Neurosymbolic Learning of Symbolic Automata

No Author Given

No Institute Given

Abstract. We present a neuro-symbolic framework for learning symbolic finite automata (SFA). Our approach is motivated by Complex Event Recognition and applications, where SFA are typically used to represent patterns of critical complex events that need to be monitored over perceptual-level data streams, and where methods for learning such patterns are necessary. Our method learns the logical structure of the guards and the transitions in the SFA graph from sub-symbolic sequences in a fully differentiable fashion and supports the extraction of a compressed, crisp automaton with performance comparable to that of the neural model. We evaluate our method on synthetic benchmarks, as well as real-world event recognition tasks. Our results show orders of magnitude improvements in training times over symbolic SFA learning and better out-of-distribution generalization compared to neural baselines.

Keywords: Automata Induction · Neurosymbolic Learning · Complex Event Recognition.

1 Introduction

Complex Event Recognition (CER) systems [15, 1] detect critical events in streaming input using patterns that are typically defined as symbolic finite automata (SFA), i.e., automata whose transition-enabling conditions are logical predicates that need to be satisfied by the input, rather than symbols from a finite alphabet. Methods that learn and revise such SFA patterns, which are often unknown, or change over time, are important for dealing with the knowledge acquisition bottleneck in such applications. However, existing techniques for learning SFA [14, 18] are symbolic learners that scale poorly to complex logical representations and long sequences. On the other hand, recent differentiable automata induction methods [16, 31, 10] are limited to classical (i.e. non-symbolic) automata. Another important limitation of both sets of techniques is that they all operate on symbolic input, whereas CER systems often need to process raw, perceptual-level data (e.g. video, high-dimensional time series).

To address these issues we present ∂SFA (Differentiable Learning of Symbolic Automata), a neuro-symbolic (NeSy) framework for learning SFA, in a differentiable fashion, from perceptual sequences. ∂SFA parametrizes transition guards as neural logical formulas over a set of base predicates, implemented via fuzzy operators on top of a perceptual neural network. Sequences are processed

via time-varying transition matrix multiplications, which yield sequence acceptance probabilities. The fuzzy guards' relaxation and the fully differentiable forward pass through the neural SFA allow us to learn the guards structure from sequence-level labels using gradient-based optimization. Post-training, weight pruning, combined with a lightweight symbolic induction step based on Answer Set Programming, are used to extract a compact, fully interpretable SFA comparable in predictive performance to the differentiable model.

We evaluate ∂SFA on a synthetic benchmark, as well as real-world CER tasks from the domains of autonomous driving, personalized medicine and robotics. Our results show that ∂SFA achieves orders of magnitude improvements in training times over purely symbolic SFA learning, without compromising the quality of the learned model, and it outperforms purely neural learners in out-of-distribution generalization.

2 Related Work

Most existing work on learning SFA has focused on *active learning* algorithms [2] with membership and equivalence queries [11, 3, 22, 26, 13]. This setting is often impractical, since it requires "probing" the environment/system and assumes the existence of a "teacher", capable of certifying that a learned SFA matches the unknown pattern. In practice, however, what is often available is only a finite log of labeled executions and no access to such oracles. In contrast, in the *passive learning* setting, the learner receives a sample of positive and negative sequences and must infer an automaton consistent with the sample.This setting is often more natural in data-driven applications [13, 4]. For classical (i.e. non-symbolic) automata, state-merging algorithms such as RPNI [27] and its variants are widely used in this setting. While such methods can, in principle, be used for SFA learning via propositionalization, this typically incurs an exponential blow-up in the alphabet size. Moreover, these methods assume that the training sample is representative of the target language and overlook the noise in the sample strings, which may result in overfitting [18].

On the other hand, techniques dedicated specifically to SFA induction rely mostly on logic-based learning [18, 17, 25, 14]. These approaches suffer from scalability issues, since they use combinatorial search in a space that grows exponentially in the number of SFA states, guard predicates and sequence lengths. Besides, scalability is an issue for all purely symbolic SFA induction techniques, either passive, or active: efficient induction algorithms in either setting exist only for certain classes of SFA, specifically those defined over monotone Boolean algebras, e.g. the interval algebra over the reals, but not in the general case, e.g. the propositional algebra where the SFA guards are Boolean combinations of arbitrary predicates [13]. Aiming to sidestep the hardness of exact automata identification, differentiable approaches for automata induction have recently emerged [16, 31, 10]. but are limited to learning classical (non-symbolic) automata form symbolic strings, rather than sub-symbolic sequences.

Contrary to the aforementioned techniques, ∂SFA goes beyond fixed-alphabet symbolic sequences by learning, from raw sequential input, a NeSy relaxation of an SFA from which a crisp, interpretable automaton can be extracted.

A substantial body of work on differentiable Inductive Logic Programming is devoted to learning logic-based representations via logical reasoning relaxations and gradient-based techniques – we refer to [24] for an overview. However, existing methods focus on static, atemporal settings and none is capable of learning temporal logical models, as we do in this work.

3 Background and Problem Statement

We denote input sequences of length T by $x_{1:T} = \langle x_1, \ldots, x_T \rangle$, with each x_t being an element of a domain $\mathcal{X}$. The domain may consist of multivariate time-series vectors ($\mathcal{X} \subseteq \mathbb{R}^d$), images or video frames, or symbolic sequences, in which case each $x_t \in x_{1:T}$ is an interpretation, i.e., a set of logical facts representing what is true in the sequence at time t. In the case of perceptual sequences, we assume a neural network $f_\theta : \mathcal{X} \to \Delta(\mathcal{S})$ that maps each percept x_t to a probability distribution $p_t = f_\theta(x_t)$ over a set of symbols $\mathcal{S}$, with $p_t(s) = P_\theta(s \mid x_t)$ for each $s \in \mathcal{S}$ and $\Delta(\mathcal{S})$ denoting the probability simplex over $\mathcal{S}$.

