Learning Useful Representations of Recurrent Neural Network Weight Matrices

Vincent Herrmann, Francesco Faccio, Jürgen Schmidhuber

Istituto Dalle Molle di studi sull'intelligenza artificiale
IDSIA

Università della Svizzera italiana

Scuola universitaria professionale della Svizzera italiana
SUPSI

KAUST

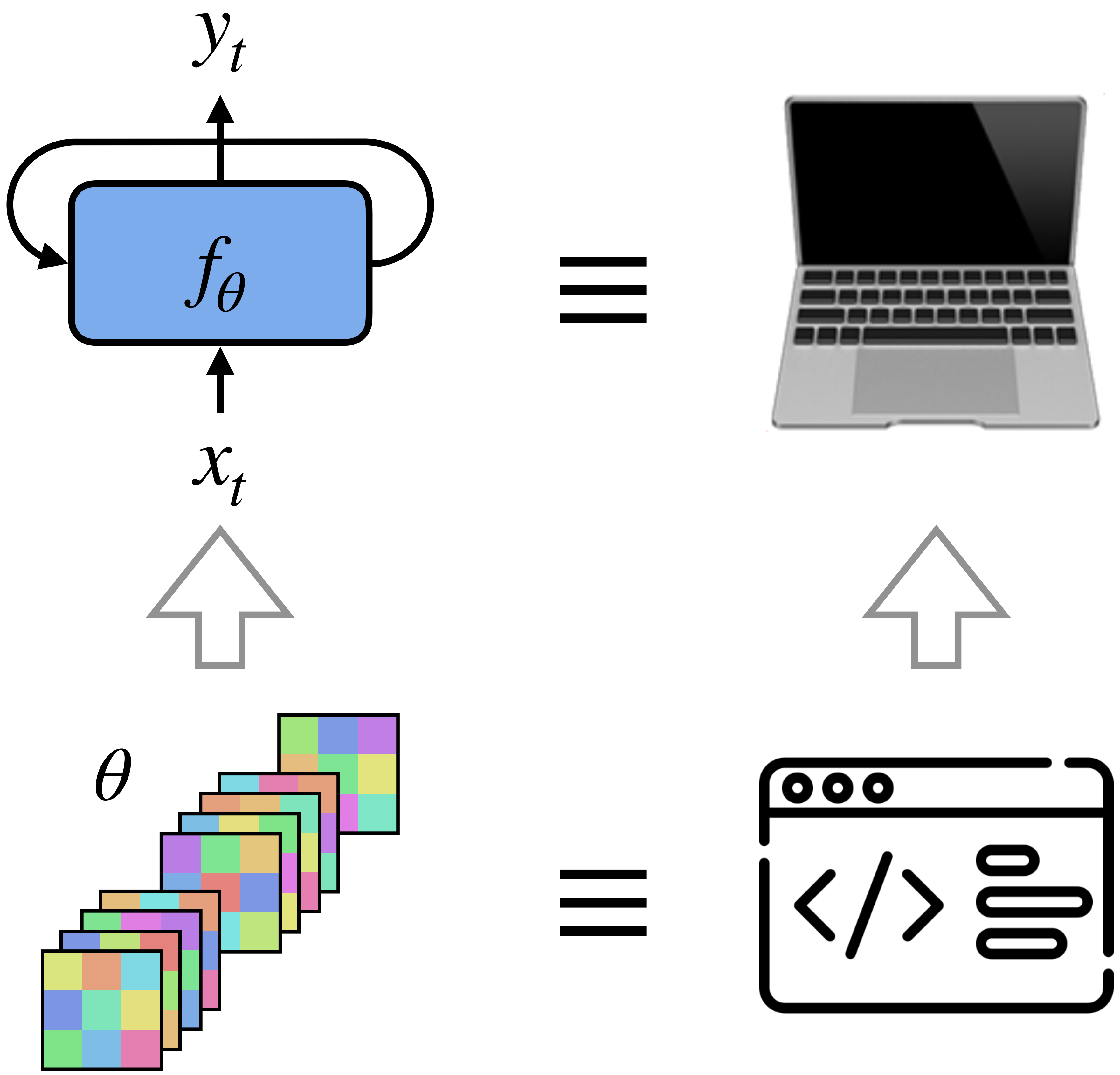
ICML 2024

Learning RNN Weight Representations

- RNNs are universal computers
- Weight matrices are their program

Applications:

- Reinforcement Learning
- Implicit Neural Representations
- Interpretability
- …



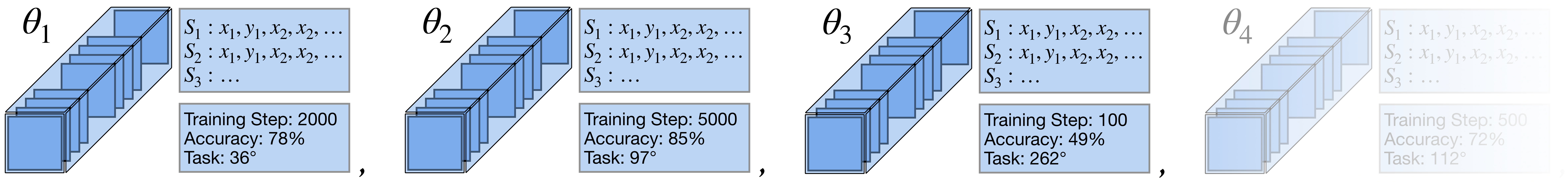
Three Ingredients

- Datasets
- Learning / Pre-Training method
- RNN Weight encoder architecture

Datasets

Two RNN 'Model Zoos'

- 1000 LSTM training runs per dataset, 9 snapshots per run
- Each LSTM is trained on a different task from a task family
- Datapoint: LSTM weights θ, 100 rollouts/trajectories, metadata



Formal Languages Dataset

- Autoregressive models of 216 different formal languages
- Language family: $L_{m_a,m_b,m_c,m_d} := \{a^{n+m_a}b^{n+m_b}c^{n+m_c}d^{n+m_d} \,|\, n \in \mathbb{N}\}$
- Examples

$L_{1,1,1,1}$:

$a\,b\,c\,d$

$a\,a\,b\,b\,c\,c\,d\,d$

$a\,a\,a\,b\,b\,b\,c\,c\,c\,d\,d\,d$

$\ldots$

$L_{1,2,1,2}$:

$a\,b\,b\,c\,d\,d$

$a\,a\,b\,b\,b\,c\,c\,d\,d\,d$

$a\,a\,a\,b\,b\,b\,b\,c\,c\,c\,d\,d\,d\,d$

$\ldots$

$L_{2,1,3,1}$:

$a\,a\,b\,c\,c\,c\,d$

$a\,a\,a\,b\,b\,c\,c\,c\,c\,d\,d$

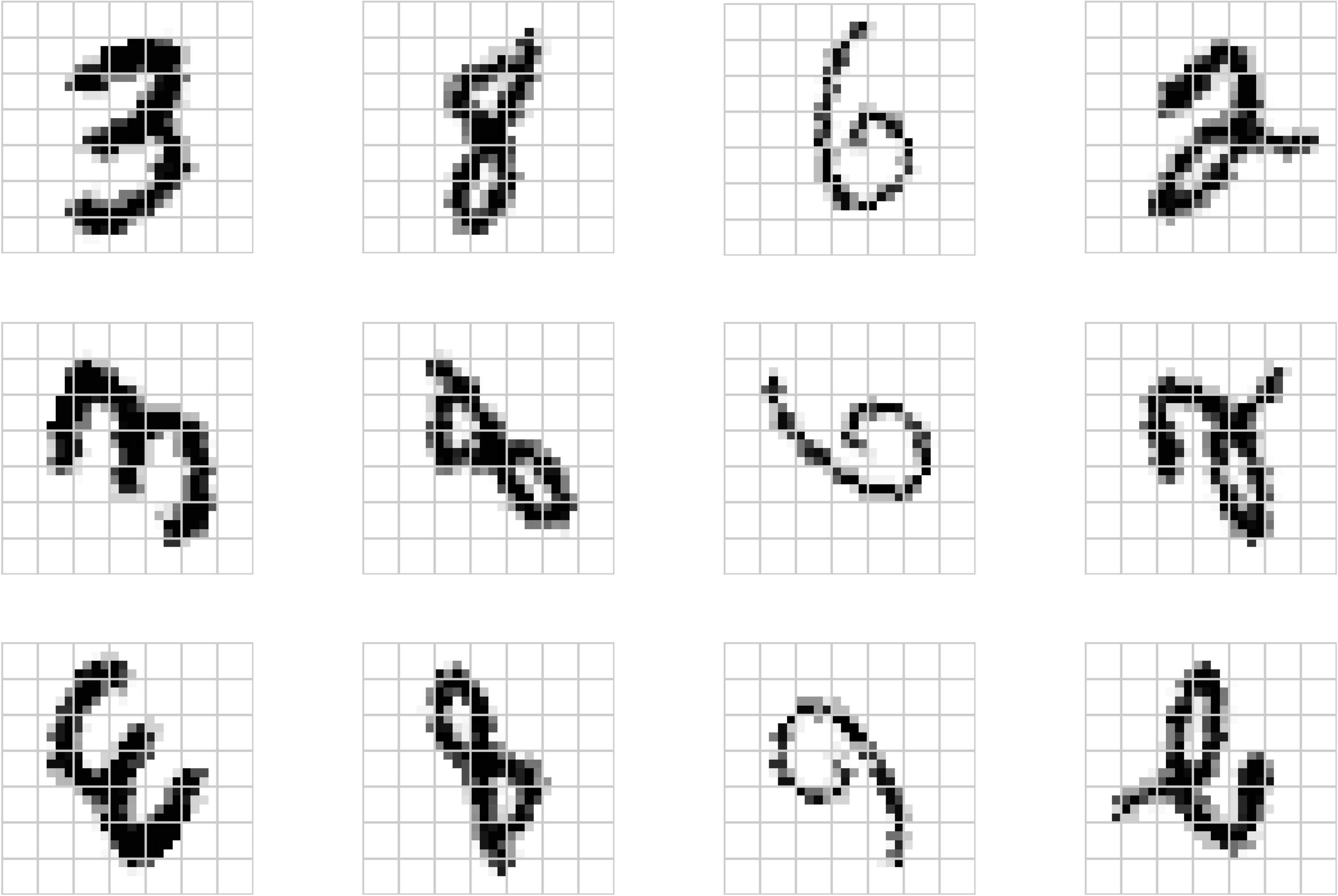
$a\,a\,a\,a\,b\,b\,b\,c\,c\,c\,c\,c\,d\,d\,d$

$\ldots$

$\ldots$

Tiled Sequential MNIST Dataset

- Classifiers of tiled sequentialised MNIST digits (prediction at every step)
- Every model is trained on a different rotation angle of the digits
- Examples



