NEURAL-BASED CLASSIFICATION RULE LEARNING FOR SEQUENTIAL DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering interpretable patterns for classification of sequential data is of key importance for a variety of fields, ranging from genomics to fraud detection or more generally interpretable decision-making. In this paper, we propose a novel differentiable fully interpretable method to discover both local and global patterns (i.e. catching a relative or absolute temporal dependency) for rule-based binary classification. It consists of a convolutional binary neural network with an interpretable neural filter and a training strategy based on dynamically-enforced sparsity. We demonstrate the validity and usefulness of the approach on synthetic datasets and on an open-source peptides dataset. Key to this end-to-end differentiable method is that the expressive patterns used in the rules are learned alongside the rules themselves.

1 INTRODUCTION

During the last decades, machine learning and in particular neural networks have made tremendous progress on classification tasks for a variety of fields such as healthcare, fraud detection or entertainment. They are able to learn from various data types ranging from images to timeseries and achieve impressive classification accuracy. However, they are difficult or impossible to understand by a human. Recently, explaining those black-box models has attracted considerable research interest under the field of Explainable AI (XAI). However, as stated by Rudin (2019), those aposteriori approaches are not the solution for high stakes decision-making and more interest should be placed on learning models that are interpretable in the first place.

Rule-based methods are interpretable, human-readable and have been widely adopted in different industrial fields with Business Rule Management Systems (BRMS). In practice however, those rules are manually written by experts. One of the reasons manually-written rule models cannot easily be replaced with learned rule models is that rule-base learning models are not able to learn as expressive rules with higher-level concepts and complex grammar (Kramer, 2020). Moreover, due to the lack of latent representations, rule-based learning methods underperform w.r.t. state-of-the-art neural networks (Beck & Fürnkranz, 2021).

Classical classification rule learning algorithms (Cohen, 1995; Breiman et al., 1984; Dash et al., 2018; Lakkaraju et al., 2016; Su et al., 2016) as well as neural-based approaches to learn rules (Qiao et al., 2021; Kusters et al., 2022) (or logical expressions with Riegel et al. (2020)) do not provide the grammar required to learn classification rules on sequential data. Numerous approaches for learning classification rules on sequential data in the field of sequential pattern mining have been studied in the past such as Egho et al. (2015); Zhou et al. (2013); Holat et al. (2014) but with a different goal in mind : improve the performance of extracted patterns for a fixed rule grammar as opposed to extending the rule grammar. Another domain of research focuses on training binary neural networks to obtain more computational efficient model storing, computation and evaluation efficiency (Geiger & Team, 2020; Helwegen et al., 2019). It comes with fundamental optimization challenges around weights updates and gradient computation.

In this paper, we bridge three domains and introduce a binary neural network to learn classification rules on sequential data. We propose a differentiable rule-based classification model for sequential data where the conditions are composed of sequence-dependent patterns that are discovered *alongside* the classification task itself. More precisely, we aim at learning a rule of the following

structure: *if* pattern *then class* = 1 *else class* = 0. In particular we consider two types of patterns: local and global patterns as introduced in Aggarwal (2002) that are in practice studied independently with a local and a global model. A local pattern describes a subsequence at a specific position in the sequence while a global pattern is invariant to the location in the sequence (Fig 2). The network, that we refer to as Convolutional Rule Neural Network (CR2N), builds on top of a *base rule model* that is comparable to rule models for tabular data presented in Qiao et al. (2021); Kusters et al. (2022).

The contributions of this paper are the following: i) We propose a convolutional binary neural network that learns classification rules together with the sequence-dependent patterns in use. ii) We present a training strategy to train a binarized neural network while dynamically enforcing sparsity. iii) We show on synthetic and real world datasets the usefulness of our architecture with the importance of the rule grammar and the validity of our training process with the importance of sparsity.

2 Base Rule Model

The base rule model we invoke is composed of three consecutive layers (Fig 1). The two last layers respectively mimic logical AND and OR operators (Qiao et al., 2021; Kusters et al., 2022). On top of these layers, we add an additional layer that is specific for categorical input data and corresponds to an OR operator for each categorical variable over every possible value it can take.