We further assume a set Π of *base predicates* over the symbol set $\mathcal{S}$, which are provided beforehand as a language bias. Each predicate $\pi \in \Pi$ may either be a primitive condition over the input (e.g., $\mathsf{attribute} >< = \mathsf{value}$) or a higher-level predicate that can be used to reason over the neural output. As an example of the latter type of predicate, consider a sequence $x_{1:T}$ of MNIST digit images and π the predicate $\mathsf{even}/1$, which is satisfied by an image $x_t \in x_{1:T}$ when the true label of the image is an even number.

Symbolic Finite Automata (SFA). Let Π be as above and let $\mathbb{B}(\Pi)$ denote the Boolean algebra generated from Π by conjunction, disjunction, and negation [12]. A *symbolic finite automaton* (SFA) over Π is a tuple $\mathcal{A} = (Q, q_0, F, \Pi, \Delta)$, where $Q = \{1, \ldots, N\}$ is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting states, and $\Delta \subseteq Q \times \mathbb{B}(\Pi) \times Q$ is a finite set of transitions. Each transition is of the form $\langle u, \varphi, v \rangle$, where $\varphi \in \mathbb{B}(\Pi)$ is a *guard*. Given a sequence $x_{1:T}$, a run of $\mathcal{A}$ is a state sequence $q_{0:T} = \langle q_0, \ldots, q_T \rangle$ such that for each $t \in \{1, \ldots, T\}$ there is a guard φ_t with $(q_{t-1}, \varphi_t, q_t) \in \Delta$ and φ_t satisfied by x_t and any background knowledge that is potentially available, in which case we write $B \cup x_t \vDash \phi_t$. The run is accepting if $q_T \in F$, otherwise it is rejecting.

An SFA is *deterministic* if for every state $u \in Q$ and every pair of distinct outgoing transitions $\langle u, \varphi_1, v_1 \rangle, \langle u, \varphi_2, v_2 \rangle \in \Delta$, the guards φ_1 and φ_2 are mutually exclusive, i.e., for every input x_t, there exists at most one transition $(u, \varphi, v) \in \Delta$ such that $B \cup x_t \vDash \varphi$. In what follows we assume that an SFA is deterministic.

Learning Task. Given a training set of sub-symbolic sequences labeled as positive and negative, a perception network that maps percepts to symbols $\mathcal{S}$ and a set of base predicates Π over $\mathcal{S}$, the problem that we address in this work is to learn, in a differentiable fashion, an SFA with guards in $\mathbb{B}(\Pi)$ that accepts the positive and rejects the negative sequences.

4 Differentiable Learning of Symbolic Automata

We next proceed to the description of ∂SFA. Our method relies on differentiable relaxations of SFA, in particular, probabilistic variants of such automata. A probabilistic SFA operates on sequences of symbol distributions $p_{1:T} = \langle p_1, \ldots, p_T \rangle$ predicted by the perception network f_θ and is equipped by a *labeling function* λ, which assigns to each guard $\varphi \in \mathbb{B}(\Pi)$ and each symbol distribution $p_t \in \Delta(\mathcal{S})$ a guard satisfaction probability $\lambda(\varphi, p_t) \in [0, 1]$. The probability of a run $q_{0:T} = (q_0, \ldots, q_T)$ is the product $p(q_{0:T}) = \prod_{t=1}^{T} \lambda(\varphi_t, p_t)$ and the acceptance probability of an input sequence is the sum of the probabilities of all accepting runs over the sequence.

We represent the SFA guards as Disjunctive Normal Form (DNF) formulas over the base predicates Π. Guard evaluation is performed through differentiable conjunction and disjunction operators, yielding for each guard and time step a soft satisfaction value in $[0, 1]$. We refer to these values as truth degrees: under predicate independence they can be interpreted as probabilities, but in general they should be viewed as fuzzy approximations of crisp guard satisfaction.

The i-th clause (disjunct) of guard φ is a conjunction of literals indexed by j, where a literal is a base predicate $\pi \in \Pi$, or its negation $\neg\pi$. We denote by $\pi_\varphi^{i,j}$ the j-th literal in the i-th clause of guard φ, and by $p_\varphi^{i,j}(t)$ the probability that $\pi_\varphi^{i,j}$ holds at time t. If $\pi_\varphi^{i,j}$ is a positive literal, then $p_\varphi^{i,j}(t) = p(\pi_\varphi^{i,j} \mid x_t)$, otherwise $p_\varphi^{i,j}(t) = 1 - p(\bar{\pi}_\varphi^{i,j} \mid x_t)$. ∂SFA learns a weight, denoted by $w_\varphi^{i,j}$, for each candidate literal in each conjunction of a guard.

At time t while processing a sequence, the literal probabilities $p_\varphi^{i,j}(t)$ are directly provided by the perception network f_θ, e.g. via a softmax, or sigmoid layer, if the base predicate set Π consists only of the equality predicate, in which case a disjunct in a guard is just a conjunction of attribute-value pairs. In the case where higher-level predicates are present in the language bias, ∂SFA adopts an approach similar to [23], preceding learning by an "offline", knowledge compilation step, during which the base predicates are compiled into probabilistic circuits [9, 6]. These circuits are used for scalable, amortized probabilistic inference during learning, in order to compute base predicate probabilities from symbol distributions predicted by the neural network.

During learning we assume a fixed SFA graph of N states and we also assume that a guard consists of up to M disjuncts. Since the optimal values of these hyper-parameters are unknown, it practice they are handled via iterative deepening, starting from small values and progressively increasing them, if necessary, over consecutive training sessions, until convergence.

4.1 Literal Conjunction Layer

∂SFA learns clauses, i.e., the disjuncts in a DNF guard, via a dedicated conjunction layer. A literal weight $w_\varphi^{i,j} \in \mathbb{R}$ is split into a positive and a negative part via rectified linear units, so that $w_\varphi^{i,j,+} = \max\{w_\varphi^{i,j}, 0\}$ and $w_\varphi^{i,j,-} = \max\{-w_\varphi^{i,j}, 0\}$. The positive part measures how strongly the i-th positive literal should be enforced in each one of guard φ's clauses, whereas $w_{\varphi,-}^{i,j}$ measures how strongly

its negation should be enforced; when both are close to zero, the corresponding predicate is effectively irrelevant for that clause.