17° : …

74° : …

231° : …

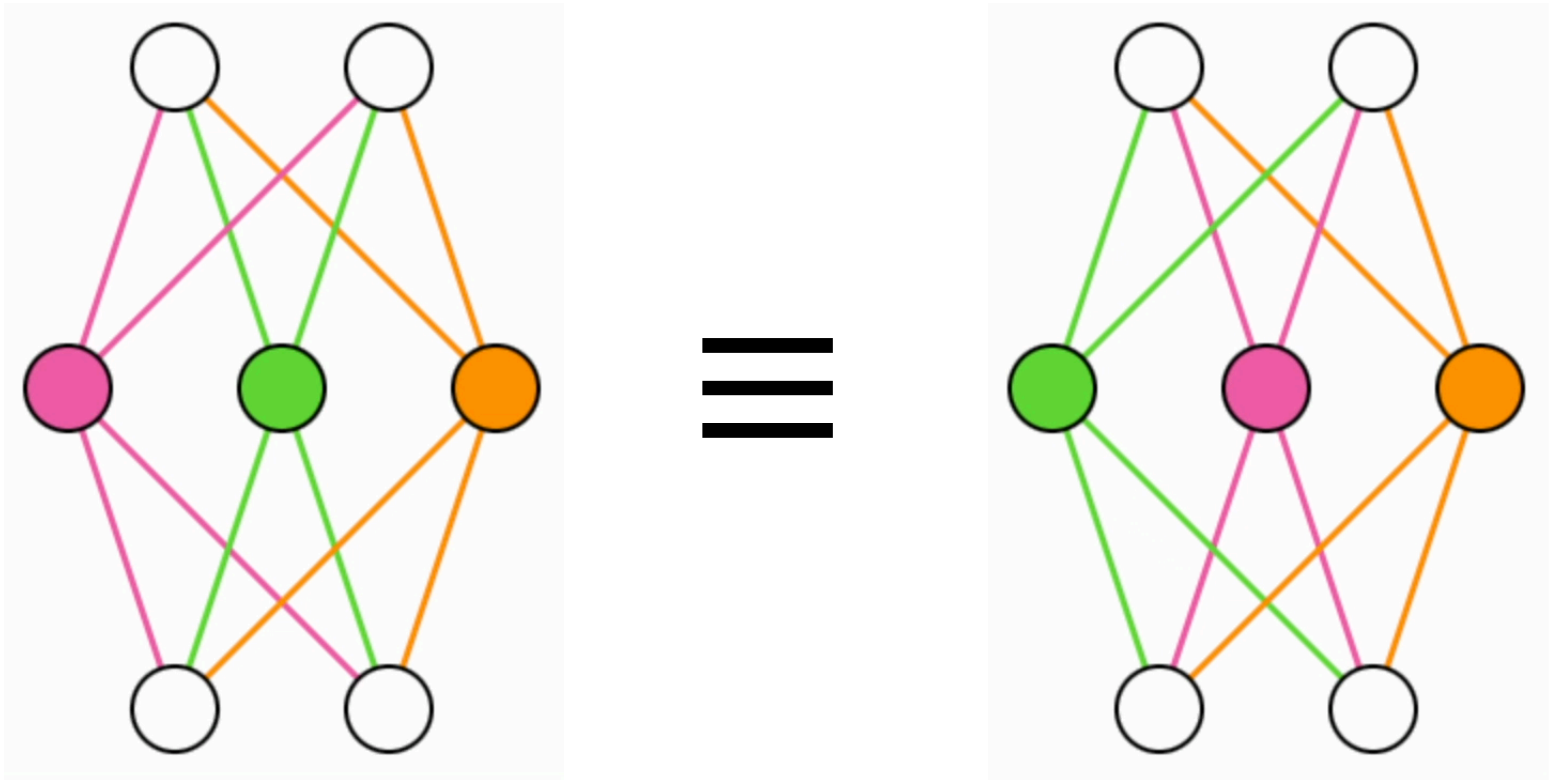
…

Weight Space Symmetries

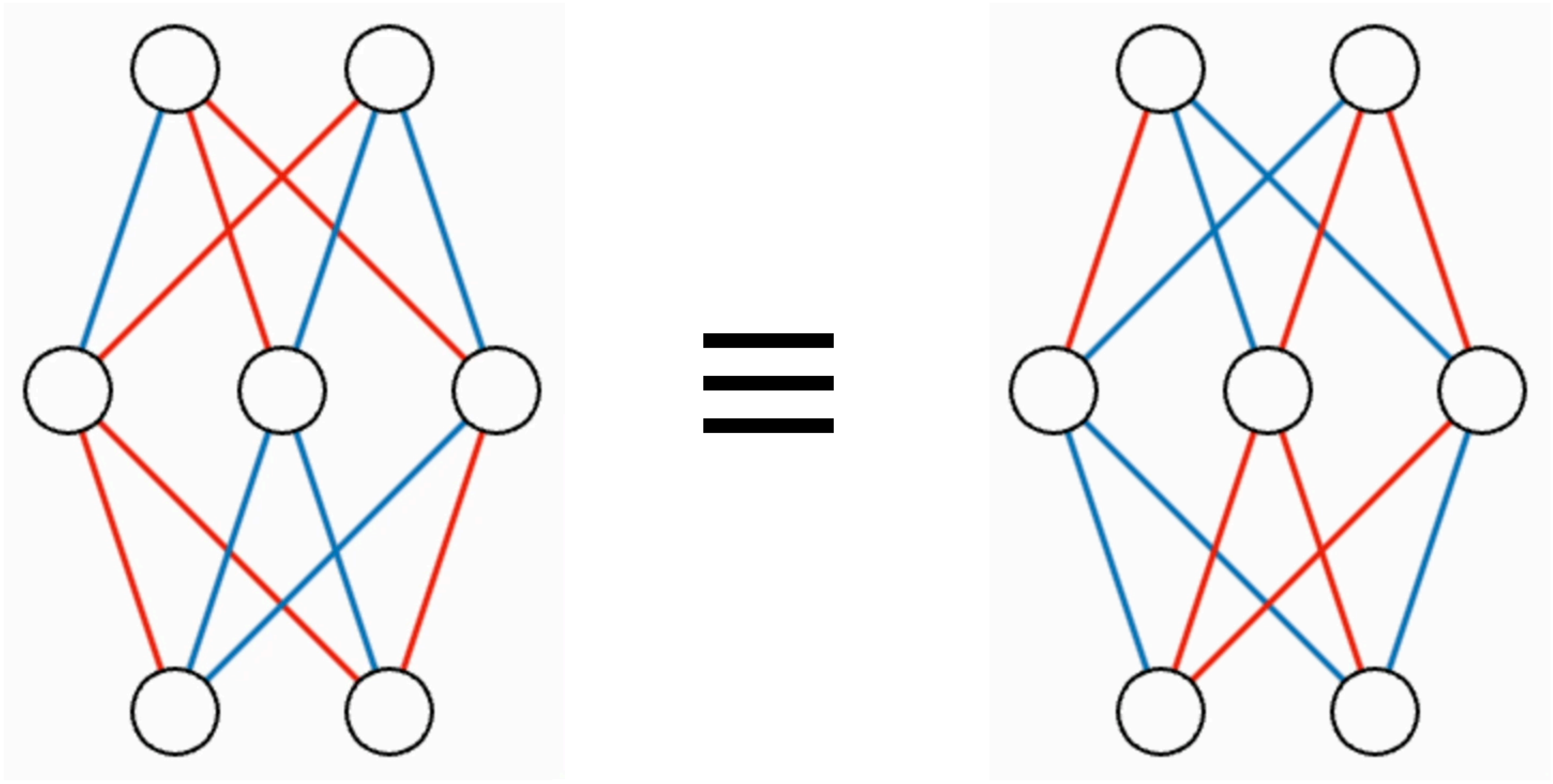
Challenge and Opportunity

Huge number of equivalent networks

- Hidden Neuron Permutation

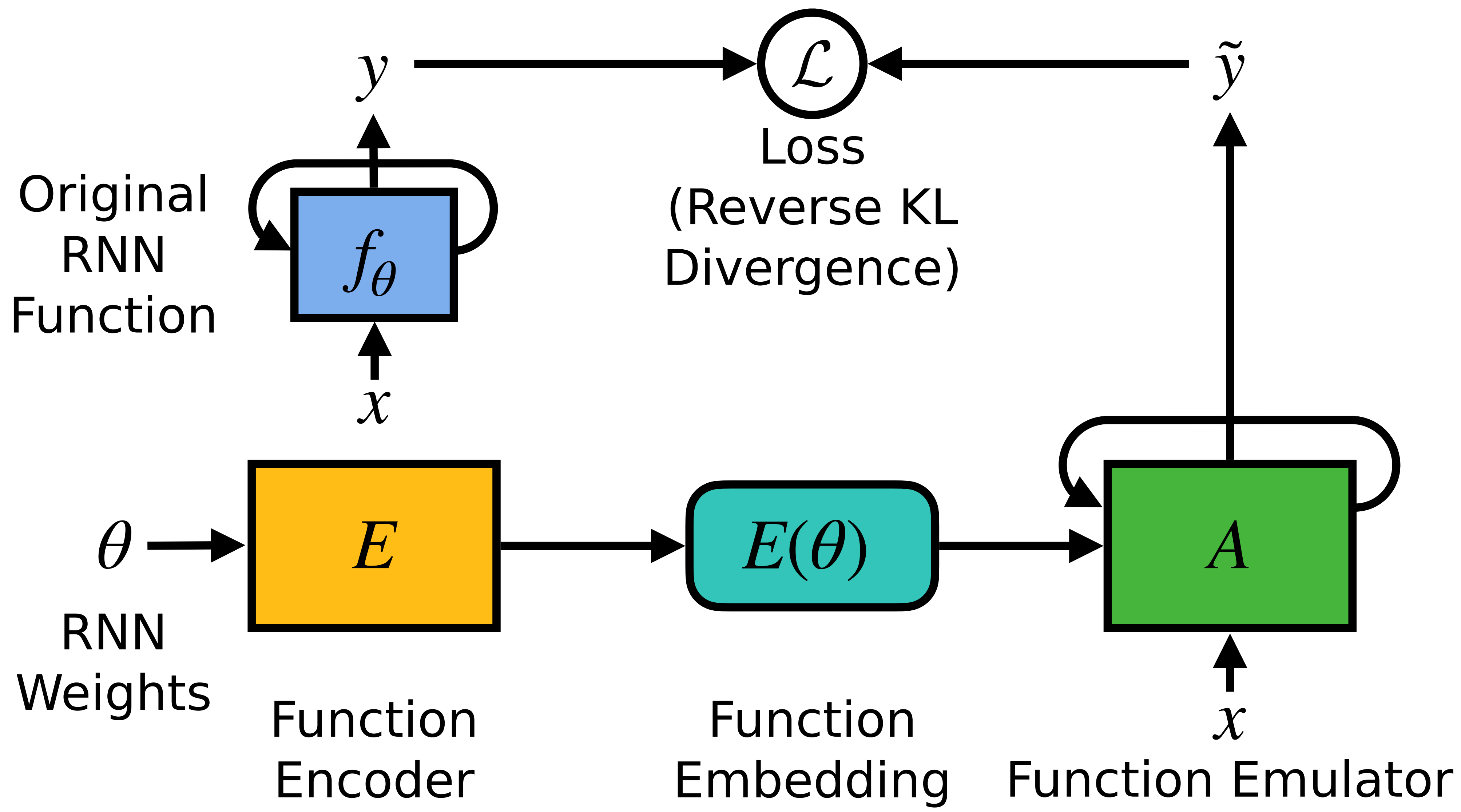


- Sign Flip / Scaling
  (depends on non-linearity)



Self-Supervised Training Method

- Reconstruction or naive contrastive methods? 🚨 *Weight Space Symmetries* 🚨
- *Instead*: Emulate the functionality of the network!