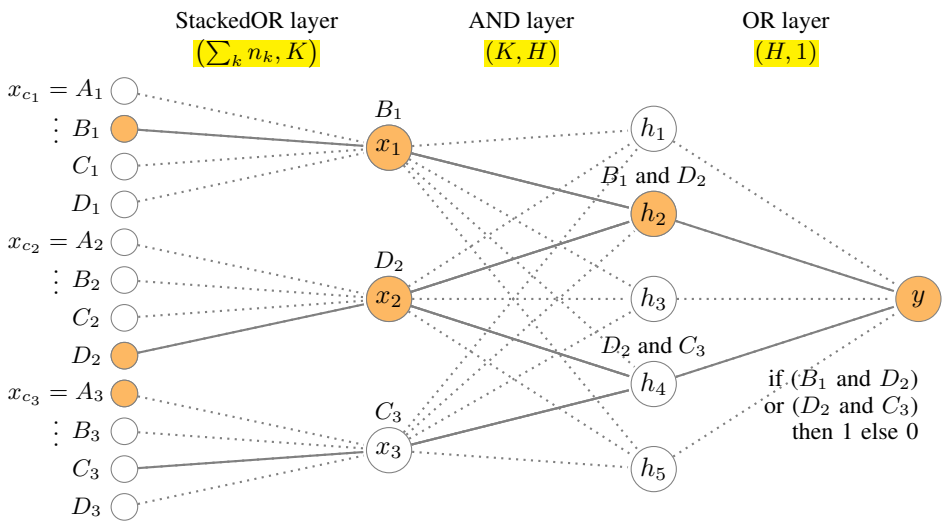

Figure 1: Example of a trained *base rule model* architecture for the rule *if* $(B_1$ *and* $D_2)$ *or* $(D_2$ *and* $C_3)$ *then 1 else 0* on 3 categorical variables x_{c_1}, x_{c_2} and x_{c_3} ($x_{c_k} \in \{A_k, B_k, C_k, D_k\}$). For simplicity, the truth value of $x_{c_1} = B_1$ is replaced by B_1 for example. Plain (dotted) lines represent activated (masked) weights. An example evaluation of the model is represented with the filled neurons (neuron=1) for the binary input $x_{c_1} = B_1$, $x_{c_2} = D_2$ and $x_{c_3} = A_3$.

The AND layer takes binary features (which are atomic boolean formulae) as input and outputs to the OR layer. The output of the OR layer is mapped to the classification label y. These layers have binary weights specifying the nodes that are included in the respective boolean expression (conjunction or disjunction). In other words, this network implements the evaluation of a DNF and has a direct equivalence with a binary classification rule like *if* $(A \wedge B) \vee C$ *then class* = 1 *else class* = 0, where A, B and C are binary input features (atoms in logical terms). In this paper, we focus on supervised binary classification where we predict the label $y \in \{0, 1\}$ given input data $\boldsymbol{x}$.

The base rule model is illustrated in Fig 1 and is composed of three binary neural layers.

- Input neurons $\boldsymbol{x}$, are binarized input features of size K ($\boldsymbol{x}_c$ are one-hot encoded categorical input features of size $\boldsymbol{n}$).
- Hidden neurons $\boldsymbol{h}$, are conjunctions of the input features of size H.
- Output neuron $\boldsymbol{y}$, is a disjunction of the (hidden) conjunctions.

We assign to each of boolean operations, i.e. AND and OR operations, a binary weight ($\boldsymbol{W}_{and}$ and $\boldsymbol{W}_{or}$ respectively) that plays the role of a mask to filter nodes with regards to their respective logical operation. For the sake of simplicity, we did not extend the model with a logical NOT operation.

The disjunction operation is implemented as,

$$y = \min(\boldsymbol{W}_{or}\boldsymbol{h}, 1). \tag{1}$$

If none of the neurons are activated then $y = 0$, and $y = 1$ if at least one is.

