Learning Useful Representations of Recurrent Neural Network Weight Matrices

IDSIA

Paper

Vincent Herrmann, Francesco Faccio, Jürgen Schmidhuber

Università della Svizzera italiana

Scuola universitaria professionale della Svizzera italiana SUPSI

KAUST

Code & Datasets

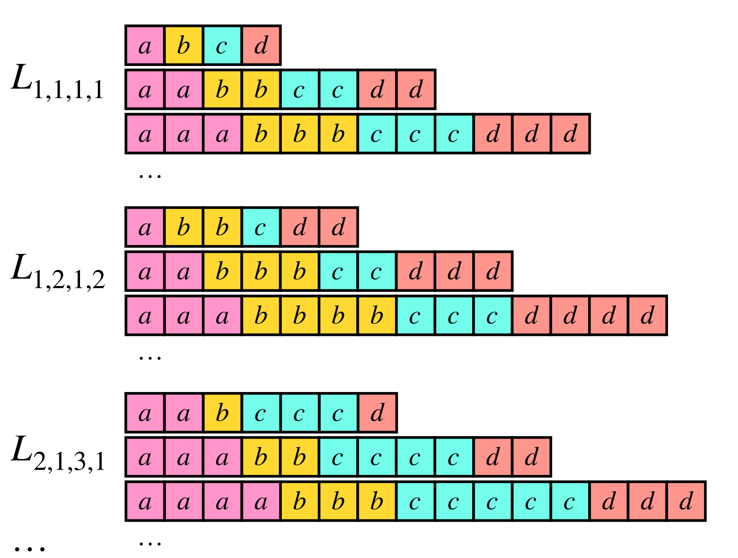
Datasets

- Two datasets of LSTM weights
- Each LSTM is trained to achieve a different task

Formal Languages Dataset

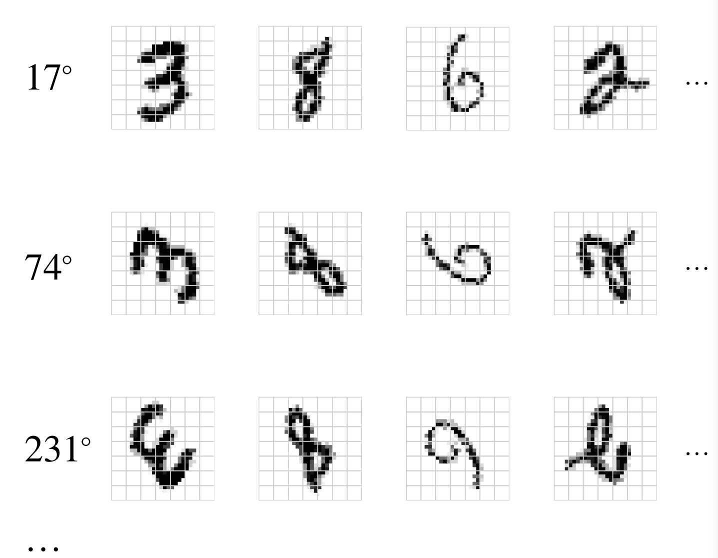
Autoregressive models of languages

$L_{m_a,m_b,m_c,\dots} := \{a^{n+m_a}b^{n+m_b}c^{n+m_c}\dots \mid n \in \mathbb{N}\}$



Tiled Sequential MNIST Dataset

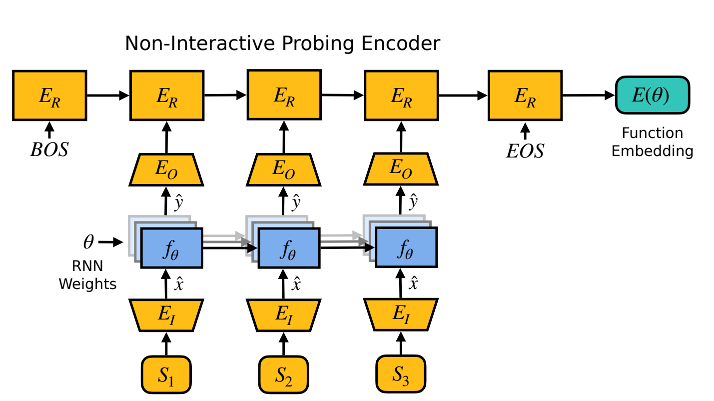
Classifiers of the MNIST dataset, rotated by different angles