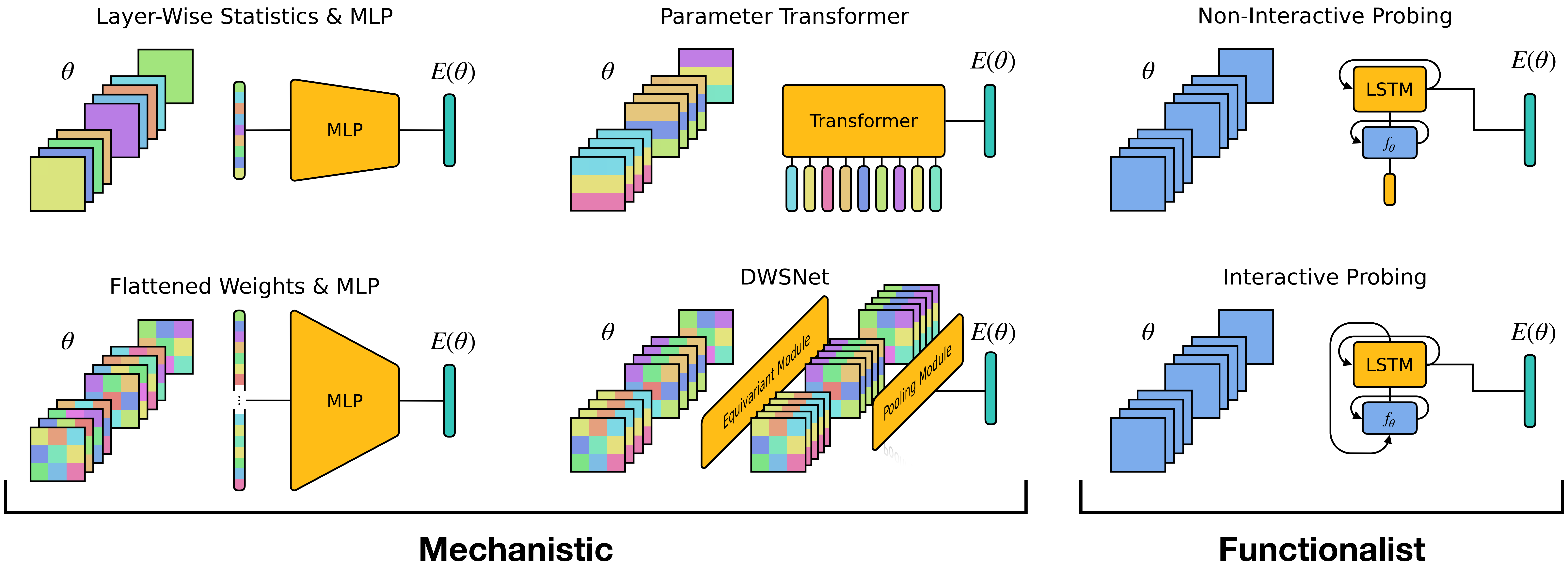


- Encoder E generates representation $E(\theta)$ of RNN f_θ
- Emulator A is conditioned on $E(\theta)$ and imitates f_θ

compare:
Raileanu et al., 2020, "Fast Adaptation via Policy-Dynamics Value Functions"

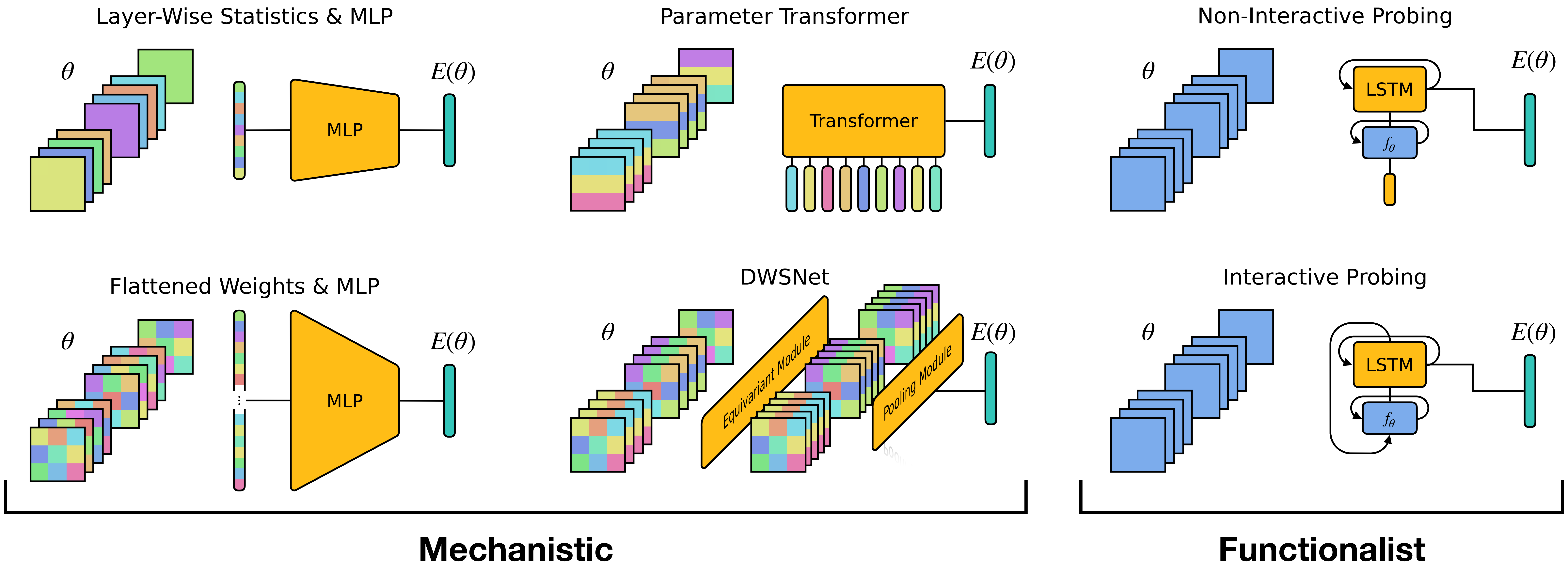
RNN Weight Encoder Architectures

- Distinction: Mechanistic Encoders & Functionalist Encoders
- **Mechanistic** encoders look at the weights θ directly
- **Functionalist** encoders look at the input-output mapping of f_θ



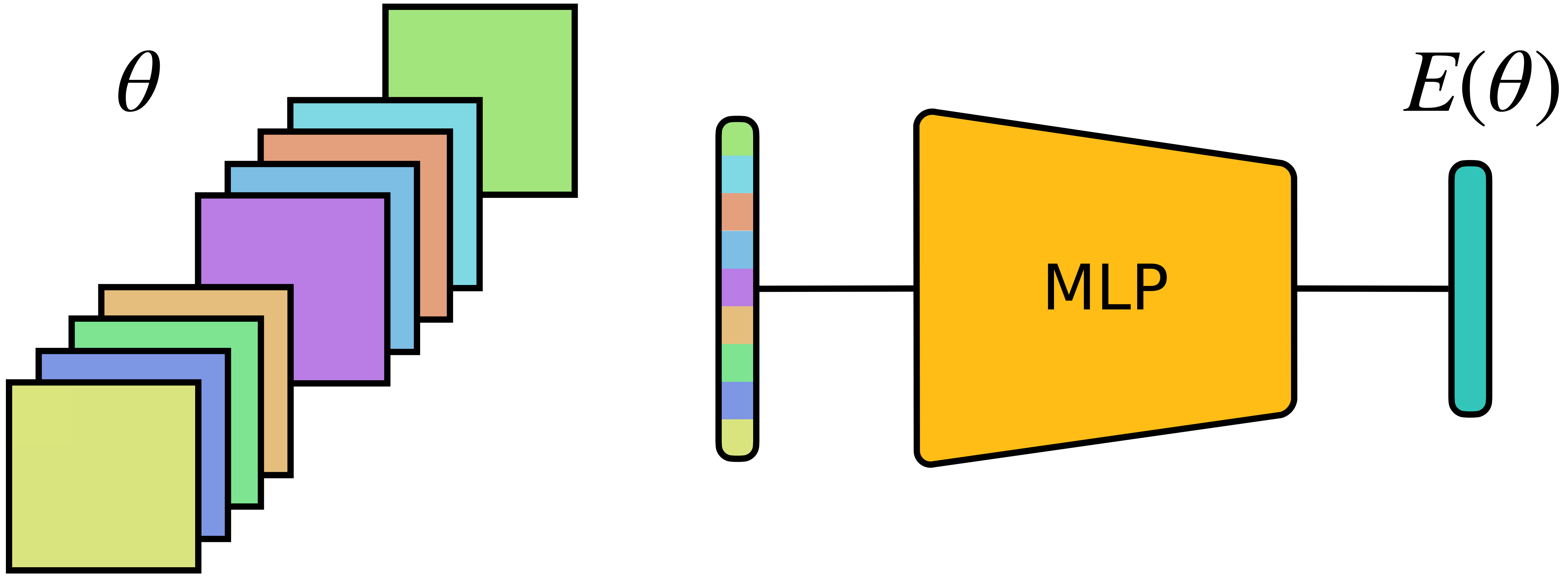
Layer-Wise Statistics & MLP (Mechanistic)

- Concatenate various global statistics of each weight matrix and feed into MLP
- Invariant to hidden neuron permutation 👍
- Not universal 👎



Layer-Wise Statistics & MLP (Mechanistic)

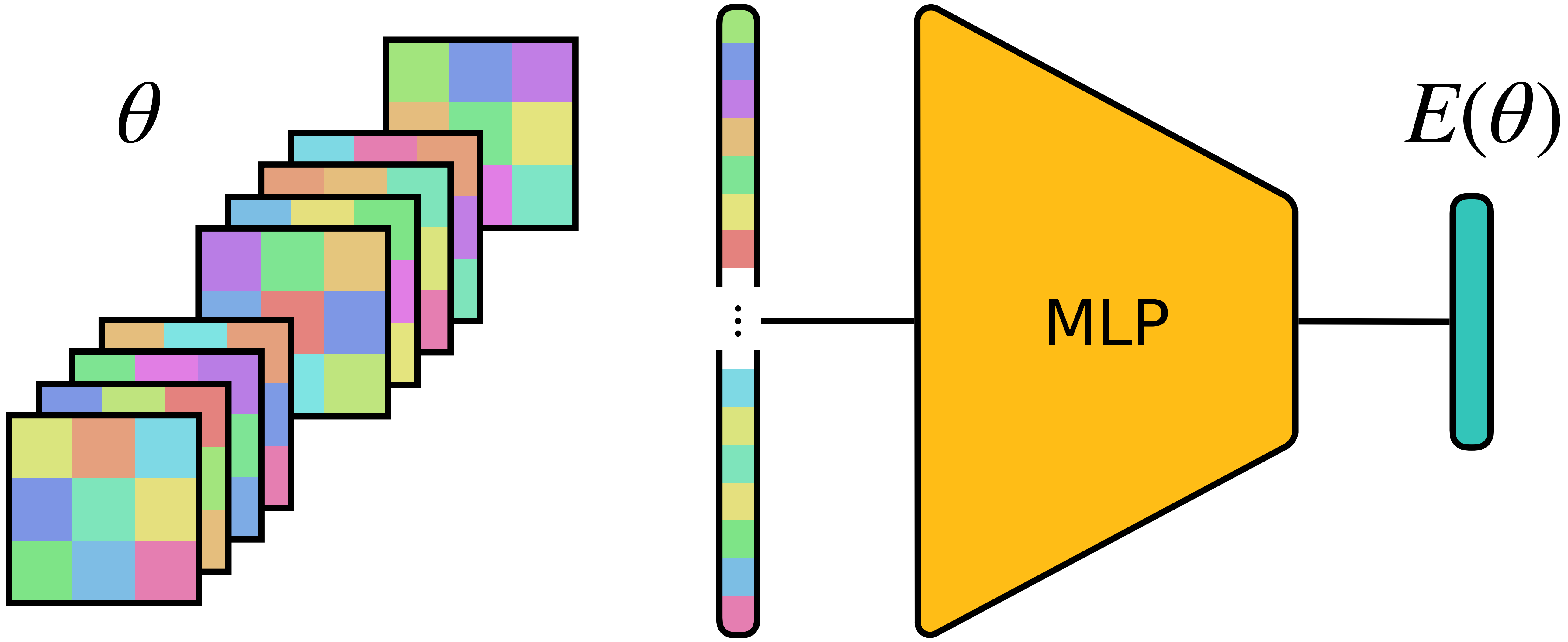
- Concatenate various global statistics of each weight matrix and feed into MLP
- Invariant to hidden neuron permutation 👍
- Not universal 👎



compare:
Unterthiner et al., 2020, “Predicting Neural Network Accuracy from Weights”

Flattened Weights & MLP (Mechanistic)

- Flatten all weights into one big vector and feed into MLP
- Not invariant to hidden neuron permutation 👎
- Universal 👍