Given the literal probabilities and split weights, we define the value of guard φ's i-th clause at time t as

$$\mathsf{clause_val}^i_\varphi(t) = \sum_j \Big(w^{i,j,+}_\varphi \log p^{i,j}_\varphi(t) + w^{i,j,-}_\varphi \log\big(1 - p^{i,j}_\varphi(t)\big) \Big), \tag{1}$$

which after exponentiation yields

$$\exp\big(\mathsf{clause_val}^i_\varphi(t)\big) = \prod_j \big(p^{i,j}_\varphi(t)\big)^{w^{i,j,+}_\varphi} \big(1 - p^{i,j}_\varphi(t)\big)^{w^{i,j,-}_\varphi} \tag{2}$$

This is a *Weighted Product Model* [30] over the positive and negative literal probabilities. Under a base-predicate independence assumption, each literal may be viewed as a Bernoulli trial with success probability $p^{i,j}_\varphi(t)$ for a positive literal, or $1 - p^{i,j}_\varphi(t)$ for a negative one. When the weights are constrained to $\{0, 1\}$ the product reduces to the joint probability that all selected literals are satisfied, i.e., that a crisp conjunction holds. Allowing real-valued weights yields a soft conjunction: the clause value remains high only when all strongly weighted literals are well supported, while literals with near-zero weights are effectively ignored. Therefore, learning the weights amounts to learning which predicates should participate in the conjunction and with what strength.

Eq. (1) can be interpreted as a log-product fuzzy conjunction, where literal evidence is aggregated additively in log space and then exponentiated to recover multiplicative conjunction semantics. This design is important for numerical stability [32], since direct products of many probability values can become very small, weak gradient signals.

4.2 Clause Disjunction Layer

The actual guards, as disjunctions of clauses, are learned in a disjunction layer, modeled via a noisy-OR operator [28]. The latter aggregates all clause values in a guard into a single truth value for the entire guard, with a soft disjunctive semantics. For the i-th clause of guard φ, the exponentiated clause value from the conjunction layer lies in $[0, 1]$ and can therefore be interpreted directly as a fuzzy truth value for clause satisfaction. However, there is an important caveat: if all literal weights of a clause are zero, then $\mathsf{clause_val}^i_\varphi(t) = 0$ and, therefore, $\exp(\mathsf{clause_val}^i_\varphi(t)) = 1$. Despite being useless, such degenerate clauses effectively behave as tautologies if passed to noisy-OR, which would make the whole DNF guard be trivially (and erroneously) satisfied.

Clause weights. To prevent this, we associate each clause with a non-learnable *clause weight* derived deterministically from the magnitudes of its literal weights, so that only clauses with sufficient literal support are allowed to contribute substantially to the disjunction. Specifically, for guard φ's i-th clause we define its weight by $z^i_\varphi = \sigma\Big(\alpha\Big(\sum_j \big|w^{i,j}_\varphi\big| - \tau\Big)\Big)$, where σ is the sigmoid function, $w^{i,j}_\varphi$

denotes the effective total literal weight, $\alpha > 0$ controls the sharpness of the transition, and $\tau > 0$ is a threshold on the total literal mass of the clause. Thus, z^i_φ is a soft thresholding function of the quantity $\sum_j |w^{i,j}_\varphi|$: when the total literal mass is below τ, the clause weight is smaller than 0.5; when it is exactly τ, the clause weight is 0.5; and when it exceeds τ, the clause weight becomes larger than 0.5 and approaches 1 as the clause accumulates more literal mass. In this way, nearly empty clauses are downweighted, while clauses whose literals carry sufficient total weight are allowed to participate in the disjunction almost fully.

The threshold τ determines the level of total literal mass that is considered sufficient for a clause to count as structurally meaningful; α determines how abruptly the clause weight changes around that threshold. Larger values of α make the transition sharper, so that the clause weight behaves almost like a hard on/off decision, while smaller values yield a smoother transition. A sensible way to choose these parameters is to first decide what total literal mass should mark the onset of meaningful clause participation, and set τ accordingly, and then choose α so that clauses well below that threshold are strongly suppressed. For example, if one wants an empty clause to receive clause weight at most ε, then it suffices to choose α such that $\sigma(-\alpha\tau) \leq \varepsilon$, i.e., $\alpha \geq \frac{\log\left((1-\varepsilon)/\varepsilon\right)}{\tau}$. As a concrete example, with $\tau = 0.5$ and target suppression level $\varepsilon = 0.01$, one obtains $\alpha \geq 9.19$, so choosing $\alpha \approx 10$ yields strong suppression of empty and near-empty clauses.

For clauses with large total literal mass, the clause weight saturates near 1. For instance, if a clause contains several strong literals, then $\sum_j |w^{i,j}_\varphi| \gg \tau$, and hence $z^i_\varphi \approx 1$. This saturation complies with the role of the clause weight, which is structural, i.e. to suppress nearly-empty clauses arising from the conjunction layer, rather than ranking already meaningful clauses by strength.

To prevent this, we associate each clause with a non-learnable clause weight derived from the magnitudes of its literal weights: $z^i_\varphi = \sigma\Big(\alpha\Big(\sum_j \big|w^{i,j}_\varphi\big| - \tau\Big)\Big)$, where σ is the sigmoid function, $\tau > 0$ is a threshold on the total literal mass of the clause, and $\alpha > 0$ controls the sharpness of the transition around that threshold. Thus, z^i_φ acts as a soft thresholding factor: clauses with very small total literal mass are downweighted, while clauses with sufficient support are assigned weights close to 1. This suppresses nearly empty clauses, while allowing meaningful ones to contribute to the disjunction.
The truth value of guard φ at time t is computed by a weighted noisy-OR:

$$\mathsf{guard_val}_\varphi(t) = 1 - \prod_{i=1}^{R}\big(1 - z^i_\varphi \cdot \exp(\mathsf{clause_val}^i_\varphi(t))\big). \tag{3}$$

Noisy-OR is a standard soft disjunction operator with roots in probabilistic graphical models, where it is used to combine conditionally independent causal influences into the probability that at least one cause succeeds [28]. Here, if $z^i_\varphi \cdot \exp(\mathsf{clause_val}^i_\varphi(t))$ is interpreted as the probability that clause i is both active and satisfied, then the product term in (3) is the probability that no

clause fires, and its complement is the probability that at least one clause fires. In this way, noisy-OR provides a differentiable analogue of logical disjunction.