For the conjunction operation, we use the De Morgan's law that express the conjunction with the OR operator $A \wedge B = \neg(\neg A \vee \neg B)$. Combined with Eq 1, we obtain:

$$\boldsymbol{h} = \neg(\min(\boldsymbol{W}_{and}(\neg\boldsymbol{x}), \mathbf{1})) = \mathbf{1} - \min(\boldsymbol{W}_{and}(\mathbf{1} - \boldsymbol{x}), \mathbf{1}). \tag{2}$$

StackedOR Input Layer As defined previously, the AND layer takes binary input features as input. In this paper, we propose to add an additional layer for categorical data. A categorical variable $\boldsymbol{x}_c$ can take one value α_i^c out of a fixed number of possible values n e.g. $\{\alpha_0, \ldots, \alpha_3\} = \{A, B, C, D\}$. Without any additional layer, it requires a one-hot encoding to be provided as input to the AND layer. Binary inputs $x_c = A$ and $x_c = B$ are then given as input to the AND layer that can in theory represent the impossible expression $x_c = A \wedge x_c = B$ i.e the model has to learn the hidden categorical relationship between the one-hot encoded variables. To prevent learning a distribution we already know, we deepen the model with the addition of a stacked architecture of OR layers as input of the AND layer as shown in Fig 1. This structure is defined by K weights, $\boldsymbol{W}_{stack}^k$, for each input categorical variables and will be referred to as the StackedOR layer with $\boldsymbol{W}_{stack}$ weights.

To conclude, the base rule model is composed of a StackedOR layer for categorical variables, a logical AND layer and an OR layer. The formal grammar that this architecture can express is specified with the following production rules (see Appendix A for the full grammar):

$$\begin{array}{rcl}
\text{rule} & \rightarrow & \textit{if}\ \text{base expression}\ \textit{then class} = 1\ \textit{else class} = 0 \\
\text{base expression} & \rightarrow & \text{conjunction} \mid \text{conjunction} \vee \text{base expression} \\
\text{conjunction} & \rightarrow & \text{predicate} \mid \text{predicate} \wedge \text{conjunction} \\
\text{predicate} & \rightarrow & \text{categorical expression} \mid \text{literal} \\
\text{categorical expression} & \rightarrow & \text{categorical literal} \mid \text{categorical literal} \vee \text{categorical expression} \\
\text{categorical literal} & \rightarrow & \underline{x_c = \alpha_0^c} \mid \ldots \mid \underline{x_c = \alpha_{n_c}^c}\ (\text{or simply}\ \underline{\alpha_0^c} \mid \ldots \mid \underline{\alpha_{n_c}^c}) \\
\text{literal} & \rightarrow & \underline{x_1} \mid \underline{x_2} \mid \ldots
\end{array} \tag{3}$$

This grammar is also limited by the model architecture: *conjunction* contains at most one occurrence of each *predicate* and the total number of *conjunction*(s) is bounded by the number of hidden nodes.

3 Convolutional Rule Neural Network

Our main contribution is to extend the base rule model for sequential data. We apply the base rule model as a 1D-convolutional window of fixed length $l \in \mathbb{N}$ over a sequence and retrieve all outputs as input for an additional disjunctive layer which we refer to as the ConvOR layer as shown in Fig 2 [1]. The base rule model learns a DNF over the window size length and the ConvOR layer indicates where along the sequence that logical expression is true. If the evaluation of the logical expression is true all along the sequence then it can be described as a *global pattern*, otherwise the learned pattern represents a *local pattern*.

The model input is now of size $\sum_k l \times n_k$ and output of StackedOR layer (or input of the AND layer) is $l \times K$. Other dimensions are not impacted. For simplicity in the following, K is fixed to 1 i.e. input data is composed of one categorical variable evolving sequentially. The method is still valid for $K > 1$. Fig 1 is also still valid with a change of index, k is now referring to the position in the window of size l.

With this approach, different sequence-dependent expressions can be extracted and their nature depends on the weights of the ConvOR layer (Fig 2). If all the weights, $\boldsymbol{W}_{conv}$, of the ConvOR layer

[1] A natural extension for sequential data of the base rule architecture would be to extend it with an explicit recursion of the base rule model, similar to a RNN. This approach was tested but faced the same limitations as any classical RNNs, i.e., vanishing gradients and only captures short-term dependencies.