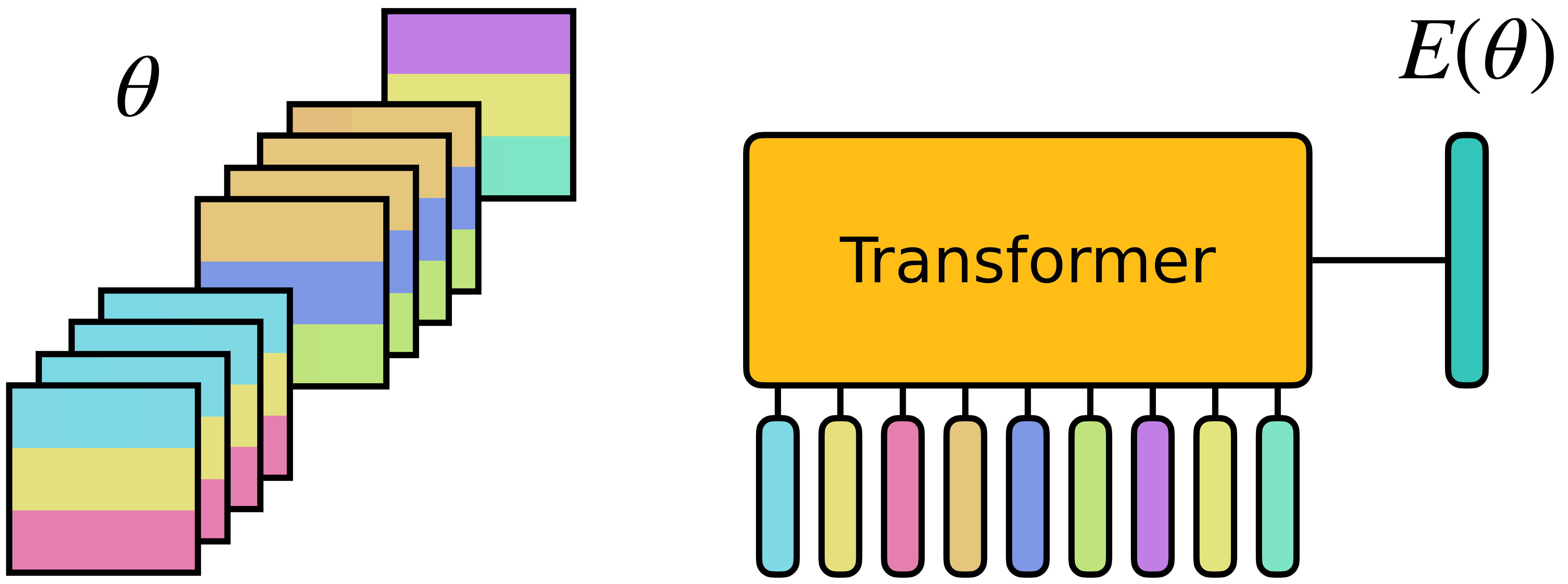


compare:
Faccio, Kirsch, Schmidhuber, 2020, "Parameter-Based Value Functions"

Herrmann, Kirsch, Schmidhuber, 2022, "Learning One Abstract Bit at a Time through Self-Invented Experiments Encoded as Neural Networks"

Parameter Transformer (Mechanistic)

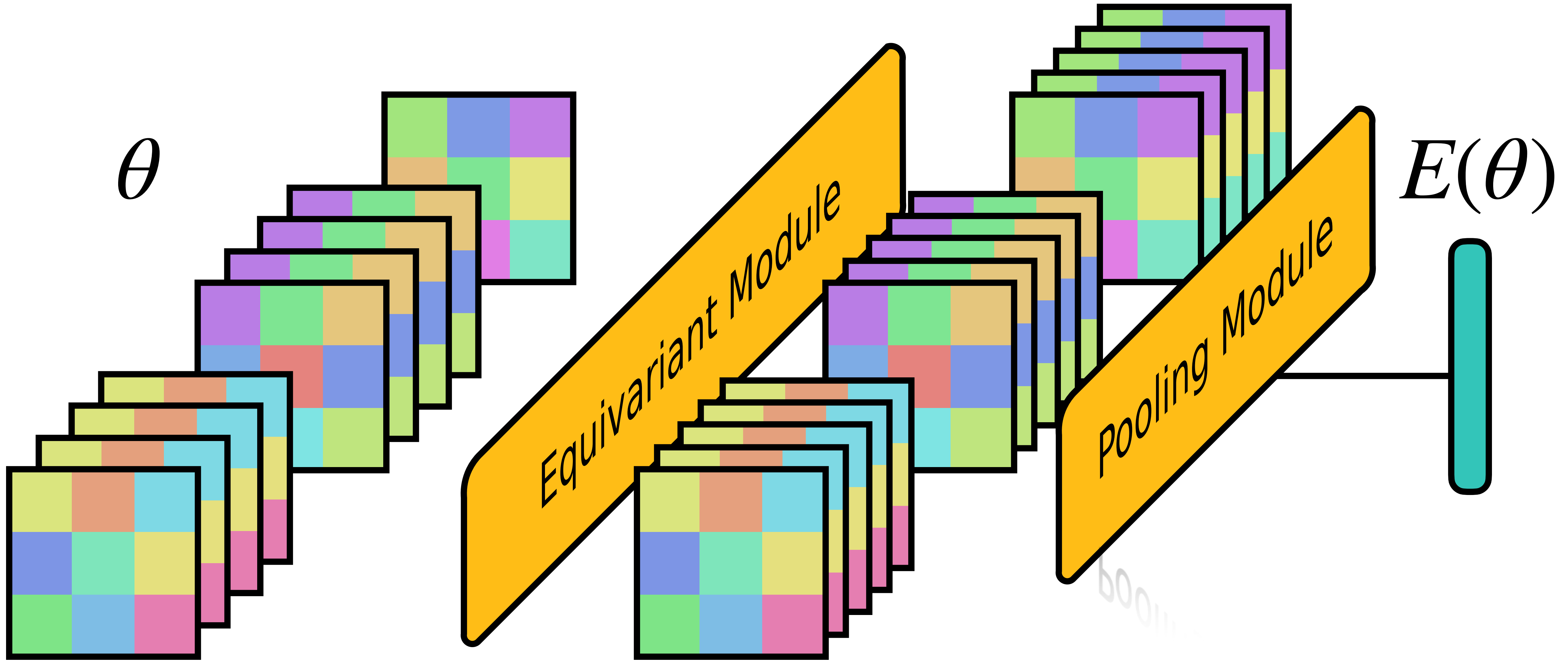
- Feed neuron weight vectors as sequence to a transformer model
- Not invariant to hidden neuron permutation 👎
- Universal 👍



compare:
Schürholt, Kostadinov, Borth, 2021, "Self-Supervised Representation Learning on Neural Network Weights for Model Characteristic Prediction"

Deep Weight Space Net (Mechanistic)

- Equivariant weight processing modules followed by invariant pooling layer
- Invariant precisely to hidden neuron permutation 👍
- Universal 👍



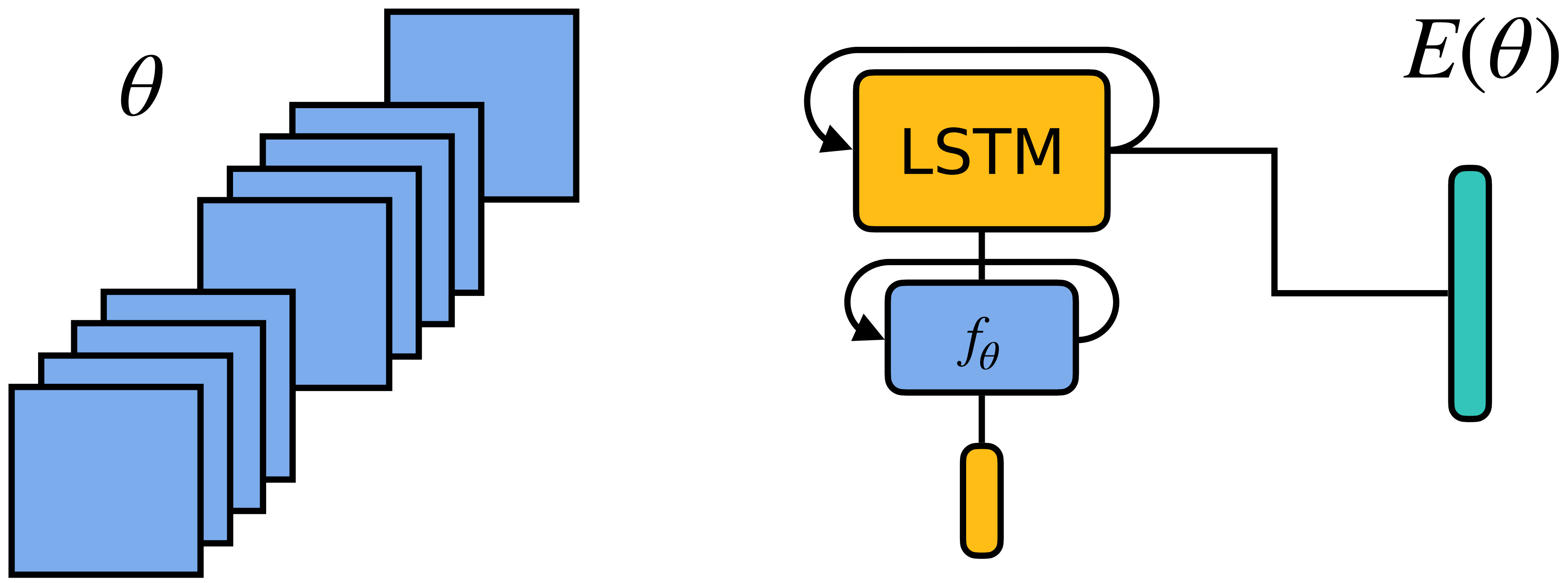
compare:
Navon et al., 2023, "Equivariant Architectures for Learning in Deep Weight Spaces"

Zhou et al., 2023, "Permutation Equivariant Neural Functionals"

Kirsch, Schmidhuber, 2020, "Meta Learning Backpropagation and Improving it"

Non-Interactive Probing (Functionalist)

- Fixed but learnable probing sequences are given to f_θ
- Based on the corresponding probing outputs, $E(\theta)$ is computed
- Invariant to everything that leaves f_θ's functionality intact 👍
- Not fully universal 🫱

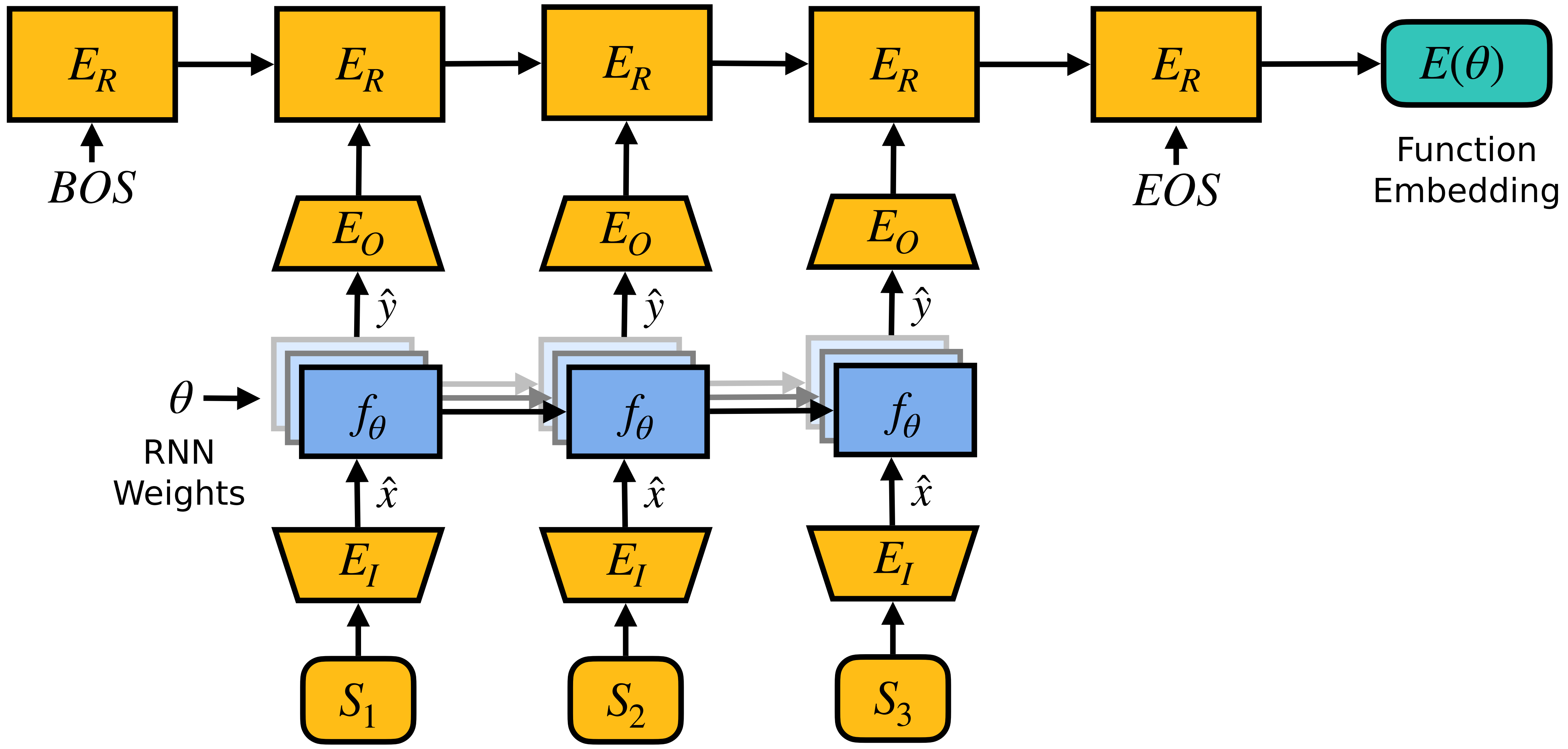


compare:
Harb et al., 2020, "Policy Evaluation Networks"