4.3 Transition Layer, Forward Pass and ∂SFA Training

We next describe how the guards' truth values are converted into soft state transitions in ∂SFA, and how these are propagated over time to determine whether an input sequence is accepted.

Let $\varphi_{u,v}$ denote the guard associated with the transition from state u to state v. At each time step t, the guard evaluation layers described above produce a truth value $g_{\varphi_{u,v}}(t) \in [0,1]$ for every transition guard. These values are then used to define a transition matrix over the automaton states.

Specifically, for each pair of states $(u,v) \in Q \times Q$, we first define the "transition logit" $r_{u,v}(t) = \frac{g_{\varphi_{u,v}}(t)}{\tau}$, where $\tau > 0$ is a temperature parameter. We then normalize with a softmax the logits that correspond to the outgoing transitions for each state, obtaining a probability distribution over the possible next states $v \in Q$, corresponding to the rows of a row-stochastic transition matrix $A_t \in [0,1]^{|Q|\times|Q|}$. The row-wise softmax over outgoing transitions induces "competition" among all successors of a given state. Although this does not enforce logical mutual exclusivity of the corresponding guards (and thus structural determinism of the corresponding crisp SFA), it does encourage them to specialize to different input conditions.τ controls the "sharpness" of this competition. Larger values of τ yield smoother transition distributions, whereas smaller values make the softmax more peaked, thereby encouraging increasingly decisive transition selection during training. ∂SFA anneals τ during training so as to gradually encourage learning of simpler automata with fewer outgoing transitions from any given state by reducing the learning signal for weakly used guards and biasing learning towards structural sparsity.

Given the sequence of time-dependent transition matrices $A_1, \ldots, A_T$, sequence acceptance probability is computed via a forward recursion scheme [29, 23]. The initial distribution over states is the one-hot distribution concentrated at q_0 (the start state), i.e. $\alpha_0(v) = 1$ if $v = q_0$, else 0 for any $v \in Q$. Then, for $t \geq 1$, $\alpha_t(v) = \sum_{u\in Q} \alpha_{t-1}(u)\, A_t(u,v)$, with $\alpha_t(v)$ representing the probability of being in state v after processing the first t elements of the input sequence. This is a standard forward pass for a non-homogeneous Markov chain, where the transition matrix may vary with time [29].

The acceptance probability assigned to an input sequence $x_{1:T}$ is then $p_\Theta(\text{acc} \mid x_{1:T}) = \sum_{v\in F} \alpha_T(v)$, where $F \subseteq Q$ is the set of accepting states and Θ denotes the trainable parameters of the model, namely the literal weights that determine the clause and guard truth values – note that this differs from θ, the parameters of the perception network f_θ. Training is performed from sequence-level binary labels using a binary cross-entropy (BCE) loss, backpropagating gradients from acceptance probabilities, all the way to literal weights.

4.4 Extracting a Crisp Model from the Neural SFA

After training converges we extract a crisp SFA with the purpose of validating the learned structure via domain expertise and recover from the neural model a crisp SFA that is compact, interpretable and usable by other downstream tools, such as event recognition and forecasting symbolic systems.

To this end, this post-processing step relies on weight pruning on the learned neural SFA, in addition to a lightweight symbolic induction step that further generalizes the crisp model. ∂SFA uses L_1-regularization during training to push as many literal weights as possible to zero, so as to favor sparsity. Therefore, many literals have already been zeroed-out and can be removed immediately. A number of additional small or redundant weights are eliminated in subsequent post-training steps. First, ∂SFA prunes entire guards whose removal does not increase the loss on the training by more than a given threshold. The same criterion is then applied at the literal level, removing literals from the surviving guards whenever their deletion has negligible effect on training performance.

Although weight pruning often yields a high-quality crisp SFA, it might be the case that the extracted crisp SFA contains redundant literals, which were deemed useful for the neural SFA during the pruning process, but are useless, or even harmful for the crisp SFA's performance, resulting in over-specific guards, and poor symbolic generalization. Moreover, although the training process pushes towards learning deterministic SFA via the tempered softmax during the forward pass, there is no guarantee that the crisp SFA that is extracted from the neural model is indeed deterministic: it can happen in practice that the DNFs that guard the outgoing transitions from some state q in the crisp SFA are not mutually exclusive, resulting in a non-deterministic crisp SFA. For these reasons, pruning is followed by a symbolic generalization step that searches for a better crisp structure within the reduced search space induced by the neural model.

For this final step we use ASAL [18], a framework for learning and revising SFA from symbolic event traces. ASAL encodes an SFA as an answer set program that combines a generic SFA interpreter, a specification of the automaton's structure (states and transitions), base predicates defined over event tuples, and transition guard rules expressed as Boolean combinations of background predicates. This formulation makes both the automaton structure and the guard definitions learnable through the strong connections of ASP to symbolic learning. In particular, ASAL casts SFA induction as an abductive, constraint-driven learning process: temporal structure and guard definitions are generated and tested against constraints related to predictive accuracy and model compression/minimality, in an effort to learn an optimal model.

The guards that survive neural pruning define a compressed starting-point search space, within which ASAL performs a final few-shot symbolic generalization step. Concretely, given a batch-size parameter B, we first select the B most "certain" training sequences, namely those whose argmax symbolic traces, predicted by the perception network, are assigned the highest confidence, as measured by their average entropy. These entropy-derived scores are further used as sequence weights, allowing ASAL to account for the inherent noise in

neural predictions.Once the surviving guards are revised from this initial batch, B is progressively augmented by the addition of a small number of weighted symbolic sequences, and the revision process is repeated until the performance of the induced crisp SFA on the training set no longer improves. ASAL is able to learn high-quality models from only a handful of informative traces, this procedure remains efficient despite operating over symbolic structure. Crucial for this is the fact that the heavily constrained hypothesis space that results from neural training makes the difficulty of the SFA induction process manageable, thus allowing the ∂SFA+ASAL hybrid to learn symbolic temporal structure from domains that was not previously possible. At the same time, the ASP formulation allows us to enforce declarative structural constraints during induction, including ensuring SFA determinism and other desired structural biases. The final outcome is therefore a crisp SFA that generalizes beyond the pruned neural model and is fully interpretable and structurally well-formed. We omit ASP-level technical details on the induction process and refer to [18].

