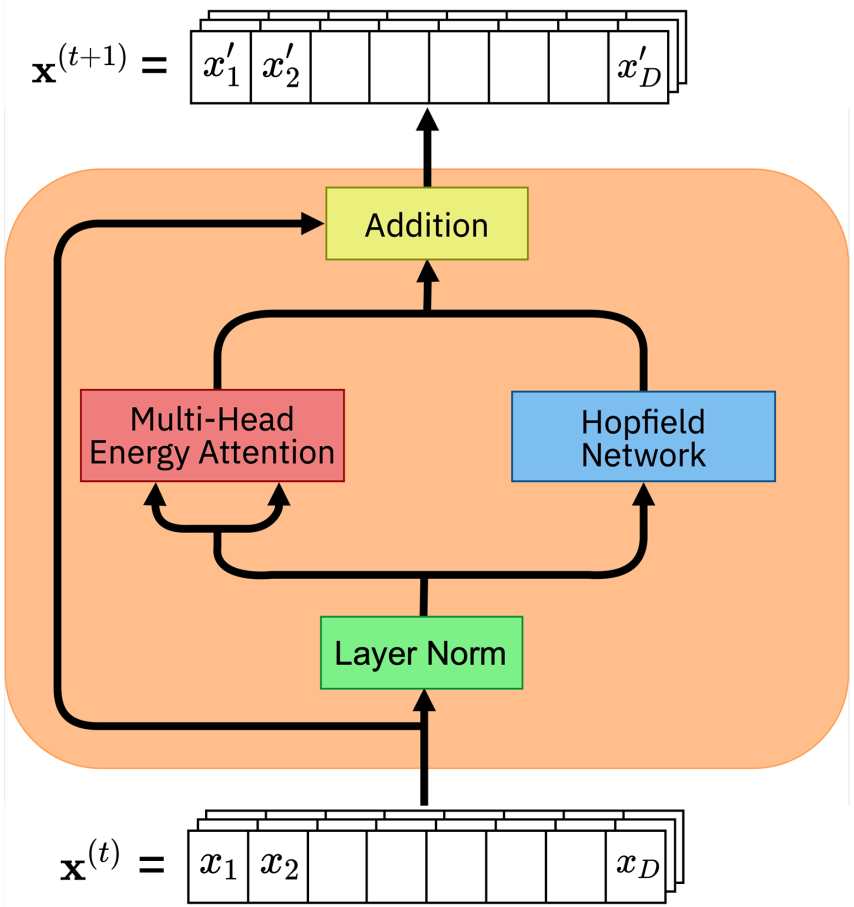
- ✅

$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + F(\hat{\mathbf{x}}^{(t)})$?