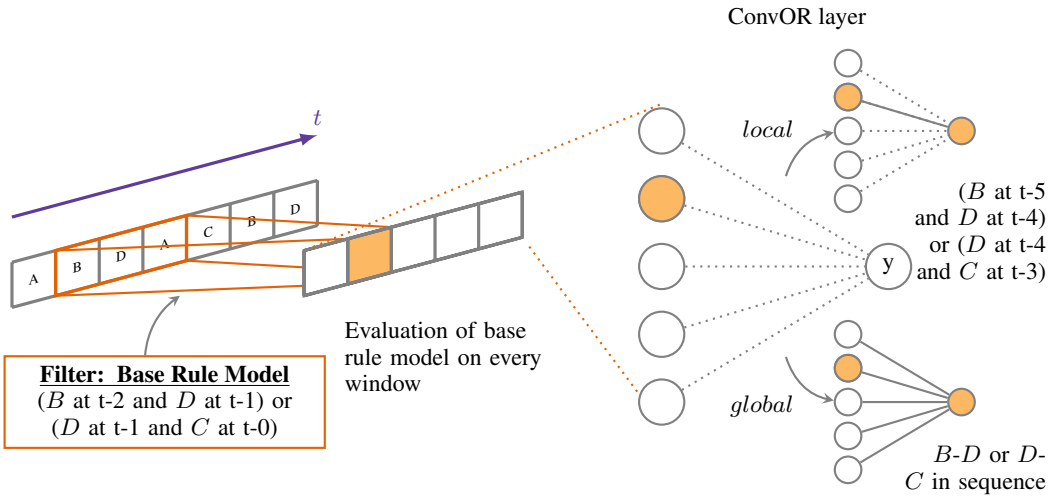

Figure 2: Example of a trained CR2N architecture. The base rule model is applied as 1D-convolutional window over the sequence (i.e. sliding window). The resulting boolean values are given as input of the ConvOR layer which indicates through its activated weights where along the sequence the expression learned by the base model is true. The output of the ConvOR layer is mapped to the label of the sequence y. For local patterns, the base model expression needs to be shifted accordingly to the ConvOR layer weights. For a real-domain application like fraud detection, by providing meaning to B, C and D, we could have for example if "receiving a transaction of amount X"(B) is followed by "emitting a transaction of amount X" (D) or "emitting a transaction of amount X"(D) is followed by "closing the bank account"(C) then class=fraud.

are activated (i.e. equal to 1), the logical expression learned by the base model is true in all the sequence: a *global* pattern is learned. If only some of the weight of the ConvOR layer are activated, the logical expression learned by the base model is valid only in the window associated to that weight: a *local* pattern is learned. The base model logical expression is modified accordingly to match that shift (see example in Fig 2 with a shift of 3 sequential steps).

The obtained weights thus translate to a rule grammar with the following production rules:

$$\begin{array}{lcl} \text{rule} & \rightarrow & \textit{if}\ \text{expression}\ \textit{then class} = 1\ \textit{else class} = 0 \\ \text{expression} & \rightarrow & \text{local pattern} \mid \text{global pattern} \end{array} \tag{4}$$

We introduce t, the position when the last observation in a sequence was made. With t being our reference, in a sequence of size $N \in \mathbb{N}$, $t - i$ refers to the moment of the ith observation before t ($0 \leq i \leq N - 1$). A, B, C and D are toy binary input possible values for our categorical variable x_c (they cannot be activated simultaneously at the same position t in the sequence). With those definitions, we list below examples of different sequence-dependent expressions that can be expressed with the proposed architecture (see Fig 2):

A **local pattern** is an expression composed of predicates that are true at a specific position i, for example A *at t-15*. Based on Eq 3 we have:

$$\begin{array}{lcl} \text{local pattern} & \rightarrow & \text{base expression} \\ \text{predicate} & \rightarrow & \text{categorical expression}\ \textit{at t-}i \mid \text{literal}\ \textit{at t-}i. \end{array} \tag{5}$$

A **global pattern** is an expression describing the presence of a pattern anywhere in the sequence, for example B-D *in sequence* is a global pattern where "$-$" sign refers to "followed by" and "$*$" correspond to any unique literal (equivalent to $\forall i \in [0; N-1]$, B *at t-i-1 and* D *at t-i*) . If inputs are sequences of characters, global patterns can be compared to simple regular expressions supporting the logical OR (metacharacter'[ ]'). Based on Eq 3 we have:

$$\begin{array}{lcl} \text{global pattern} & \rightarrow & \text{base expression} \\ \text{conjunction} & \rightarrow & \text{predicate} \mid * \mid \text{predicate} - \text{conjunction} \end{array} \tag{6}$$