Functionalist Approaches

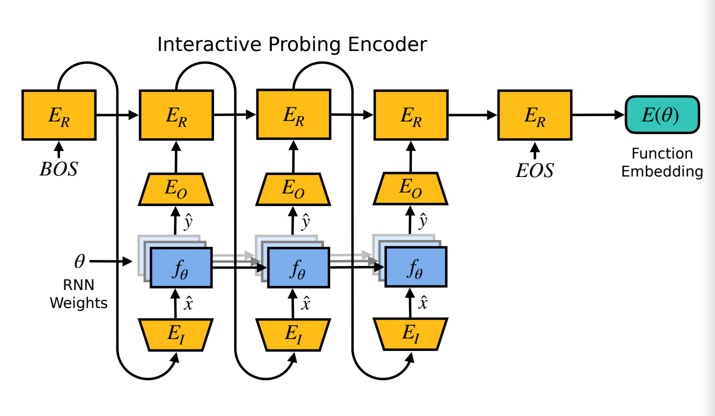
Non-Interactive Probing Encoder

- Fixed but learnable probing sequences are given as input to the input RNN f_θ
- Based on the corresponding output sequences, the core LSTM E_R computes the representation $E(\theta)$



Interactive Probing Encoder

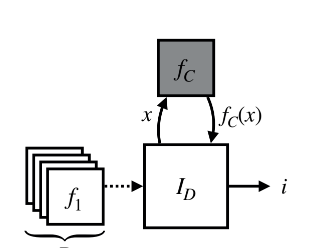
- Probing sequences are dynamically generated by the core LSTM E_R
- The next probing input depends on all the previous probing inputs and corresponding outputs



Theory for the Functionalist Approach

Setting:

- Interrogator I_D has to Identify a specific function f_C from a known set D of total computable functions
- It has to use as few interactions as possible



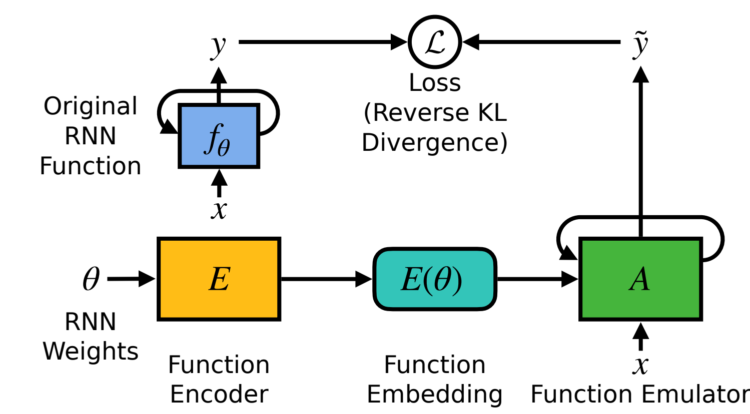
Results:

- The general upper bound of required interactions is the same for interactive and non-interactive Interrogators
- For certain function sets, an interactive Interrogator needs exponentially fewer interactions