- F_{attn} and F_{mlp} both same shape as $\mathbf{x}$

If both F_{attn} and F_{mlp} could be written

Energy Transformer



ET Update Block

Full Walkthrough

Multi-head **Attn Energy** | Dynamic **token states** | **Hopfield Network**

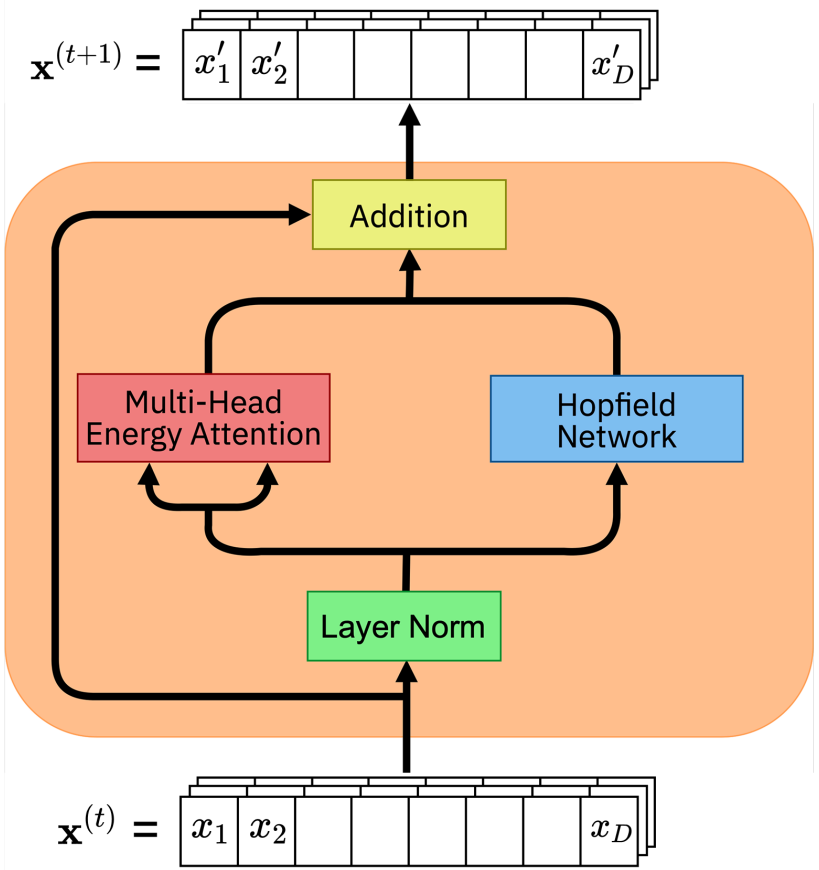
ATTN — X — MLP

E_{ATTN} E_x E_{HN}

$$E = E_{\text{attn}} + E_x + E_{\text{HN}}$$

Energy Transformer

Full Walkthrough



ET Update Block

Differences

- **Attention** and **MLP** are in parallel
- **Residual** over pre-activations
- **Shared weights** through “layers”
- Everything is an **energy**

Energy Attention

> Make queries and keys align in latent space

N tokens, each with dim D.

H heads, each projecting to head dim Y.

Two parameters: $\mathbf{W}^Q$ and $\mathbf{W}^K$ in (H, D, Y).

```python
def attn_energy(xhat, Wq, Wk):
    K = jnp.einsum("kd,hdy->khy", xhat, Wk)
    Q = jnp.einsum("qd,hdy->qhy", xhat, Wq)
    A = jnp.einsum("khy,qhy->hqk", K, Q)
    beta = 1 / jnp.sqrt(Wq.shape[-1])
    return -1 / beta * jax.nn.logsumexp(beta * A, axis=-1).sum()
```

Hopfield Network

> Make each token look like a stored pattern

M stored patterns, each with dim D.

One parameter: Memory matrix Ξ in (M, D).

```
def hn_energy(xhat, Xi):
  hid = jnp.einsum("nd,md->nm", xhat, Xi)
  return -0.5 * (hid.clip(0) ** 2).sum()
```

LayerNorm

$$\hat{\mathbf{x}} = \gamma \left(\frac{\mathbf{x} - E[\mathbf{x}]}{\sqrt{Var[\mathbf{x}] + \epsilon}} \right) + \boldsymbol{\delta}$$

Vanilla LayerNorm, with one difference:

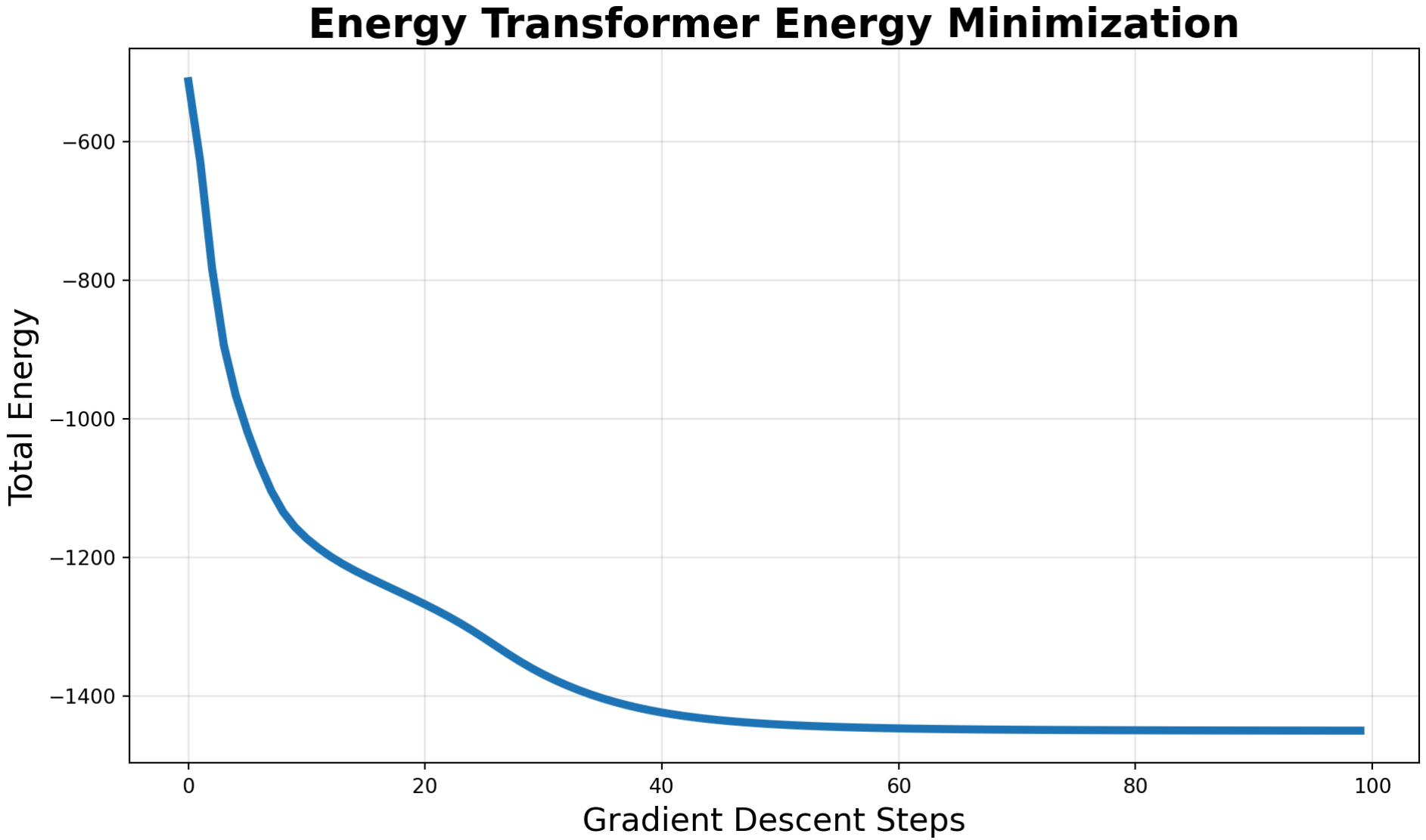
γ is a scalar. This way we can ensure it has an energy.

```
def layer_norm(x, gamma=1., delta=0., eps=1e-5, axis=-1):
    return gamma * (x - jnp.mean(x, axis=axis, keepdims=True)) / jnp.sqrt(j

def layer_norm_lagrangian(x, gamma=1., delta=0., eps=1e-5, axis=-1):
    D = x.shape[-1]
    xmeaned = x - x.mean(axis=axis, keepdims=True)
    t1 = D * gamma * jnp.sqrt((1 / D * xmeaned**2).sum(axis=axis, keepdims=
    t2 = (delta * x).sum(axis=axis, keepdims=True)
```

```
    return (t1 + t2).sum()

layer_norm_energy = legendre_transform(layer_norm_lagrangian)
```