4.5 A Note on End-to-End Training

In principle, the ∂SFA architecture can be used for end-to-end training, where a neural SFA is learned, while simultaneously a perception network is trained to map percepts to symbols. We do not address this learning setting in this work, where we focus on the SFA structure learning part, using pre-trained networks as oracles. The interesting case of indirect joint training, from downstream (sequence-level) labels alone, is extremely challenging in temporal domains. The few existing methods in the NeSy literature that address the joint training problem [21, 8, 5] focus entirely on atemporal tasks and operate by abducing weak latent concept labels from the downstream ones, in order to train the neural component. This approach is infeasible in temporal domains, where the number of abductive explanations for sequence-level failure points is immense, for substantially long sequences. A viable approach, in the absence of dense latent labels might be ground truth augmentation via complementary techniques, such as active learning, or data programming. ∂SFA is a good starting point for exploring such approaches.

5 Experimental Evaluation

In this section we present an empirical evaluation[1] of our approach, aiming to answer the following questions: **[Q1]:** can ∂SFA outperform symbolic SFA learning in terms of scalability? **[Q2]:** can ∂SFA learn high-quality, interpretable models comparable in predictive performance to those learned by symbolic learning techniques? **[Q3]:** can ∂SFA outperform purely neural sequence learning techniques in terms of out-of-distribution (OOD) generalization?

Datasets: to answer these questions we used several datasets from diverse domains of different modalities, including vision (image sequence datasets) and

[1] The implementation is available at https://anonymous.4open.science/r/dSFA-3E5C.

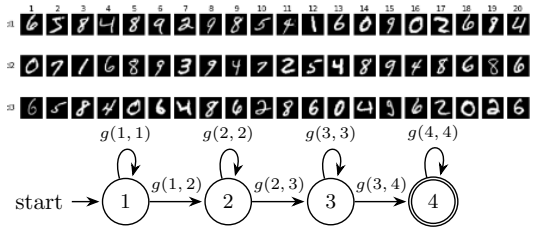
multivariate time series: (i) multivariate temporal MNIST: a synthetic, vision-based, temporal variant of MNIST arithmetic tasks that are being used in the NeSy literature; (ii) autonomous driving: a real-world, vision-based dataset of driving events; (iii) cancer stage progression: a synthetic, yet biologically significant, multivariate time-series dataset consisting of gene expression levels of virtual patients that transition from early to late cancer stages; (iii) robot deadlocks: a real-world, multivariate time-series dataset of mobile carrier robots' trajectories in a smart factory, annotated with robot deadlock incidents. The cancer progression and robot deadlock datasets were compiled in the corresponding pilots of the EU-funded project XXX[2]. The characteristics of the various datasets used in the experiments will be presented in detail in the rest of this section.

Methods compared: we compare ∂SFA to (i) ASAL, the only existing, purely symbolic SFA induction method. Given enough resources (time and memory) ASAL is guaranteed to learn an optimal SFA with the best trade-off between predictive performance and model complexity. However, it is often infeasible to scale this version to input data of increased length and dimensionality, since the induction process scales exponentially in those quantities; (ii) $\mathsf{ASAL}_{\mathsf{inc}}$, an incremental version of ASAL that attempts to improve ASAL's scalability by trading optimality of the learned models with efficiency. In particular, $\mathsf{ASAL}_{\mathsf{inc}}$ learns partial SFA from data mini-batches and iteratively refines them over the data in a Monte Carlo Tree Search (MCTS)-based approach, trying to approximate a global optimum from partial data views. Given moderate mini batch sizes, $\mathsf{ASAL}_{\mathsf{inc}}$ scales much better than ASAL, but it can often yield models of inferior quality; (iii) ∂SFA+ASAL the hybrid approach from Section 4.4; (iv) for OOD generalization comparison, purely neural methods, such as stacks of CNN vision components with LSTM and transformer layers to deal with temporal progression.

A note on Inductive Logic Programming (ILP): as far as purely symbolic learning methods are concerned, we attempted to evaluate ILASP [19], an ILP system based on Answer Set Programming, on the task of SFA induction from noisy argmaxed symbolic sequences predicted by trained perception neural networks. ILASP is the standard reference in the "new generation" of ILP systems [7] and is widely considered as the state-of-the-art in the field. It is also particularly well-suited for the learning tasks that we consider in this work, since it is capable of learning non-monotonic theories from sequential data (non-monotonicity via the Negation-as-Failure operator is required to capture guards' mutual exclusivity conditions, in order to express deterministic SFA as logic programs). Moreover, similarly to ASAL, ILASP is capable of learning from weighted sequences, with weights that correspond to the confidence of the neural argmax prediction, and thus it can, in principle, handle the inherent noise of NeSy pipelines that combine perception networks with symbolic learners.

We evaluated ILASP on a simplified, univariate version of the temporal MNIST task (to be presented shortly) that we used in the experiments, consisting of short (length-10) digit sequences, in a noise-free setting (unweighted sequences)

[2] Omitting project name and URL to preserve anonymity.



$g(1,1) \leftarrow not\ g(1,2).$
$g(1,2) \leftarrow even(d_1),\ odd(d_2),\ odd(d_3).$
$g(1,2) \leftarrow even(d_3).$
$g(2,2) \leftarrow not\ g(2,3).$
$g(2,3) \leftarrow odd(d_1),\ even(d_2).$
$g(2,3) \leftarrow even(d_1),\ even(d_3).$
$g(3,3) \leftarrow not\ g(3,4).$
$g(3,4) \leftarrow odd(d_3).$
$g(4,4) \leftarrow$ `#true`.