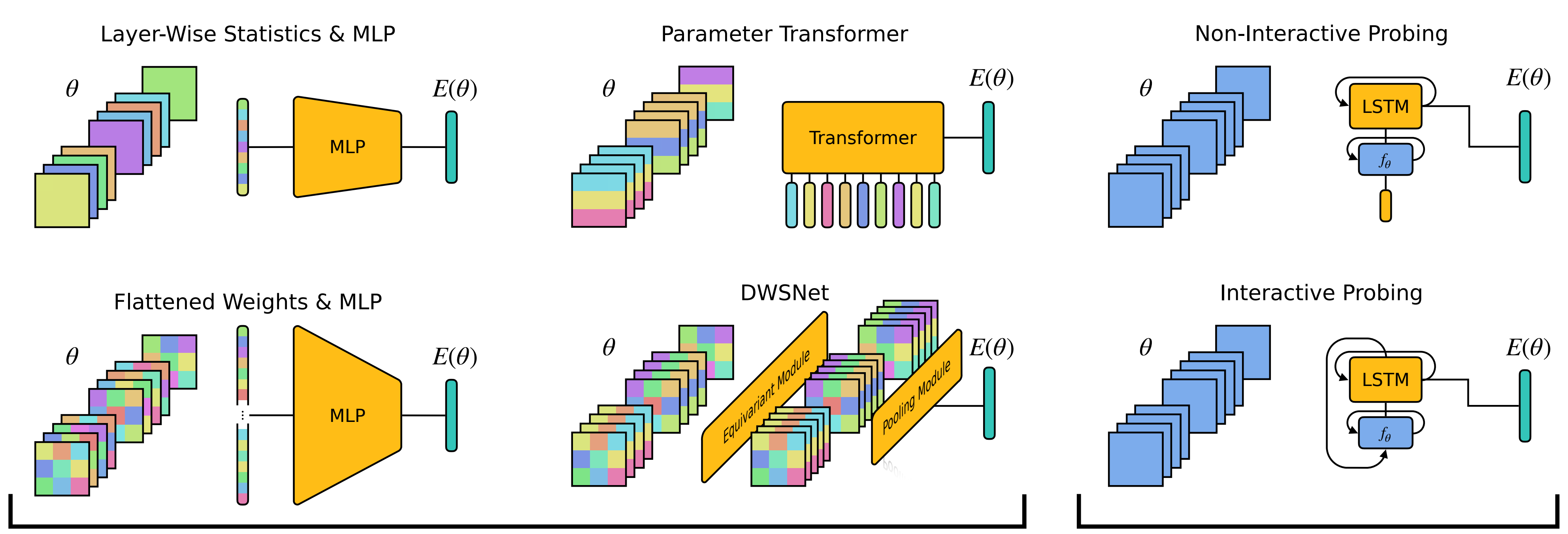
Recurrent Neural Networks are universal computers. Their weights can represent any program. Can we learn useful representations of the weights of RNNs?

Self-Supervised Learning of RNN Weight Representations

- Recurrent function f_θ with parameters θ is run in an environment, we get a trajectory $S_\theta = (x_1, y_1, x_2, y_2, \dots)$
- Encoder E generates representation $E(\theta)$
- Emulator A is conditioned on $E(\theta)$ and imitates f_θ