Additional special cases can be pointed out such as the learning of a global pattern over an interval (e.g. *B-*-D in window [t-6; t-3]*) or the learning of sequence characteristics dependant expression such as *4* $\leq$ *len(sequence)* $\leq$ *6* based on the sequence length (not shown on Fig 2 but it corresponds to a specific case where the base model has learned an always true rule). Also, it is important to note that *baseexpression* and *conjunction* in both grammars are bounded by the fixed window size l.

To ensure full equivalence between the model and rule, sequences boundaries need to be considered, especially for global patterns. All sequences are padded on both ends with a sequence of 0 of size $l-1$ (not shown for simplicity on Fig 2). Also sequences of different lengths are supported by creating a model based on the maximal available sequence length M in the data and padding shorter sequences with a sequence of 0 of appropriate length. ConvOR layer input size is then $M+l-1$.

With this one architecture we can model both local and global patterns. However for optimization reasons detailed below, we choose to differentiate the two into two distinct models: a **local** and a **global model**. The ConvOR layer weights for the global model are set and fixed to 1 during training.

4 Training Strategy

To overcome training challenges attributed to binarized neural networks (Geiger & Team, 2020), we use latent weights and enforce sparsity dynamically. We define a loss function that penalize complex rules and the model is trained via automatic differentiation (Pytorch) with Adam optimizer.

Latent weights The binary model parameters introduced above ($\boldsymbol{W}_{and}$, $\boldsymbol{W}_{or}$, $\boldsymbol{W}_{stack}$, $\boldsymbol{W}_{conv}$) are trained indirectly via the training of a continuous parameter loc which is activated (binarized) by a sigmoid function (Kusters et al., 2022). With such binary weights and continuous relaxation Eq 1 and 2 are differentiable with nonzero derivatives (Kusters et al., 2022). As opposed to when using a straight through estimator (Qiao et al., 2021), non-zero gradients are ensured during the backward pass. To overcome training limitations, we use a hard concrete distribution (Qiao et al., 2021; Louizos et al., 2018). It rescales the weights and the random variable introduced during training prevents from obtaining local minima (Appendix B). Weight values are in $[0,1]$ during training, while for testing and rule extraction, a Heaviside is applied to them (≥ 0.5) to ensure strict binarization.

Loss function We define the loss function $\mathcal{L}$ composed of a mean-squared error component along with a regularization term that penalize the complexity of the rule,

$$\mathcal{L} = \mathcal{L}_{mse} + \lambda \Pi \tag{7}$$

That regularization term Π, or *penalty*, evaluates the number of terminal conditions in the rule. In practice we use $\lambda = 10^{-5}$. For a layer_n of input size I and output size O, the number of terminal conditions per output corresponds to the weighted sum of the number of terminal conditions of each output of the previous layer i.e. $\mathbf{\Pi}_{\text{layer}_\text{n}} = \sum_I \mathbf{W}_{\text{layer}_\text{n}} \mathbf{\Pi}_{\text{layer}_{\text{n}-1}}$, a vector of size O. For the first layer, the StackedOR layer, $\mathbf{\Pi}_{stack}$ is defined as the sum over the input dimension of the weights and we can then express the number of terminal conditions of the base rule model Π_{base}.

$$\mathbf{\Pi}_{\text{layer}_0} = \mathbf{\Pi}_{stack} = \begin{bmatrix} \sum \mathbf{W}^1_{stack} \\ \vdots \\ \sum \mathbf{W}^K_{stack} \end{bmatrix}, \qquad \Pi_{base} = \sum \mathbf{W}_{or} \sum \mathbf{W}_{and} \mathbf{\Pi}_{stack} \tag{8}$$

For optimizing Π for local patterns, we have to minimize the activated ConvOR layer weights. For global patterns, we want them to all be activated. A condition could be set on the sum of ConvOR layer weights (Eq 9) to shift from one optimization problem to the other but with loss of continuity and thus differentiability (interesting values of τ being $M+l-1$, the ConvOR layer input size, that would correspond to all ConvOR layer weights being equal to 1, or $M-l+1$ that would allow for $2(l-1)$ weights to be 0, and corresponds to the padding required for properly accounting for sequence boundaries (Section 3)).