Faccio et al., 2022, "Goal-Conditioned Generator of Deep Policies"

Non-Interactive Probing (Functionalist)

- Fixed but learnable probing sequences are given to f_θ
- Based on the corresponding probing outputs, $E(\theta)$ is computed
- Invariant to everything that leaves f_θ's functionality intact 👍
- Not fully universal 🫱

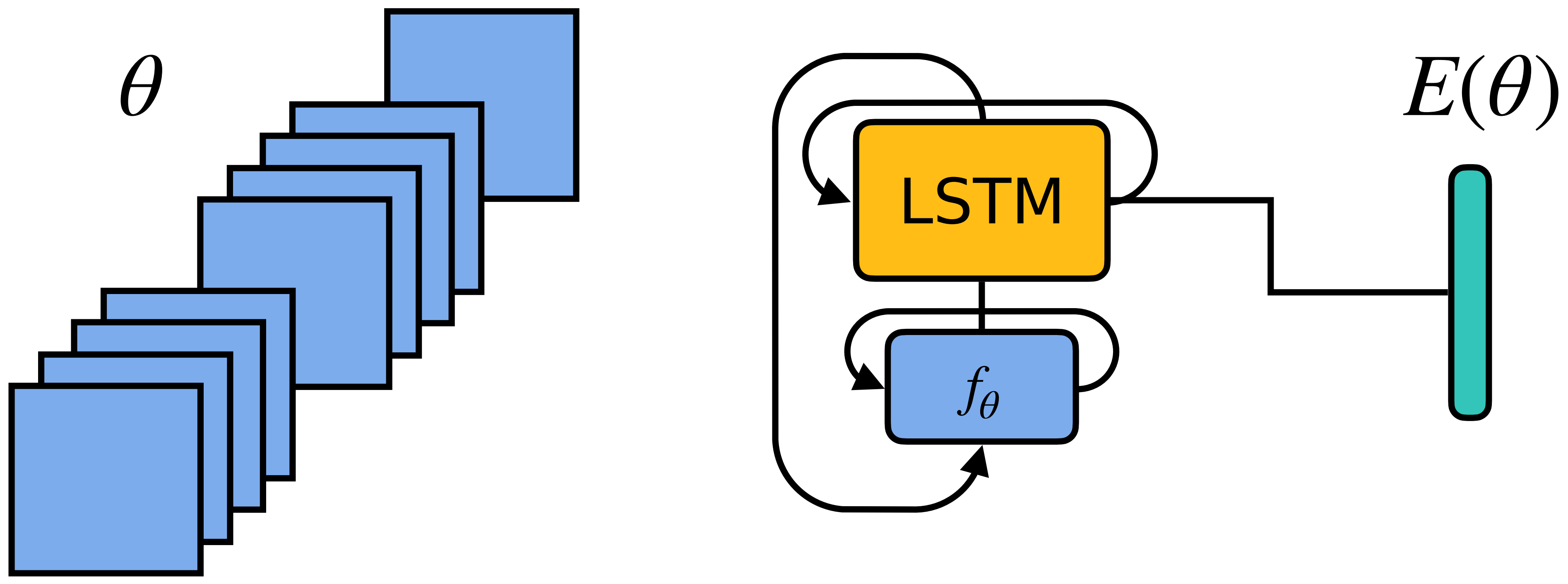


compare:
Harb et al., 2020, "Policy Evaluation Networks"

Faccio et al., 2022, "Goal-Conditioned Generator of Deep Policies"

Interactive Probing (Functionalist)

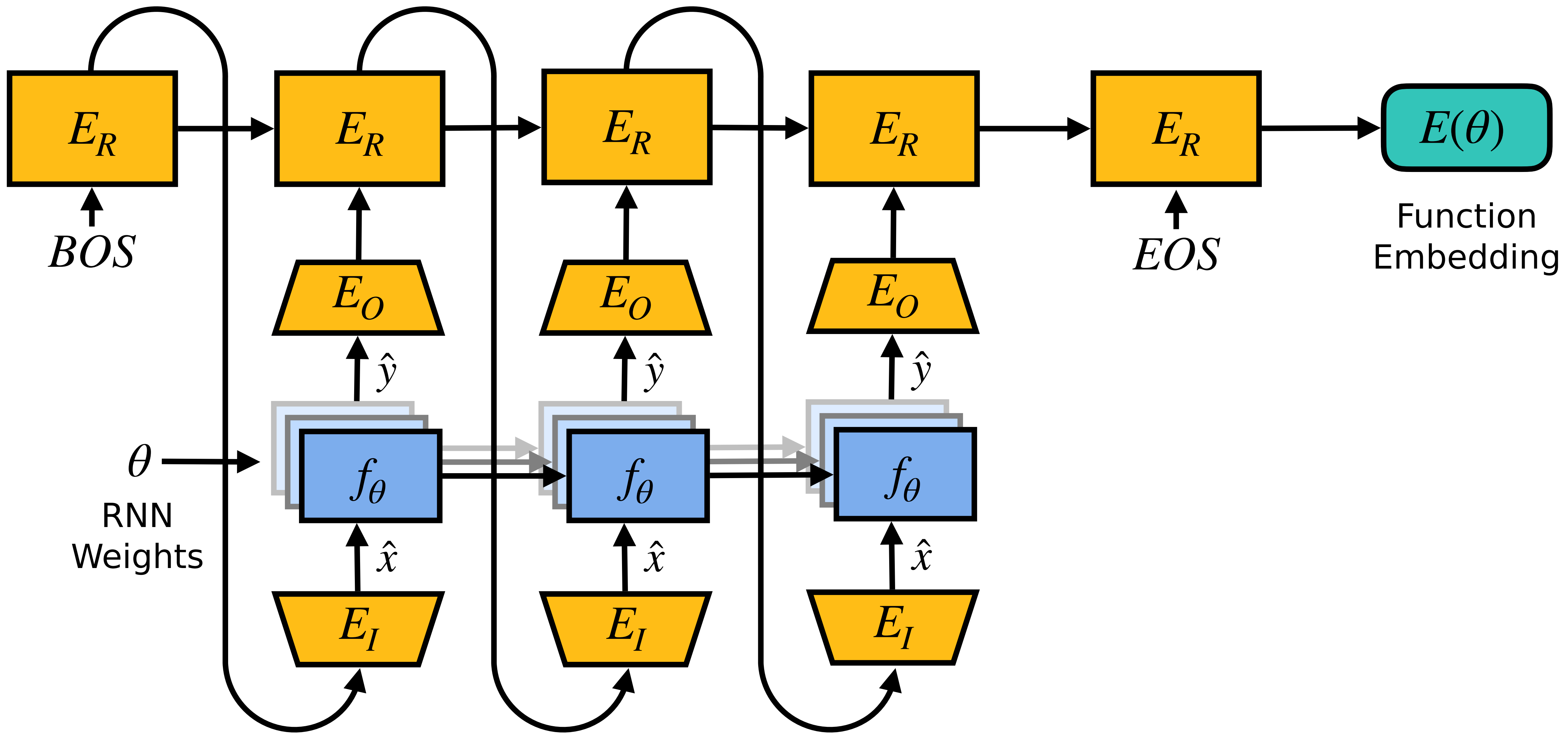
- Probing sequences are dynamically generated by core LSTM
- Next probing inputs depend on f_θ's response to all previous probing inputs
- Invariant to everything that leaves f_θ's functionality intact 👍
- Not fully universal 🫳



compare:
Schmidhuber, 2015, "On Learning to Think"

Interactive Probing (Functionalist)

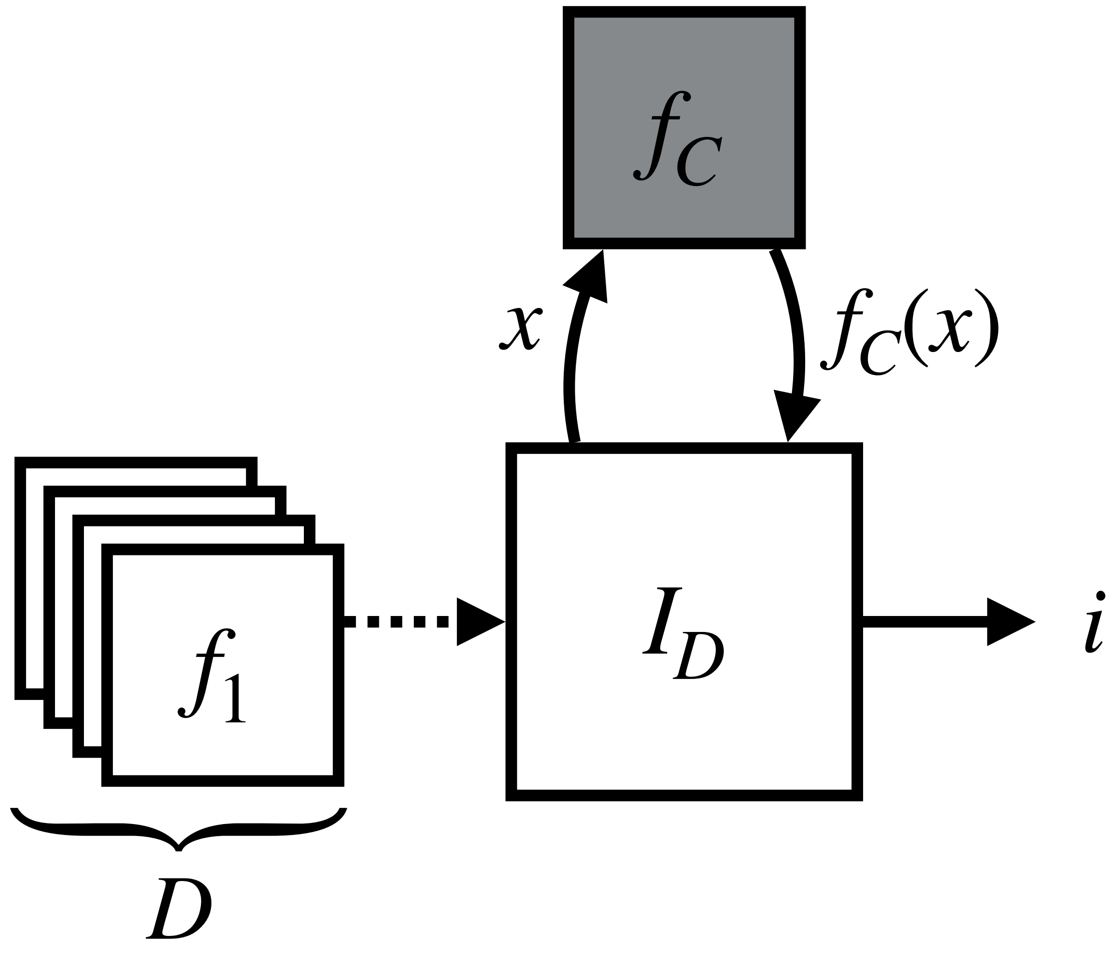
- Probing sequences are dynamically generated by core LSTM
- Next probing inputs depend on f_θ's response to all previous probing inputs
- Invariant to everything that leaves f_θ's functionality intact 👍
- Not fully universal 🫳



compare:
Schmidhuber, 2015, "On Learning to Think"