Fig. 1. Example of the multivariate temporal MNIST task. A length-20, 3-variate MNIST sequence, with each sub-sequence corresponding to to a digit, d_1, d_2, d_3, and a simple 4-state deterministic SFA, represented as a logic program, whose guards use the predicates *even/1, odd/1*. Guards of the form $g(i, j)$ specify the conditions for transitioning from state i to state j. Guards with the same head (e.g. $g(1, 2)$) express disjunctions. Self-loop guards dictate that irrelevant sequence points in-between the transition points should be skipped. The SFA has a single, accepting state (4), which is absorbing, as the $g(4, 4)$ guard indicates.

to simplify learning, using a 4-state target SFA with guards combining four predicates (digit_even, digit_odd, digit$\leq$ 3, digit$\geq$ 6), resulting in a search space of 283 rules corresponding to candidate guards. The evaluation showed that ILASP was unable to handle the task, terminating with a failure message after running for approximately two hours. It was able to identify the target SFA in about two minutes only after severely pruning the search space, keeping 30 rules that included the target SFA guards. For these reasons, we omit ILASP results from our comparison to symbolic learning SFA techniques.

For the vision-based tasks in the experiments, the perception modules (CNNs) are pre-trained with dense, latent concept supervision and the goal is to learn the SFA structure from sequence-level labels. For the multivariate time series learning tasks, we discretized the data to a number of symbols using the SAX algorithm [20] and, during training, we pass to ∂SFA near one-hot encodings of the symbols, simulating a near-perfectly trained network (e.g. an MLP) that predicts these symbols from the time series data.

All experiments were run on a Linux machine with an AMD Ryzen AI 9 HX 370 processor (12 cores, 24 threads), 32 GB of RAM, and an integrated AMD Radeon 890M GPU.

5.1 Multivariate Temporal MNIST

The purpose of this experiment was to stress-test the compared structure learning methods to perceptual input sequences of different lengths and to target

SFA of different complexity, in a principled and fully controlled fashion. To that end, we have created a temporal, multivariate version of an MNIST arithmetic task, illustrated in Figure 1. Each point in a sequence consists of three MNIST digit images (therefore, the dataset is 3-variate). The sequences are labeled as positive or negative, based on whether they are accepted by a target SFA or not. The target SFA patterns the were used in these experiments were defined in terms of the predicates $\mathsf{digit_even}, \mathsf{digit_odd}, \mathsf{digit} \leq X, \mathsf{digit} \geq X$ with $X \in \{2, 3, 4, 5, 6, 7\}$, and express sequential conditions that correspond to complex events that occur, or do not occur in the image sequences.

Data Generation: the target SFA patterns were defined as Answer Set programs and the Clingo ASP solver was used to abductively generate multivariate digit sequences that do, or do not satisfy the patterns, using acceptance constraints to guide the abductive search. We also used a dedicated ASP meta-encoding of Hamming distance-based diversity between the sequence-representing stable models returned by Clingo, in order to ensure that the generated sequences are adequately diverse. The generated symbolic sequences were paired with MNIST images, selecting an image with the same label as the corresponding digit in a sequence. The images required for this pairing well exceed the size of the original MNIST dataset. Therefore, to avoid data leakages due to repeated images in the training and testing sets, once the original MNIST dataset was exhausted during the generation process, we applied simple transformations to each image selected from that point onward, to ensure that each image in the generated sequences is unique. We used two target SFA patterns with 5 and 10 states respectively. Each guard in these patterns consists of up to 3 disjuncts, with each disjunct consisting of up to 4 conjunctions. For each of the two patterns we generated datasets with sequence lengths of 10, 30 and 50, for a total of 6 multivariate image sequence datasets.

Experimental setting: the purpose of the experiment was to recover the target SFA patterns from the training sequences. We compare ∂SFA to ASAL and $\mathsf{ASAL_{inc}}$ in terms of scalability and predictive accuracy. Given the three input images $\langle x_1^t, x_2^t, x_3^t \rangle$ at point t in an input sequence, a CNN predicts three probability vectors $v_i^t = \langle p(d_i = 0 \mid x_i^t), \ldots, p(d_i = 9 \mid x_i^t) \rangle$ for $i = 1, 2, 3$, with d_i referring to the i-th digit. ∂SFA learned directly from the CNN predictions. For the ASAL variants, the CNN was used to make argmax predictions from the image sequences, each inferred symbolic sequence was weighted by its average entropy and the learners used these weights as an extra bias during SFA induction, encoding a preference towards low-entropy sequences. We note that although this weighted sequence setting allows ASAL to simulate learning from neural outputs, while handling noise in a principled fashion, it is very challenging in terms of scalability, since it introduces additional WeightedMaxSAT optimization objectives in the learning process. It is also particularly challenging for $\mathsf{ASAL_{inc}}$, since global access to the weighted sequences is typically required in order to learn a good approximation to a global optimum.

We used a standard-for-MNIST CNN architecture from the literature, pretrained for 100 epochs on 1K images sampled randomly from the image se-

Pattern	$\|Q\|$	T	∂SFA		∂SFA+ASAL		ASAL		ASAL$_{\text{inc}}$	
			F_1	Time	F_1	Time	F_1	Time	F_1	Time
Pattern A	5	10	1.0	0.7	1.0	0.9	1.0	3.8	0.955	4.7
		30	1.0	1.2	1.0	2.3	1.0	7.3	0.702	4.9
		50	1.0	1.8	1.0	3.5	0.948	(t/o)	0.723	5.3
Pattern B	10	10	1.0	1.4	1.0	2.2	0.913	(t/o)	0.718	8.23
		30	1.0	2.6	1.0	3.3	0.887	(t/o)	0.725	8.82
		50	1.0	3.8	1.0	5.2	0.687	(t/o)	0.620	12.4

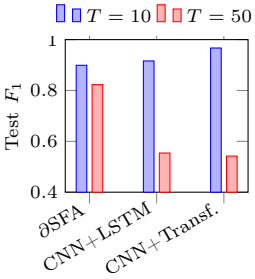


Table 1. Left: Test F_1-scores and training times (**in minutes**) for the two target SFAs with different numbers of states $|Q|$ and datasets with different sequence lengths T. **Right:** Out-of-distribution generalisation from length-10 to length-50 image sequences; all models are trained on length-10 sequences.

quences, achieving test-set F_1-scores for digit recognition that varied between 0.89 and 0.92 across the various runs. We used 70/30 train/test splits on 2K positive and negative sequences in each one of the six datasets and the reported results are averages from a 3-fold cross-validation, omitting the (generally negligible) standard deviations for brevity. ∂SFA was trained for 100 epochs in each run. ASAL was used with a cut-off induction time of 10 minutes. ASAL$_{\text{inc}}$ was run on mini-batches of 100 sequences for 20 MCTS iterations and a cut-off induction time per iteration of 1 minute.