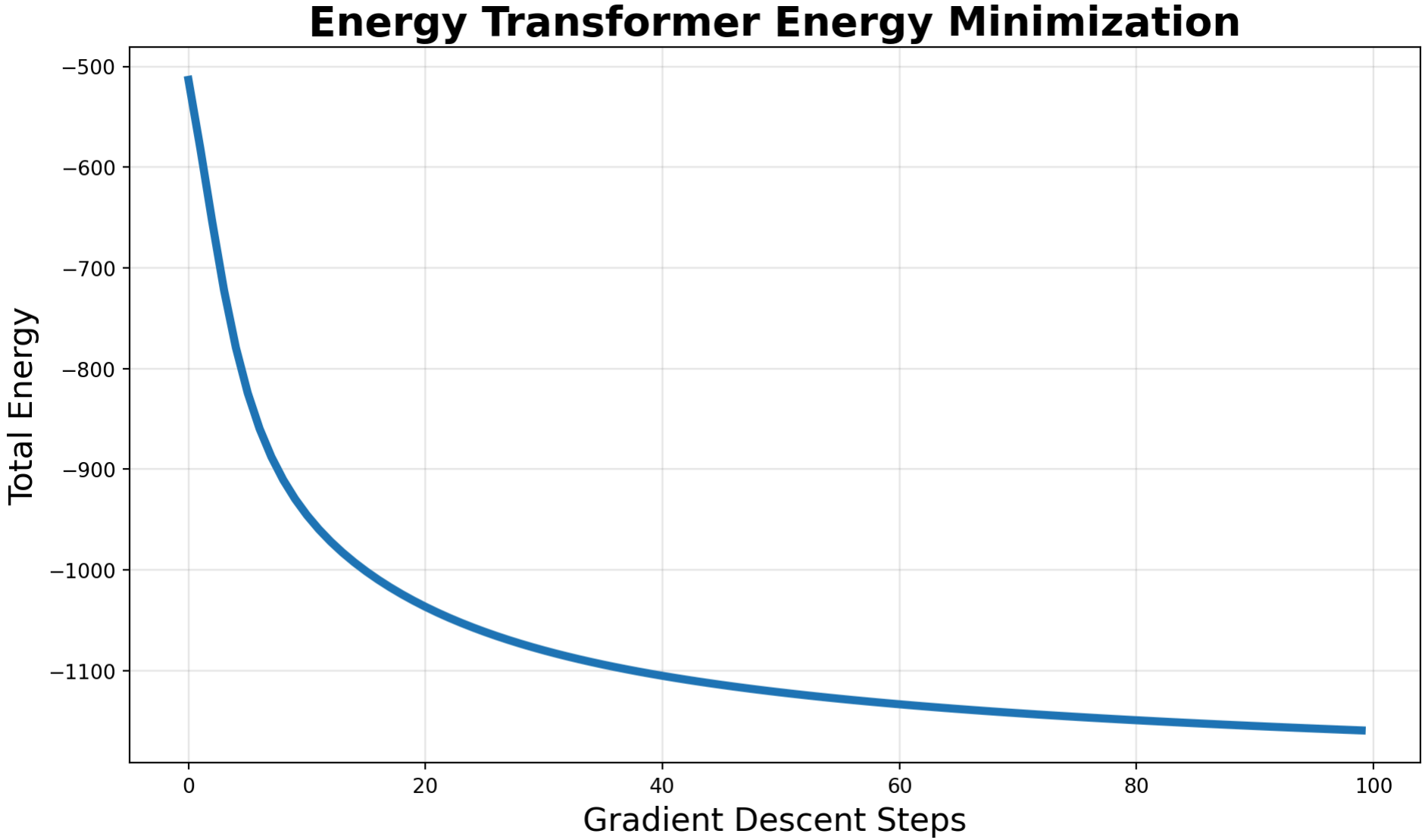
“Forward pass” through ET

```
N_layers = 100
step_size = 0.5
x0 = jr.normal(jr.PRNGKey(0), (N, D))

def synapse_energy(xhat, Wq, Wk, Xi):
    return attn_energy(xhat, Wq, Wk) + hn_energy(xhat, Xi)

def total_energy(xhat, x, Wq, Wk, Xi):
    return layer_norm_energy(xhat, x) + synapse_energy(xhat, Wq, Wk, Xi)

x = x0
energies = []
for i in range(N_layers):
    xhat = layer_norm(x)
    E, dE = jax.value_and_grad(total_energy)(xhat, x, Wq, Wk, Xi)
    x = x - step_size * dE
    energies.append(float(E))
```