Functionalist Approach: Theoretical Results

What's the difference between interactive and non-interactive probing?



details in the paper

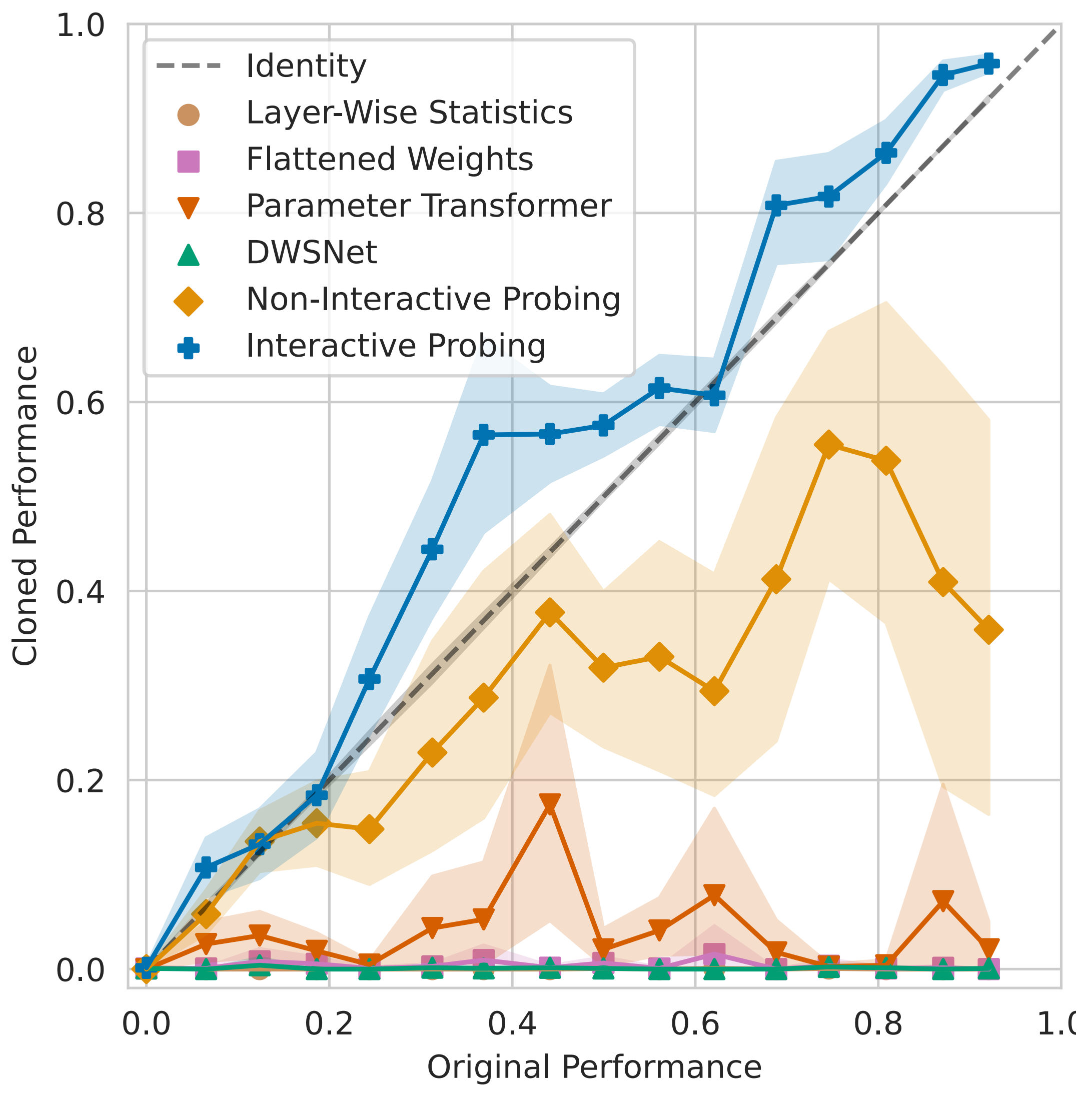
Results

- General upper bound for required interactions is the same
- In certain settings, interactive probing is exponentially more efficient