Results I: scalability vs predictive accuracy. Table 1(Left) presents the results from the MNIST experiment in terms of F_1-scores and training times for the methods compared. Both ∂SFA and ∂SFA+ASAL are able to correctly recover the target SFA in all runs, achieving a perfect test-set F_1-score. Moreover, the extra overhead in induction time that stems from searching for an optimal crisp SFA using a neural SFA as a starting point (∂SFA+ASAL) is tolerable in general and does not exceed an additional 2 minutes on top of neural training. In contrast, ASAL exceeds the cut-off induction time of 10 minutes in most runs and the model returned within this time is sub-optimal, as can be seen by the imperfect test-set F_1-scores. Moreover, ASAL's predictive performance deteriorates rapidly in the more challenging settings (larger number of states in the target SFA, larger sequence lengths), indicating that ASAL requires significantly more time to learn a high-quality SFA, or adequately approximate the optimum in such settings. ASAL$_{\text{inc}}$ performs even worse. This is mainly due to the relatively small batch size that we used for ASAL$_{\text{inc}}$ in this experiment, which incurs an increased "myopia", caused by locally optimal solutions from which the system cannot effectively recover. ASAL$_{\text{inc}}$ can achieve significantly better results in this experiment with a larger mini-batch size, at the cost of increased learning times that approximate those of the batch version.

In contrast to the above ∂SFA can learn a perfect neural model in significantly less time and, when coupled with ASAL-based symbolic induction, a perfect, fully interpretable crisp model at the cost of a small increase in total training

times. Moreover, it is worth mentioning that ASAL has a very large memory footprint, requiring ≈13GB of RAM in this experiment, in contrast to the neural method that has no significant memory requirements. These results demonstrate empirically the efficacy of our new approach for differentiable SFA learning.

Results II: out-of-distribution generalization. we compare ∂SFA to neural baselines in terms of OOD performance. The models under comparison are trained on sequences of length 10 and tested on sequences of length 50 generated by the same underlying SFA pattern (the 5-state SFA from the previous experiment was used). For the neural baselines, we couple the perception CNN from the previous experiment with an LSTM and a transformer layer respectively. Table 1(Right) presents the results. All three models achieve comparable test F_1-scores on length-10 sequences. However, when evaluated on longer sequences, ∂SFA suffers a moderate performance reduction, while the purely neural models drop to near random guessing. ∂SFA's good performance is thanks to the learned logical structure, which is length-invariant, allowing handle this OOD setting.

5.2 Evaluation on Complex Event Recognition Domains

We present additional experiments where we compare ∂SFA to symbolic learning on tasks from challenging, real-world event recognition domains:

Vehicle overtake detection. The dataset consists of 250 vehicle image sequences of length 6-10 from the ROAD-R dataset[3], where vehicle overtake incidents occur (positive sequences), along with 750 "close negative" sequences of similar length. The purpose of learning here was to extract interpretable overtake SFA patterns from the raw data. The perception module was a 3D CNN, pre-trained on the entire ROAD-R dataset, making predictions about the vehicles' actions (e.g. *moving towards/away* from the camera, *turning* and so on) and locations (e.g. *incoming/outgoing lane, bus stop* etc). The experimental setting for ∂SFA, ASAL and $\mathsf{ASAL_{inc}}$ was identical to the MNIST experiment, with a 4-fold cross validation on 80/20 train/test splits.

Cancer stage progression in virtual patients. The dataset consists of gene expression time series for virtual patients that either progress from an early stage cancer profile to a late one (positive trajectory), or remain in early stage (negative trajectory). The application domain is personalized medicine and the purpose of learning is to learn interpretable SFA patterns of cancer progression at the gene level, as boolean combinations of gene expression thresholds for different genes. The virtual patient data have been generated via a VAE, trained on real transcriptomics data, and consist of 1494 positive and 264 negative trajectories of length 50 . A panel of 5 genes that sufficed for good results on this data was identified via feature importance techniques and exploratory learning, therefore, the training data consist of 5-variate time series[4]. The high class imbalance,

[3] https://sites.google.com/view/road-r/home

[4] We restricted the trajectory dimensionality to 5, in order for the symbolic techniques to work in a reasonable amount of time. In practice, ∂SFA can handle much larger dimensionalities

	∂SFA		∂SFA+ASAL		ASAL		$\mathsf{ASAL}_{\mathsf{inc}}$	
Dataset	Test F_1	Time	Test F_1	Time	Test F_1	Time	Test F_1	Time
Road-R	**0.962**	**0.8**	0.921	0.9	0.950	2.5	0.864	0.9
Cancer	**0.992**	**4.6**	0.968	5.3	0.975	180	0.822	15.8
Deadlocks	**0.903**	**3.8**	0.872	4.6	0.898	30	0.425	8.0

Table 2. Results from real-world datasets.

due to difficulties in the generation of realistic control (negative) data, made necessary the use of weighted learning schemes, both in purely symbolic (ASAL variants) and NeSy (∂SFA) learning. 4-fold cross-validation on 80/20 splits was used in this experiment.

Robot deadlock avoidance. The dataset consists of 100 robot trajectories with an average length of approximately $6.5K$ points and separate signals for various mobility attributes, such as robots' (x, y, z) coordinates over time, velocity, orientation etc. The trajectories were collected via simulations in a smart factory where the robots move around, executing of a set of pre-defined tasks. The trajectories are annotated with deadlock labels (i.e., incidents where two robots block each others' way, resulting in unforeseen delays). The purpose of learning was to learn deadlock SFA patterns, with the end-goal of using such patterns for early deadlock avoidance, via event forecasting techniques and path re-planning. To learn the patterns, the trajectories where compressed and segmented into smaller sequences, resulting in a training set of 3K, length-50 sequences with two signals for the robots' distance and velocity attributes. We used 4-fold cross-validation with 80/20 splits in this experiment.