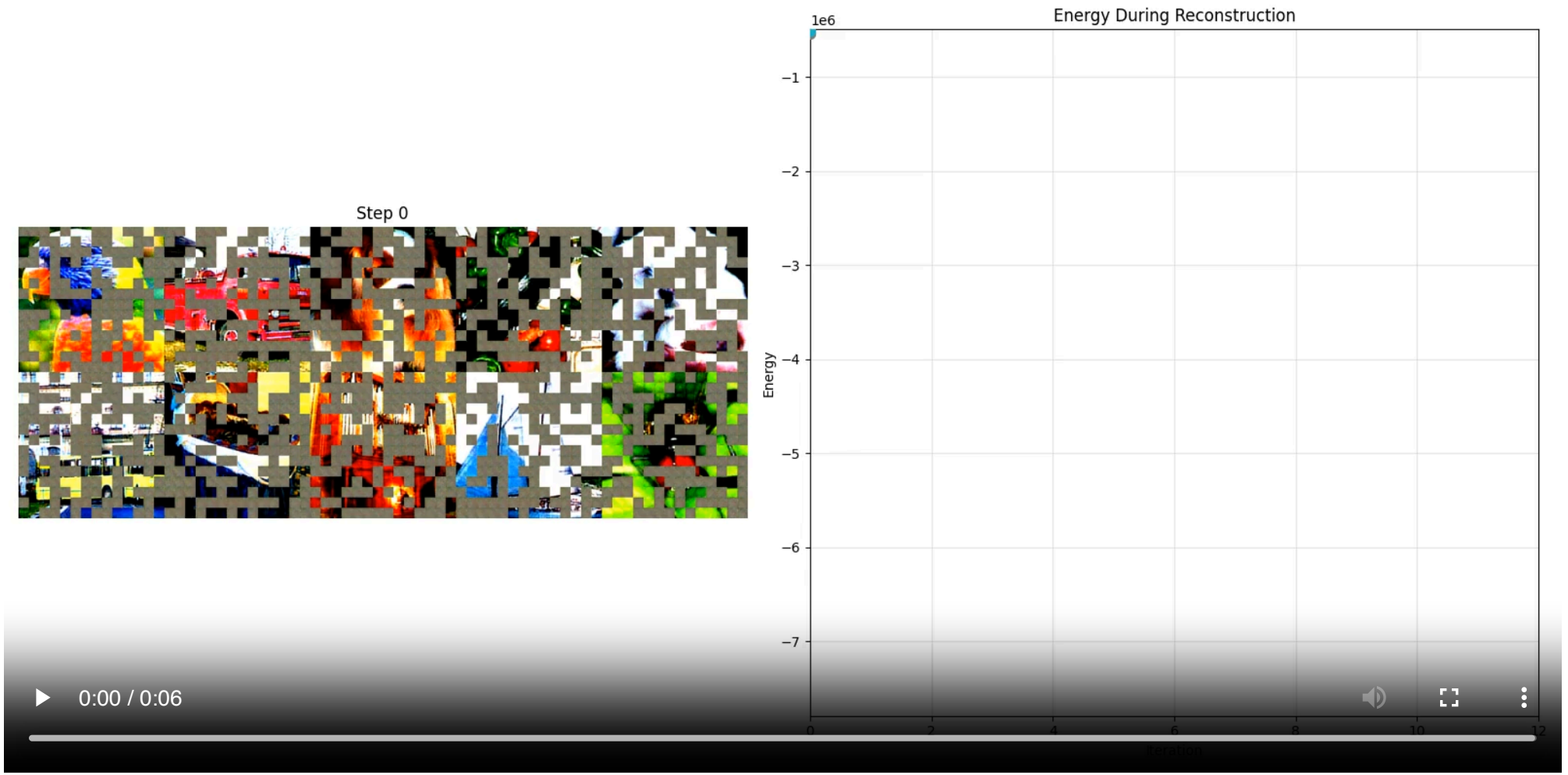

Energy Transformer Energy Minimization
Total Energy
−600
−800
−1000
−1200
−1400
0
20
40
60
80
100
Gradient Descent Steps