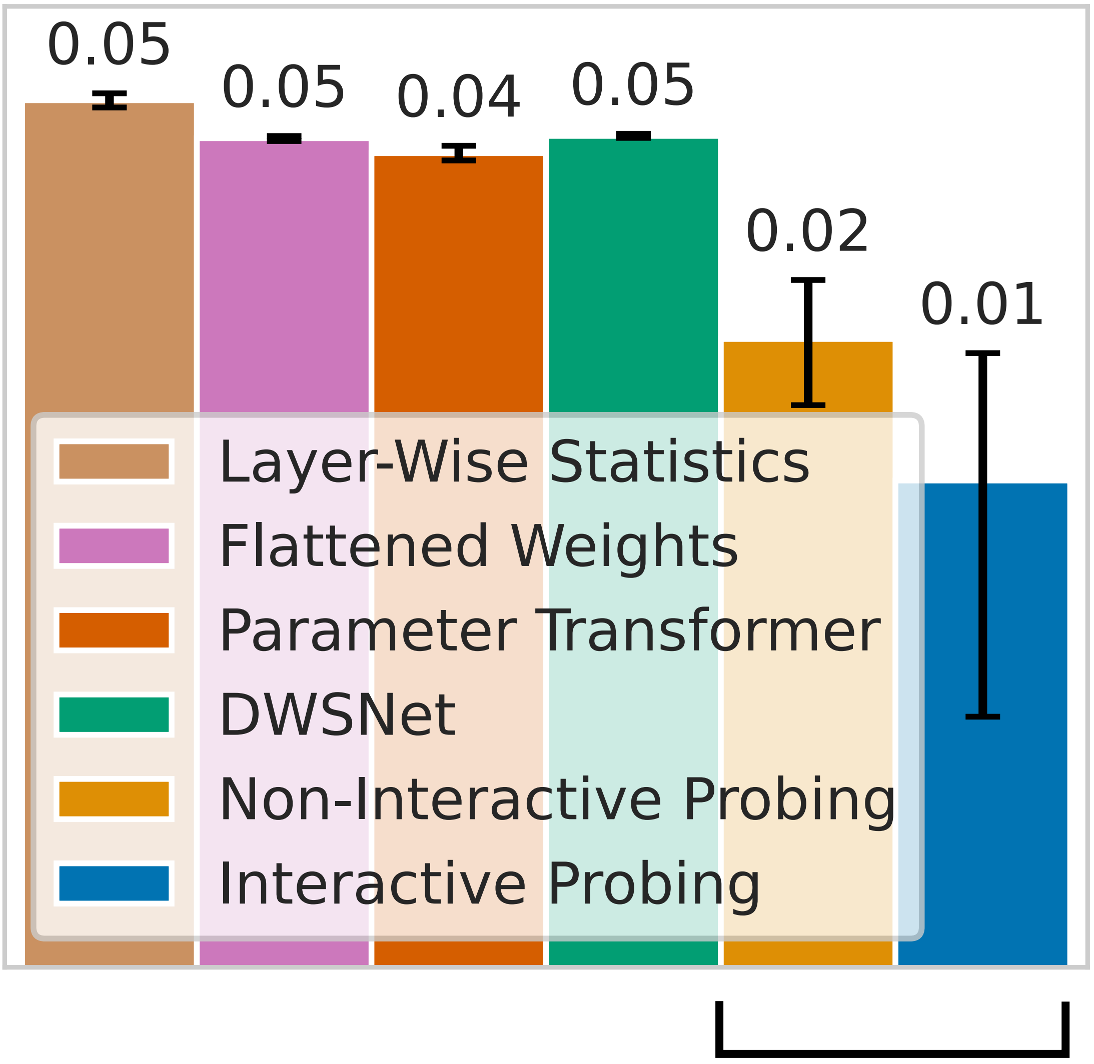
Pre-Training Results

Formal Languages

Original vs. Emulated



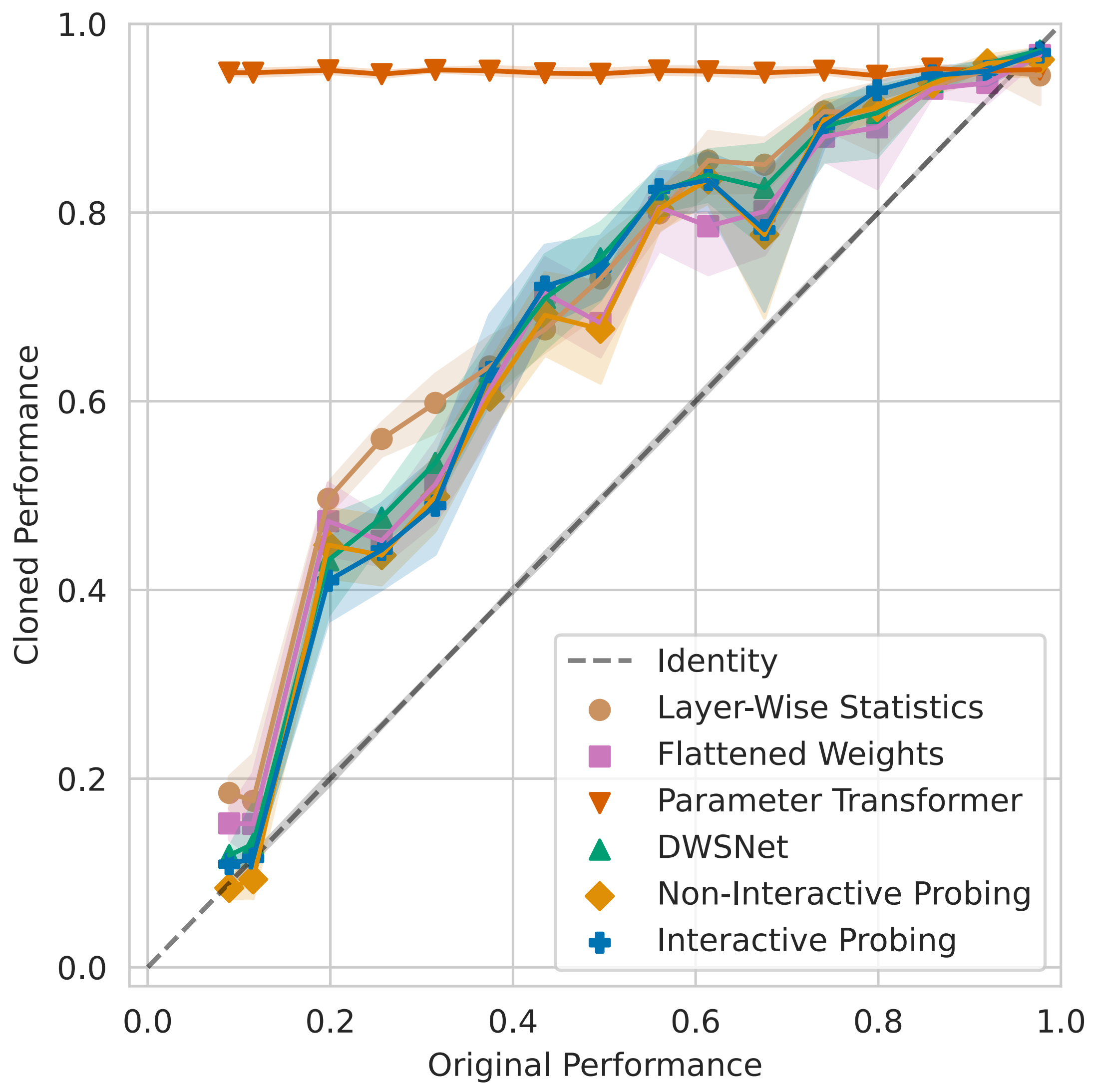
Validation Loss



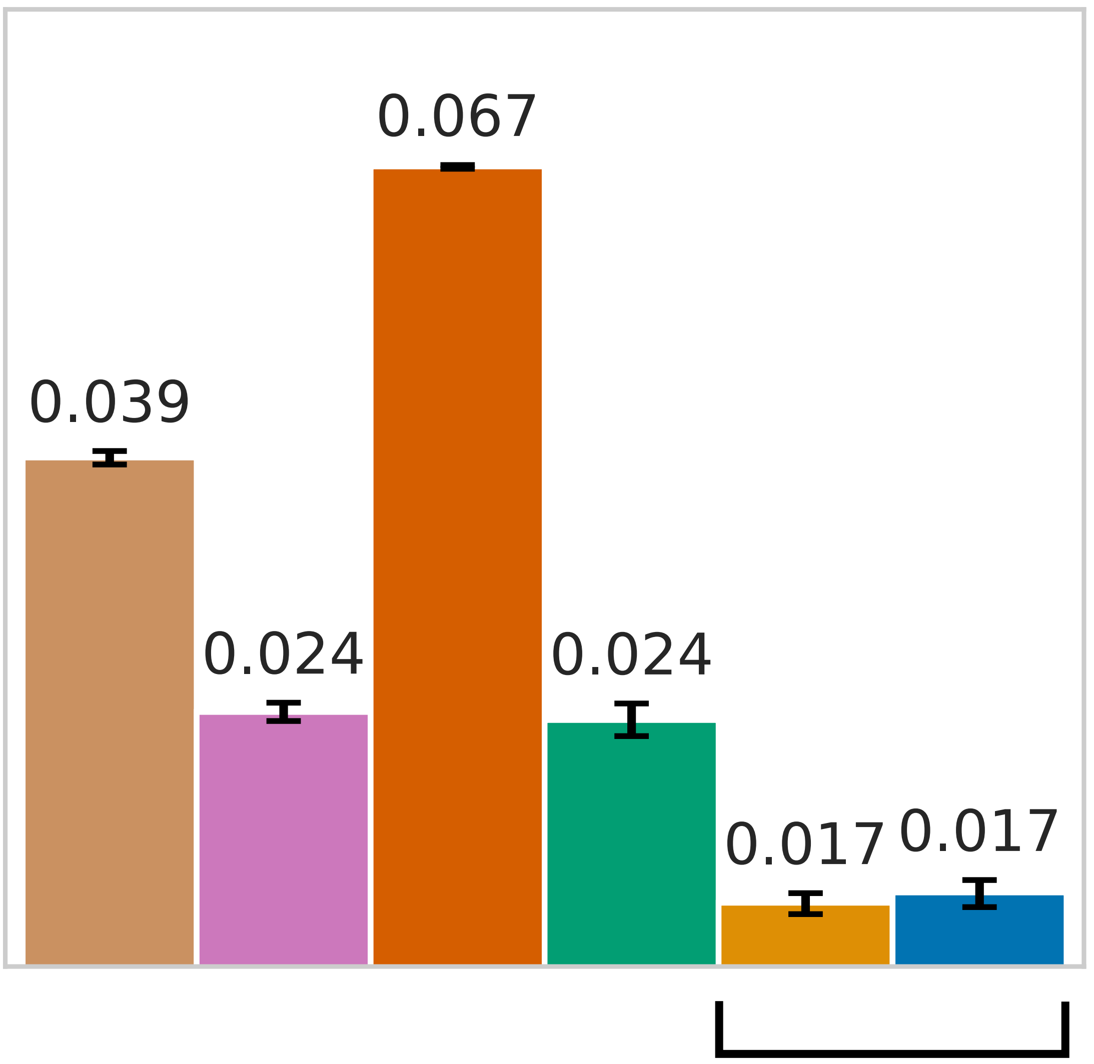
Functionalist

Sequential MNIST

Original vs. Emulated



Validation Loss

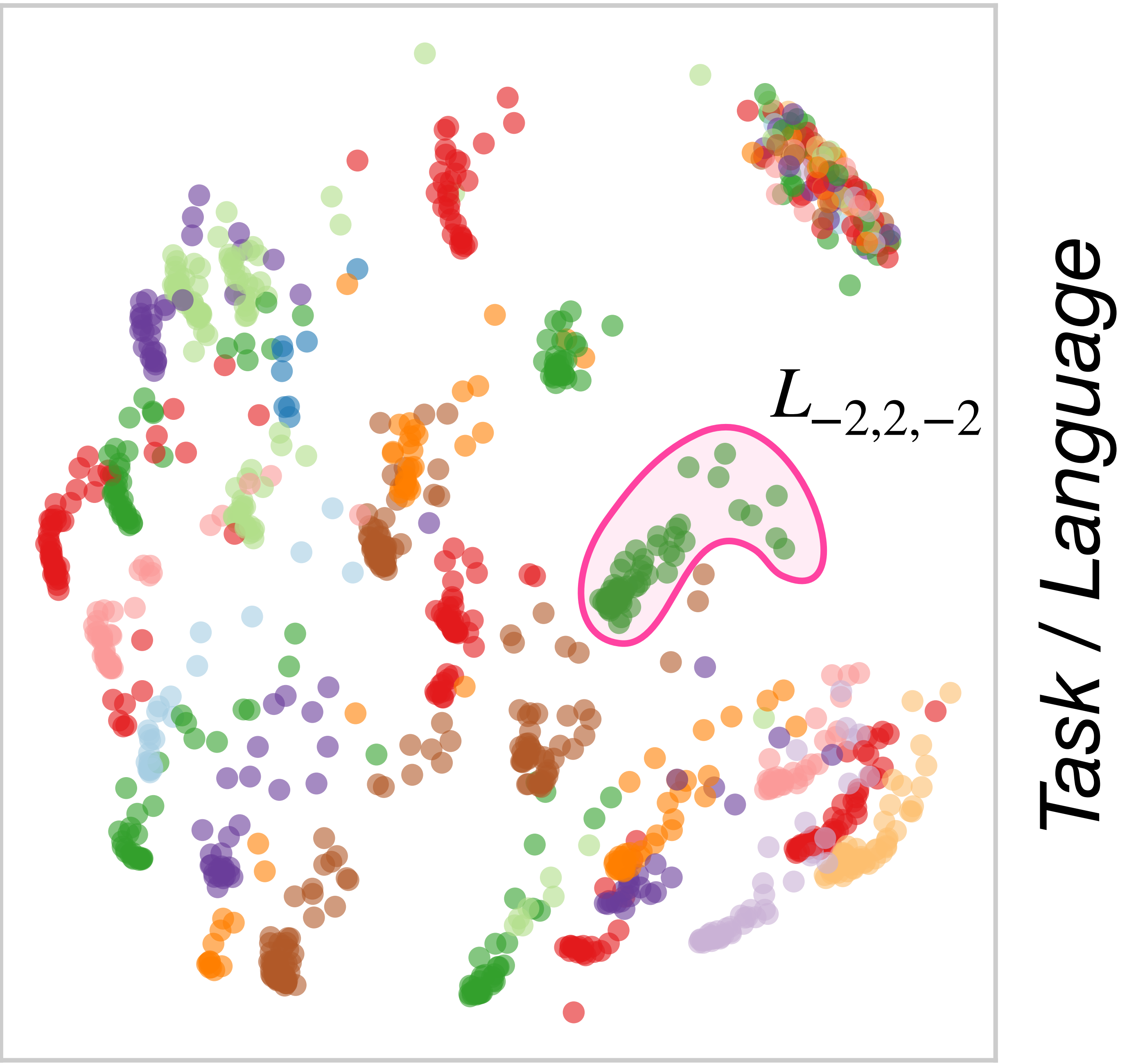


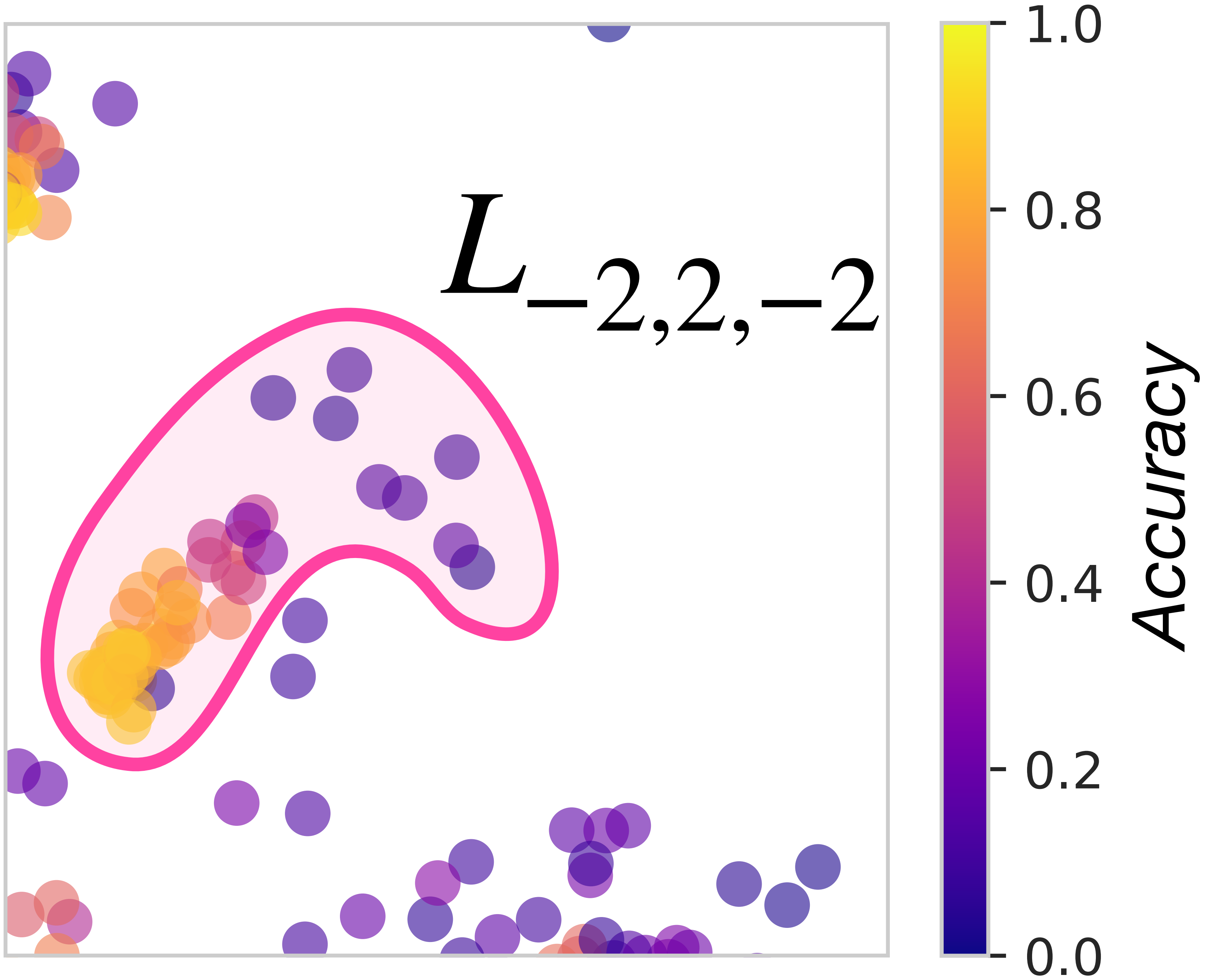
Functionalist

Learned Embedding Spaces

Formal Languages

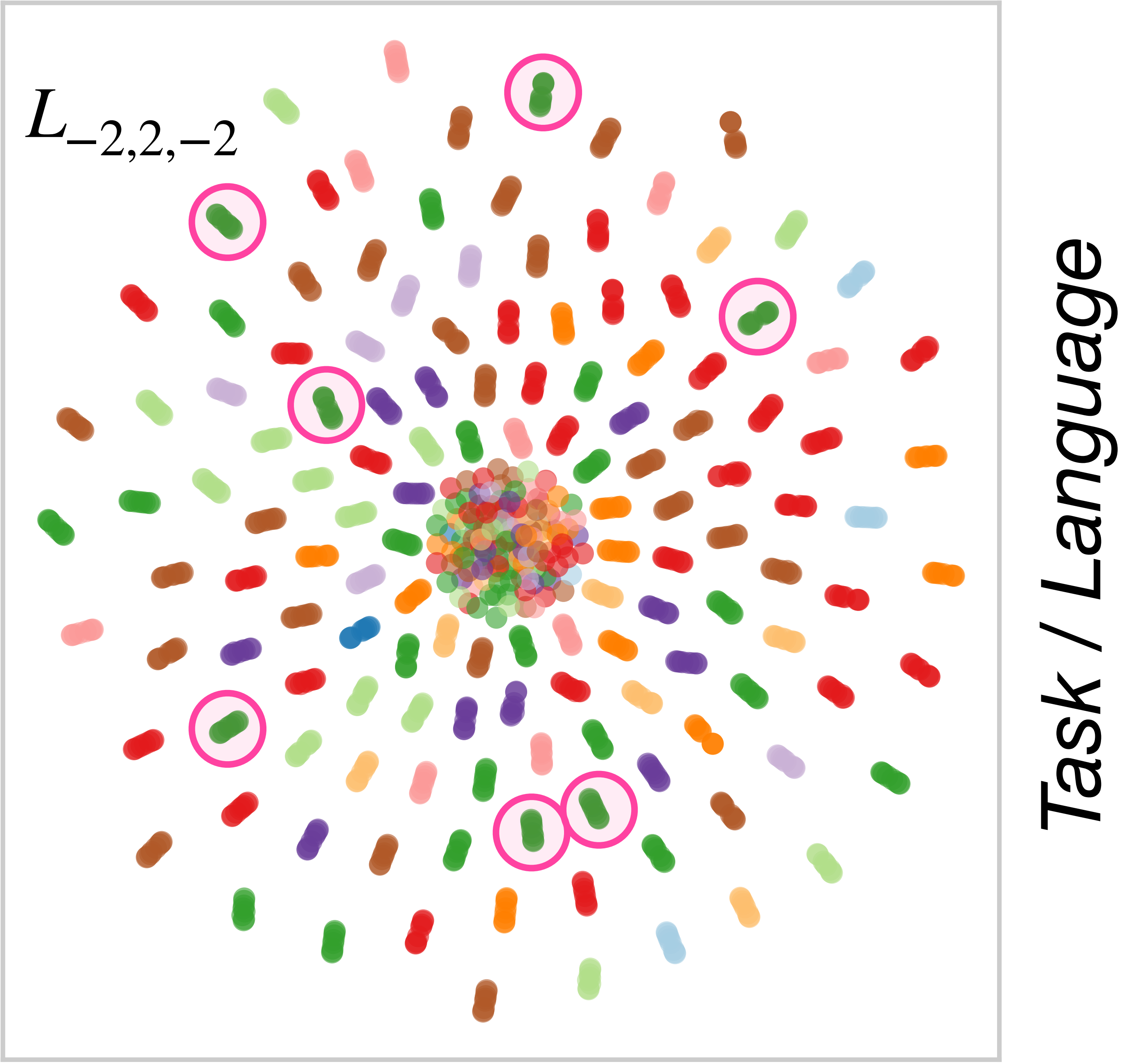
Interactive Fingerprinting Embeddings (PCA)





Comparison:

t-SNE of weights

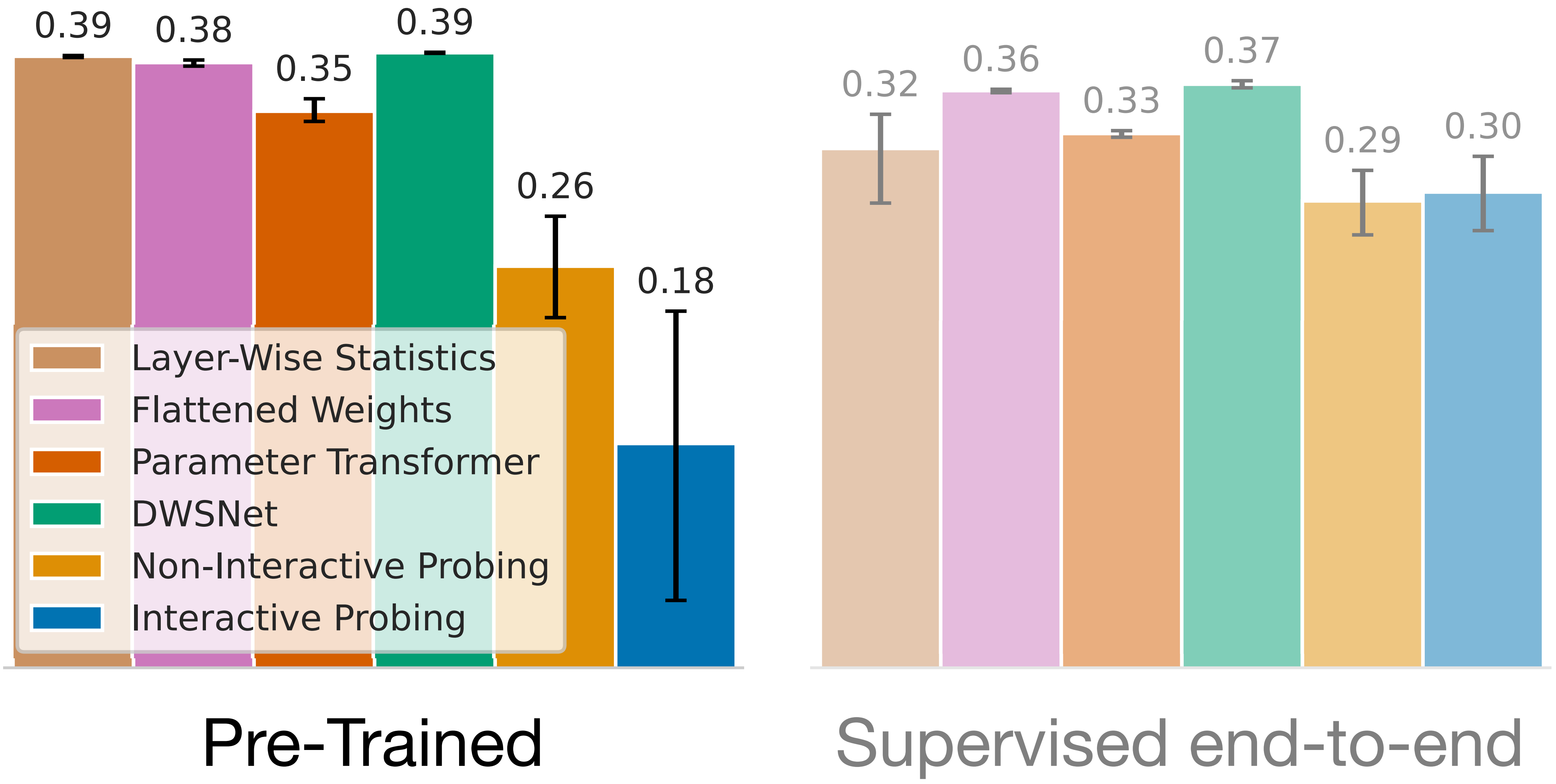