$$\Pi_{local} = \Pi_{base} \sum \mathbf{W}_{conv}, \qquad \Pi_{global} = \Pi_{base}, \qquad \Pi^* = \begin{cases} \Pi_{global} & \text{if } \sum \mathbf{W}_{conv} \geq \tau \\ \Pi_{local} & \text{otherwise} \end{cases} \tag{9}$$

Table 1: Ground truth applied on sequences of letters (A to F) to generate synthetic unbalanced datasets 1, 2, 3 and 4 along with the distributions of the positive class. In the patterns, t refers to the position when the last observation in a sequence was made. Balanced datasets with same ground truth are generated and are referred to as the dataset number followed by the letter *b* (Appendix D).

Ground truth	#	**Distribution (%)**
C at t-4	1	14.2
A at t-6 and C at t-4	2	1.5
(A at t-6 and C at t-4) or (B at t-5 and C at t-3)	3	3.6
B-D in sequence	4	20.4

Due to non continuity of Π^* in Eq 9, we choose to have two models with the same architecture for the two cases: the *local* and the *global model* respectively more relevant for their associated pattern. For the local model, all weights are trainable and $\Pi = \Pi_{local}$. For the global model, weights in the ConvOR layer are fixed and set to 1, and $\Pi = \Pi_{global}$.

Enforced sparsity Sparsity of the model is crucial to learn concise expressions, the model needs to generalize without observing *all* possible instances at training time. The first requirement for that matter is sparsity in the base rule model. In addition to the regularization term in the loss function, we propose to use a sparsify-during-training method (Hoefler et al., 2021) and dynamically enforce sparsity in weights from 0% to an end rate r_f set to 99% in our case (Lin et al., 2020). Sparsify-during-training method can also benefit the quality of the training in terms of convergence by correcting for approximation errors due to premature pruning in early iterations but is highly dependant on the sparsification schedule (Hoefler et al., 2021).

Every 16 iterations s and for a total of s_f training iterations, every trainable weight is pruned with a binary mask, $\mathbf{m}$ (of size of its associated weight and applied with Hadamard product ($\odot$)) (Lin et al., 2020; Zhu & Gupta, 2017). We propose a mask based on the maximum of weight magnitude loc and pruning rate r (Zhu & Gupta, 2017) making the assumption that it contributes to generalization (Eq 10). This strategy can be more aggressive than state-of-art contributions (Lin et al., 2020) due to its dependency to the loc maximum value. During training, the model with the highest prediction accuracy on validation dataset and the highest sparsity (evaluated at each epoch) is kept.

$$r = r_f - r_f\left(1 - \frac{s}{s_f}\right)^3, \qquad m_{i,j} = |loc_{i,j}| \geq r \times \max_{\mathrm{i,j}}(loc), \qquad \hat{\mathbf{W}} = \mathbf{W} \odot \mathbf{m} \tag{10}$$

Additional training optimizations have been tested out such as for example using a binarized optimizer (Helwegen et al., 2019; Geiger & Team, 2020), adding a scheduled cooling on the sigmoid of the binarized weights, alternating the training of each layer every few epochs (Qiao et al., 2021) or using a learning rate scheduler. Those techniques are not presented here but would be of interest for improving results on specific datasets.

5 Experiments

In order to evaluate the validity and usefulness of this method, we apply it to both synthetic datasets and UCI membranolytic anticancer peptides dataset (Grisoni et al., 2019; Dua & Graff, 2017).

Synthetic Datasets We propose 8 synthetic datasets based on 4 ground truth expressions in both balanced and unbalanced distributions for discovering simple binary classification rules with local or global patterns as shown in Table 1. There are 1000 sequences of letters (A to F) of different length from 4 to 14 letters in each of them (Mean around 9$\pm$3). Generation is detailed in Appendix D.