“Forward pass” without exponential decay

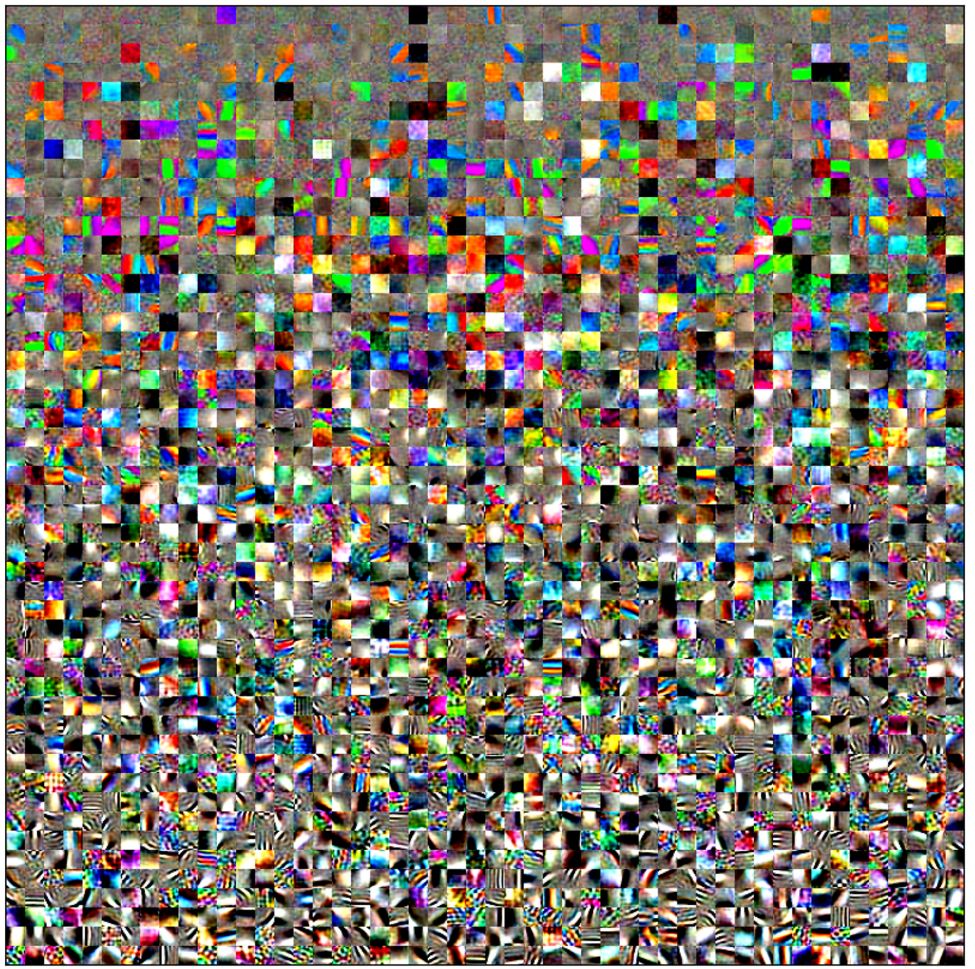
```
N_layers = 100
step_size = 0.5
x0 = jr.normal(jr.PRNGKey(0), (N, D))

def synapse_energy(xhat, Wq, Wk, Xi):
    return attn_energy(xhat, Wq, Wk) + hn_energy(xhat, Xi)

total_energy = synapse_energy

x = x0
energies = []
for i in range(N_layers):
    xhat = layer_norm(x)
    E, dE = jax.value_and_grad(total_energy)(xhat, Wq, Wk, Xi)
    x = x - step_size * dE
    energies.append(float(E))
```


Energy Transformer Energy Minimization
−500
−600
−700
−800
−900
−1000
−1100
Total Energy
0
20
40
60
80
100
Gradient Descent Steps

ET error-corrects unseen, masked images



ET is interpretable by design



Visualizing Ξ in the Hopfield Net shows what patterns the model has stored.