Types of Encoders for RNN Weights θ

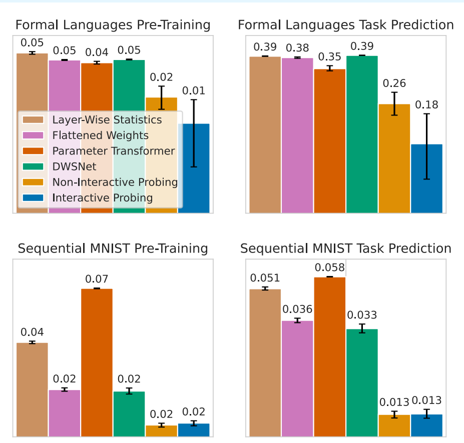


Mechanistic: Look directly at weights

Functionalist: Look at input-output mapping

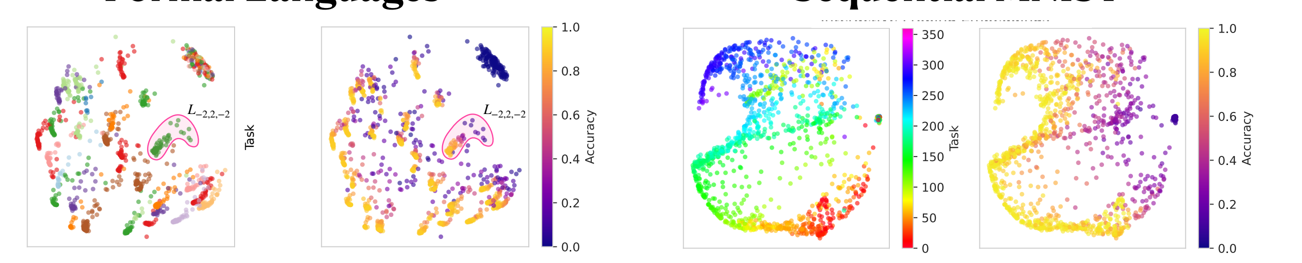
Results

- The learned representations can be used for various downstream tasks, such as task, performance or generalisation gap prediction
- Functionalist approaches are superior at more complex problems
- Only Interactive Probing learns generally useful representations for the Formal Languages dataset