Peptides Dataset Besides the synthetic datasets, real-world UCI anticancer peptides dataset composed of labeled one-letter amino acid sequences, is used (Grisoni et al., 2019; Dua & Graff, 2017). The multi-classification problem is transformed into a binary classification problem is the same manner as Nwegbu et al. (2022) (see Appendix C). Sequence length are from 5 to 38 letters (Mean: 17 $\pm$ 5.5) and positive class distribution is 79%.

Experimental Setting All datasets are partitioned in a stratified fashion with 60% for training, 20% for validation and 20% for testing datasets and we use a batch size of 100 sequences. The hidden size in the base rule model is set to the double of the input size of the AND layer (which is the window size of the convolution). More details on experimental setting can be found in Appendix E. At each epoch (200 in total), we evaluate the model against the validation dataset and keep the model with the highest accuracy and in case of equality the model with lowest penalty. For each experiment, we run the algorithm 10 times with different weights initializations. Resulting metrics are averaged over these runs.

We run the experiments with two different window sizes (3 and 6) for the CR2N convolution filter size. We compare the two versions of the architecture: the local and global models described in Section 4 and study three different dynamic pruning strategies: none, dynamic enforced sparsity from epoch=0 and from epoch=30 (arbitrary).

6 Results

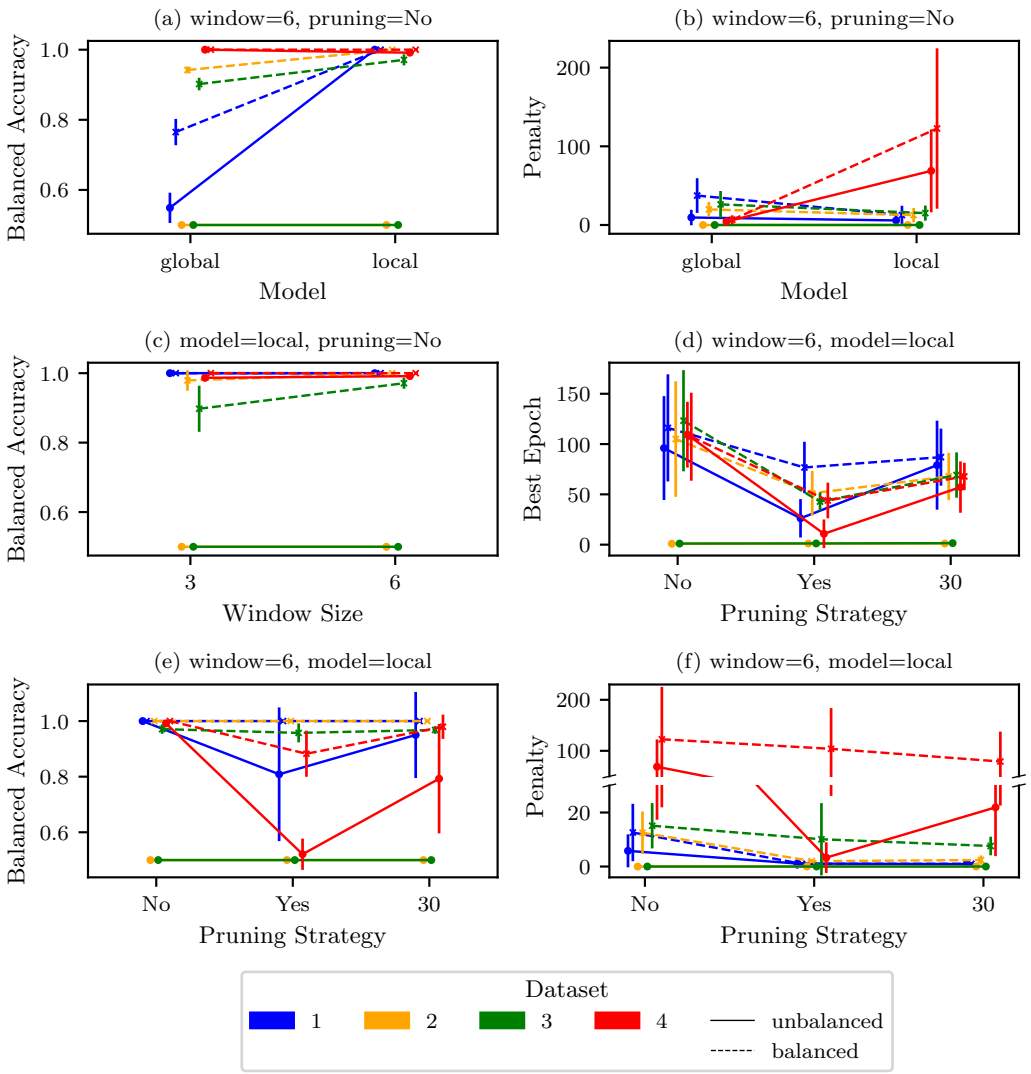

Figure 3: Representations of key results obtained on the synthethic datasets. Error bars represent the standard deviation over the 10 executions with different weights initializations. Full results are available in Appendix F.

Table 2: Performance metrics obtained for the different models, window size and pruning strategy on the peptides dataset, along with the standard deviations over the 10 executions with different weights initializations. (Bal. Acc.: balanced accuracy, Epoch: best epoch).

Model	Window	Pruning	Accuracy	Bal. Acc.	Penalty (Π)	Epoch
global	3	No	88.9 ± 5.0	75.3 ± 12.7	58.7 ± 35.9	105 ± 60
		Yes	85.3 ± 5.3	67.3 ± 14.3	19.4 ± 15.9	23 ± 20
		30	88.7 ± 5.0	75.4 ± 12.9	46.9 ± 28.0	48 ± 22
	6	No	**91.2** ± 1.0	**81.8** ± 2.0	220.0 ± 56.5	92 ± 82
		Yes	89.5 ± 3.6	77.6 ± 9.3	132.7 ± 93.9	33 ± 15
		30	**91.0** ± 1.3	**82.0** ± 2.8	**97.3** ± **60.6**	71 ± 19
local	3	No	90.0 ± 3.7	78.7 ± 9.7	694.3 ± 269.3	77 ± 48
		Yes	**90.3** ± 1.5	**81.6** ± 1.6	**885.2** ± 328.7	25 ± 8