The results are presented in Table 2. In all domains ∂SFA yields the best combined performance with the ∂SFA+ASAL hybrid following closely. ASAL learns near-optimal models, but runs for hours in some cases, while its incremental version lags behind the other methods in terms of predictive performance.

6 Conclusions and Future Work

We presented ∂SFA, a neuro-symbolic framework for differentiable learning of symbolic finite automata from perceptual sequences, combining neural induction of guards' structure with a lightweight symbolic generalization step based on Answer Set Programming. We evaluated our method on a challenging synthetic benchmark, as well as real-world Complex Event Recognition applications and demonstrated empirically superior performance over purely symbolic learning in terms of scalability, and over purely neural learning in terms of out-of-distribution generalization. Future work involves fully end-to-end training, where the perception module and the neural SFA are optimized jointly rather than relying on pre-trained perceptual components.

Declaration on the use of Generative AI: Generative AI was used only for language and writing assistance during manuscript preparation. All scientific content was produced, verified, and approved by the authors, who take full responsibility for the paper.

References

1. Alevizos, E., Artikis, A., Paliouras, G.: Complex event forecasting with prediction suffix trees. The VLDB Journal **31**(1), 157–180 (2022)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and computation **75**(2), 87–106 (1987)
3. Argyros, G., D'Antoni, L.: The learnability of symbolic automata. In: International Conference on Computer Aided Verification. pp. 427–445. Springer (2018)
4. Balachander, M., Filiot, E., Gentilini, R.: Passive learning of regular data languages in polynomial time and data. In: 35th International Conference on Concurrency Theory (CONCUR 2024). pp. 10–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2024)
5. Charalambous, T., Aspis, Y., Russo, A.: Neuralfastlas: fast logic-based learning from raw data. arXiv preprint arXiv:2310.05145 (2023)
6. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artificial Intelligence **172**(6-7), 772–799 (2008)
7. Cropper, A., Dumančić, S., Evans, R., Muggleton, S.H.: Inductive logic programming at 30. Machine Learning **111**(1), 147–172 (2022)
8. Dai, W.Z., Muggleton, S.H.: Abductive knowledge induction from raw data. arXiv preprint arXiv:2010.03514 (2020)
9. Darwiche, A., Marquis, P.: A knowledge compilation map. Journal of Artificial Intelligence Research **17**, 229–264 (2002)
10. Dhayalkar, S.R.: Symbolic feedforward networks for probabilistic finite automata: Exact simulation and learnability. arXiv preprint arXiv:2509.10034 (2025)
11. Drews, S., D'Antoni, L.: Learning symbolic automata. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 173–189. Springer (2017)
12. D'Antoni, L., Veanes, M.: The power of symbolic automata and transducers. In: International Conference on Computer Aided Verification. pp. 47–67. Springer (2017)
13. Fisman, D., Frenkel, H., Zilles, S.: Inferring symbolic automata. Logical Methods in Computer Science **19** (2023)
14. Furelos-Blanco, D., Law, M., Jonsson, A., Broda, K., Russo, A.: Induction and exploitation of subgoal automata for reinforcement learning. Journal of Artificial Intelligence Research **70**, 1031–1116 (2021)
15. Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., Garofalakis, M.N.: Complex event recognition in the big data era: a survey. VLDB J. **29**(1), 313–352 (2020)
16. Hannun, A., Pratap, V., Kahn, J., Hsu, W.N.: Differentiable weighted finite-state transducers. arXiv preprint arXiv:2010.01003 (2020)
17. Katzouris, N., Paliouras, G.: Learning automata-based complex event patterns in answer set programming. In: International Conference on Inductive Logic Programming. pp. 52–68. Springer (2022)
18. Katzouris, N., Paliouras, G.: Answer set automata: A learnable pattern specification framework for complex event recognition. In: 30th International Symposium on Temporal Representation and Reasoning (TIME 2023). pp. 17–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2023)

19. Law, M., Russo, A., Broda, K.: The ilasp system for inductive learning of answer set programs. In: The Association for Logic Programming Newsletter (2020)
20. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and knowledge discovery **15**(2), 107–144 (2007)
21. Liu, A., Xu, H., Van den Broeck, G., Liang, Y.: Out-of-distribution generalization by neural-symbolic joint training. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 12252–12259 (2023)
22. Maler, O., Mens, I.E.: A generic algorithm for learning symbolic automata from membership queries. In: Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday, pp. 146–169. Springer (2017)
23. Manginas, N., Paliouras, G., De Raedt, L.: Nesya: Neurosymbolic automata. arXiv preprint arXiv:2412.07331 (2024)
24. Marra, G., Dumančić, S., Manhaeve, R., De Raedt, L.: From statistical relational to neurosymbolic artificial intelligence: A survey. Artificial Intelligence **328**, 104062 (2024)
25. Muggleton, S.H., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Machine learning **94**(1), 25–49 (2014)
26. Okudono, T., Waga, M., Sekiyama, T., Hasuo, I.: Learning weighted finite automata over the max-plus semiring and its termination. arXiv preprint arXiv:2407.09775 (2024)
27. Oncina, J., García, P.: Inferring regular languages in polynomial update time. In: Pattern Recognition and Image Analysis (1990)
28. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Elsevier (2014)
29. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2), 257–286 (2002)
30. Tzeng, G.H., Huang, J.J.: Multiple attribute decision making: methods and applications. CRC press (2011)
31. Umili, E., Capobianco, R.: Deepdfa: Automata learning through neural probabilistic relaxations. arXiv preprint arXiv:2408.08622 (2024)
32. Van Krieken, E., Acar, E., Van Harmelen, F.: Analyzing differentiable fuzzy logic operators. Artificial Intelligence **302**, 103602 (2022)