Learned Embedding Spaces

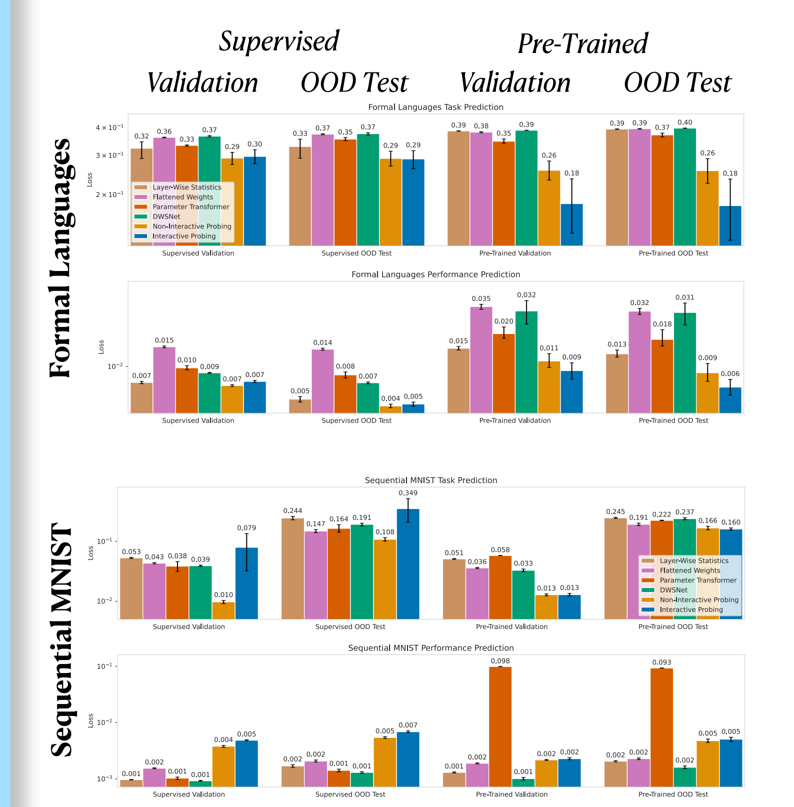
Interactive Probing



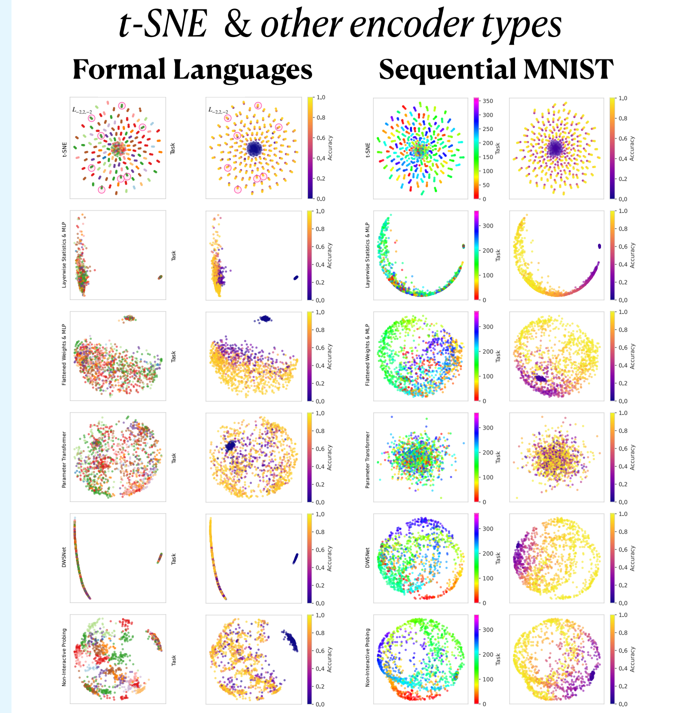
PCA of learned embedding spaces

Every dot represents a network f_θ from the validation datasets

Downstream Results

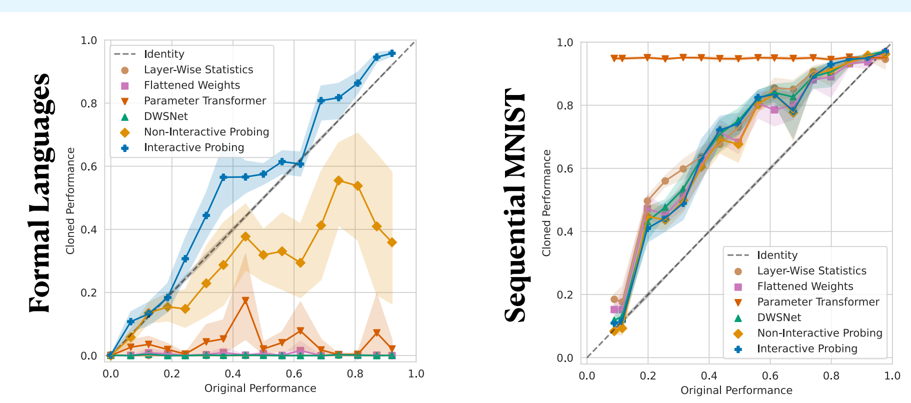


t-SNE & other encoder types



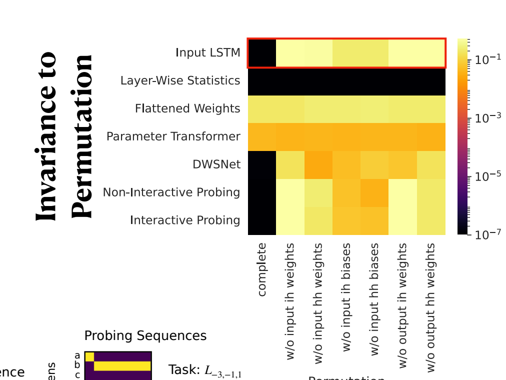
Original vs. Emulated Performance

f_θ's original performance vs. the performance of A_ξ's emulation based on $E_\phi(\theta)$. Validation set.



Encoder Properties

Encoder	Permutation Invariant	Universal Approx.	#Params	Type
Layerwise Statistics	Yes	No	const.	Mechanistic
Flattened Weights	No	Yes	$O(N^2)$	Mechanistic
Parameter Transformer	No	Yes	$O(N)$	Mechanistic
DWSNet	Yes	Yes	const.	Mechanistic
Non-Interactive Probing	Yes	No	const.	Functionalist
Interactive Probing	Yes	No	const.	Functionalist



Interactive Probing loss

vs. number of probing sequences *vs. probing sequence length*