Downstream Results

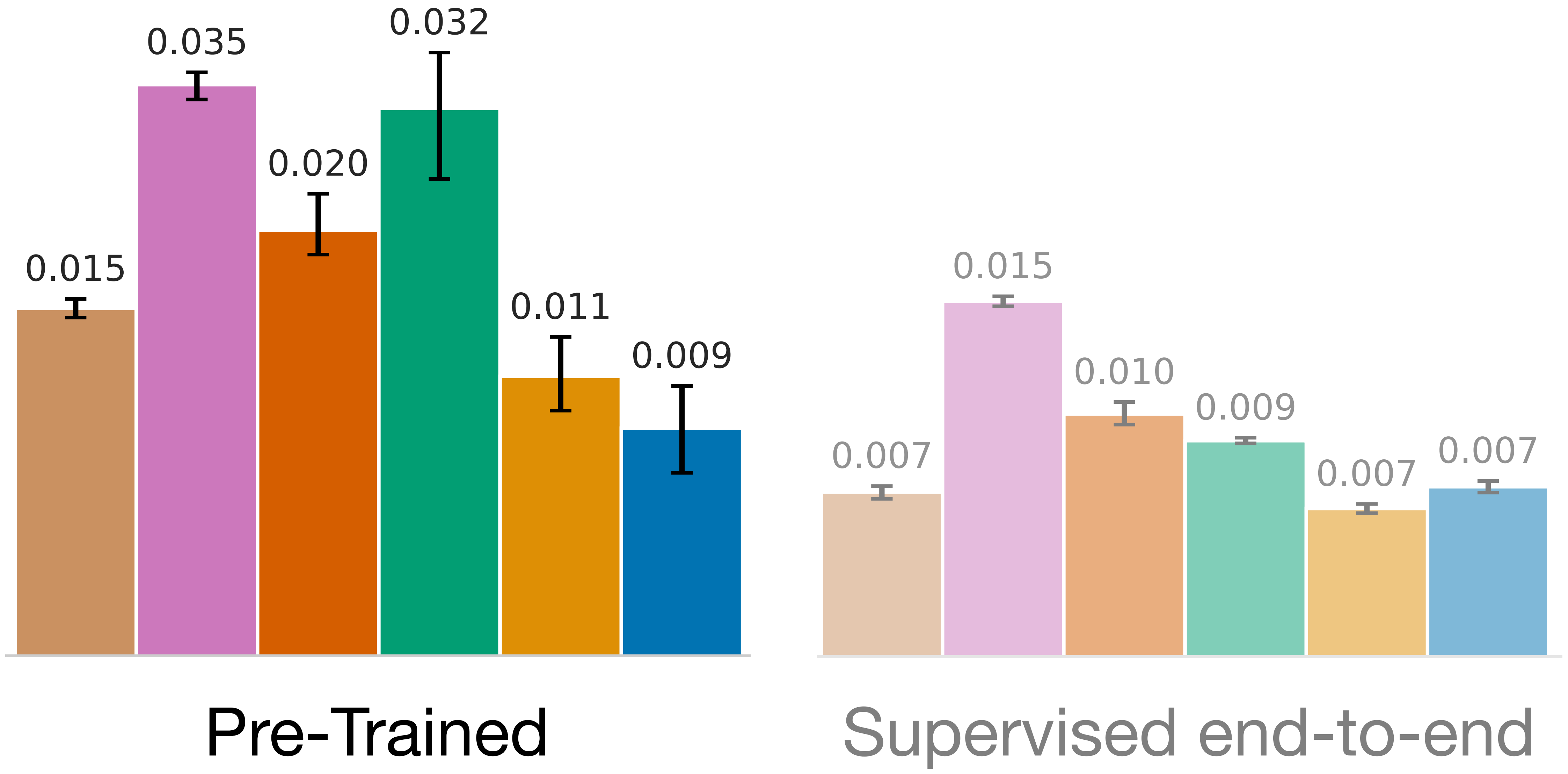
Predictor MLP on top of learned representations

Formal Languages

Task prediction loss



Accuracy prediction loss



more results in the paper

Conclusion

Learning RNN Weight Representations

- Two RNN ‘Model Zoo’ datasets
- Emulation-based pre-training method
- Distinction between **mechanistic** and **functionalist** weight encoders
- Two novel functionalist encoder types
- Comparison of six different RNN weight encoder architectures

Functionalist encoders are superior at complex tasks

Thank you!

Paper

Code & Datasets