We provide **evidence** that **gradient-based exact learning** of algorithms is **possible.**

We train a **simple**, **differentiable**, **Turing-complete** model on a **tiny** dataset of expert computation traces.

The result: **SotA** length-generalization on arithmetic tasks. No output errors as far as we could test.

Theory of Machine Learning Lab

EPFL

Exact Learning of Arithmetic with Differentiable Agents

Hristo Papazov, Francesco D'Angelo, Nicolas Flammarion

NEURAL INFORMATION PROCESSING SYSTEMS

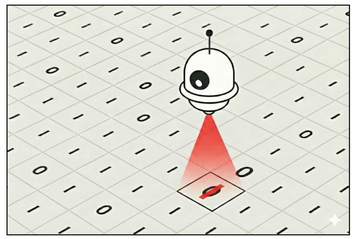
Grid Agents

A grid agent $M = (Q, \Sigma, \delta, q_0, q_f)$ is essentially a Turing machine moving continuously on a 2D tape:

$Q = \{q_0, q_1, \ldots, q_{d-1} = q_f\}$ — states

$\Sigma = \{\alpha_0, \alpha_1, \ldots, \alpha_{k-1}\}$ — grid alphabet

$\delta : Q \times \Sigma \to Q \times \Sigma \times \{U, D, L, R, S\}$ — transitions.



Policy-Trajectory Observations (PTOs):

$\pi_M(\sigma_0, \sigma_1, \ldots, \sigma_t) = a_t = (s_t, m_t)$ — history-dependent policy.

Observation at time t: (σ_t, s_t, m_t). **PTO**: $(\sigma_t, s_t, m_t)_{t=0}^{T}$.

Differentiable Finite-State Transducers (DFSTs)

A DFST $D = (A, B, C, h_0)$ is a minimalist **state-space model**, designed as a continuous counterpart to a grid agent with alphabet Σ.

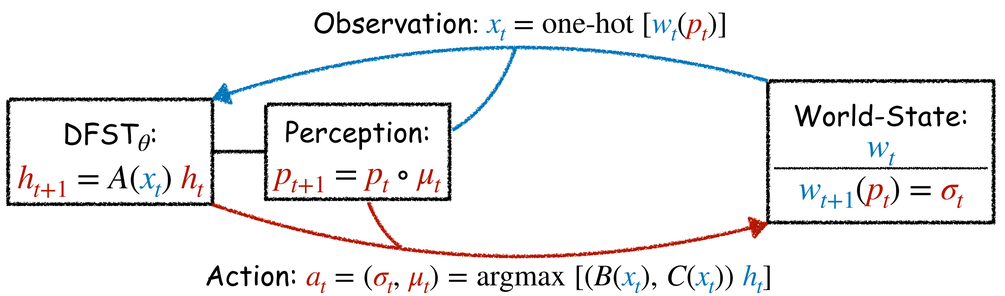
$$
\begin{aligned}
&h_0 \in \mathbb{R}^d \\
&A \in \mathbb{R}^{k \times d \times d} \longrightarrow h_{t+1} = A(x_t) h_t \\
&B \in \mathbb{R}^{k \times k \times d} \longrightarrow \hat{s}_t = B(x_t) h_t \\
&C \in \mathbb{R}^{k \times 5 \times d} \longrightarrow \hat{m}_t = C(x_t) h_t \\
&\#\text{params} = kd(k + d + r) = O(d^2)
\end{aligned}
$$

A DSFT simulating the grid agent M:

$\delta(q_i, s_j) = (q_{i'}, s_{j'}, m_{\ell}) \Rightarrow$

$h_0^{\delta} = (1,0,\ldots,0)^T,\ A^{\delta}[j, i', i] = 1,\ B^{\delta}[j, j', i] = 1,\ C^{\delta}[j, \ell, i] = 1.$

Given an expert agent M, we train on the **Next-Action Prediction** loss with **teacher-forcing**: $L(A, B, C, h_0) = \frac{1}{2T} \sum_{t=0}^{T} [(\hat{s}_t - s_t)^2 + (\hat{m}_t - m_t)^2]$.



Robust Length Generalization

Achieving 100% Exact-Match Accuracy on **hard** structured instances with N digits and 10 random pairs with N digits.

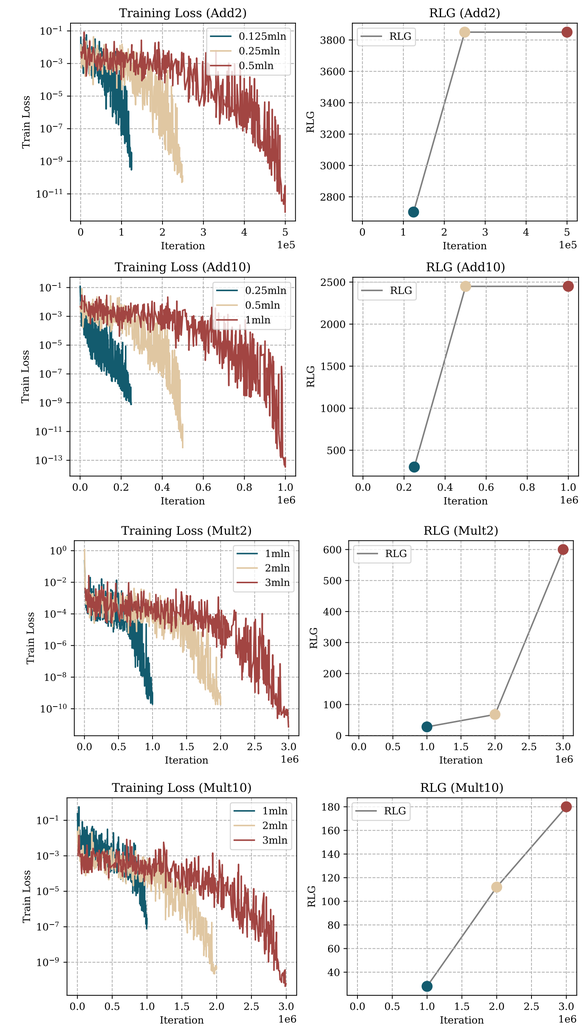


Table 1: Neural GPU vs. DFST on Arithmetic Tasks

Framework	Metric	add2	add10	mult2	mult10
Neural GPU	# Samples	$\sim$ 200k	N/A	$\sim$ 200k	N/A
	# Model Parameters	10368	N/A	10368	N/A
	Max Train Number Length	20	N/A	20	N/A
w/ Input-Output Data	Robust LG	$\leq$ 20	N/A	$\leq$ 8	N/A
	Probabilistic LG	$\geq$ **2000**	N/A	$\geq$ **2000**	N/A
DFST (ours)	# Samples	20	225	750	10000
	# Model Parameters	1020	8900	5280	162108
	Max Train Number Length	3	3	5	5
w/ PTO Data	Robust LG	$\geq$ **3850**	$\geq$ **2450**	$\geq$ **600**	$\geq$ **180**
	Probabilistic LG	$\geq$ **3850**	$\geq$ **2450**	$\geq$ **600**	$\geq$ **180**

Bold **numbers** indicate inability to test on longer inputs due to GPU memory constraints.