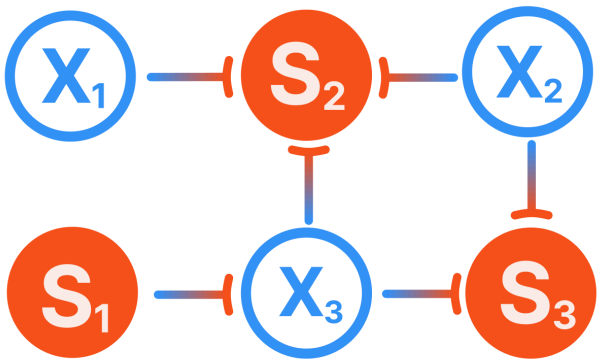
- Each weight is an **attractor**.
- Predicted patches are **linear combinations** of these attractors.
- Interpretability is a core computation of the model

> **Interpretability is a natural byproduct of good architecture design**

Associative Memory

*A hypergraph of **neurons** communicating via **synapses***

One total energy
Local computation
Guaranteed convergence

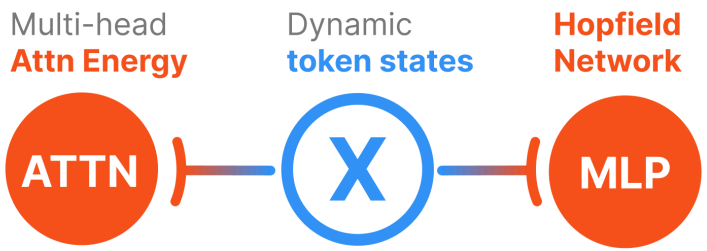


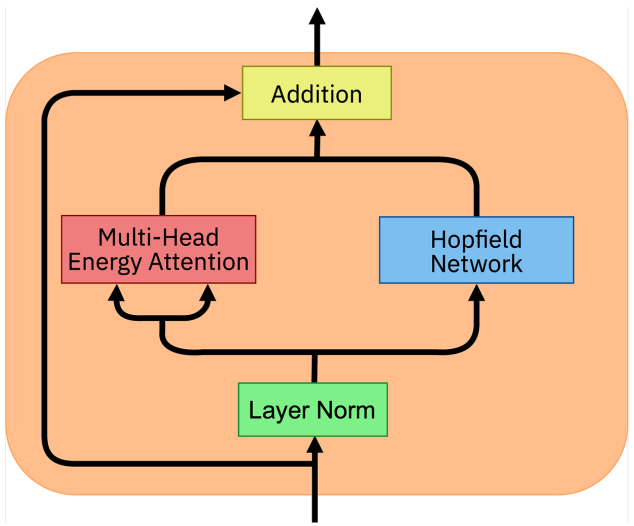
$$E_{\text{total}} = \sum_{\text{neurons}} E + \sum_{\text{synapses}} E$$

HAMUX documentation

Energy Transformer

Associative Memory meets Transformers





ET Walkthrough

Hoover, Benjamin, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. 2023. “Energy Transformer.” *Advances in Neural Information Processing Systems* 36. https://proceedings.neurips.cc/paper_files/paper/2023/file/57a9b97477b67936298-Paper-Conference.pdf.

Hopfield, John. 1984. “Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons.” *Proceedings of the National Academy of Sciences of the United States of America* 81 (June): 3088–92. https://doi.org/10.1073/pnas.81.10.3088.

Hopfield, John J. 1982. “Neural Networks and Physical Systems with Emergent Collective Computational Abilities.” *Proceedings of the National Academy of Sciences* 79 (8): 2554–58.

Kozachkov, Leo, Jean-Jacques Slotine, and Dmitry Krotov. 2023. “Neuron-Astrocyte Associative Memory.” *arXiv Preprint arXiv:2311.08135*.

Krotov, Dmitry. 2021. “Hierarchical Associative Memory.” https://arxiv.org/abs/2107.06446.

Krotov, Dmitry, and John J Hopfield. 2016. “Dense Associative Memory for Pattern Recognition.” *Advances in Neural Information Processing Systems* 29.

———. 2021. “Large Associative Memory Problem in Neurobiology and Machine Learning.” In *International Conference on Learning Representations*.