		30	89.2 ± 5.2	76.3 ± 13.2	674.7 ± 483.9	49 ± 13
	6	No	**92.1** ± 1.0	**83.5** ± 2.5	**3.9k** ± **1.4k**	91 ± 58
		Yes	88.0 ± 4.7	74.6 ± 12.4	1.0k ± 1.0k	34 ± 16
		30	**91.7** ± 1.5	**83.5** ± 2.6	1.8k ± 0.6k	49 ± 10

Rule grammar and expressivity The importance of the rule model expressivity can be seen concretely by comparing the different patterns the local and global models have learned for dataset 3b for example: 1. (A or B)-*-(C)-*-*-(A or B or C or D or E or F) in sequence (global, no pruning, window size=6), and 2. (B at t-5 and C at t-3) or (A at t-6 and C at t-4) (local, pruning, window size=6). In the first case, the grammar is not appropriate to model the data (as a reminder, the global model is a constrained version of the local model) as opposed to the local model that learned the perfect rule. In practice, on real data, obtained patterns, such as "if (D or E or G or H or I or N or Q or T or Y)-*-*-*-(D or E or G or I or N or Q or S or T or V or Y) in sequence" obtained for labelling 'invalid-virtual' peptides, can be explored further by a domain expert. Black box approaches do not provide such insights.

It is also highlighted by comparing experiments with local or global model and experiments with different window sizes. First of all, the accuracy of the local model is higher compared to the global model on balanced synthetic datasets 1, 2 and 3 (Fig 3(a)). For balanced and unbalanced dataset 4, both models achieve very high accuracies ($> 95\%$). However, as shown on Fig 3(b), it is at the cost of rule complexity for the local approach with averaged penalty values higher than 60 (and standard deviation higher than 50) compared to lower than 10 for the global model (and standard deviation lower than 5). It points out that the local model in that case requires on average at least more than 6 times more terminal conditions in the learned rule than the global model for comparable accuracies, but also that the weights initial states have a huge impact on the rule complexity when the rule grammar is not expressive enough (with no pruning). Those results are confirmed on real-world dataset with the peptides dataset, accuracies between the local and global models especially for a window size of 6 are comparable. However there is an order of magnitude difference for the penalty, global approach being more concise. It is important to note that by architecture the global approach has less weights to train and thus a much lower maximum penalty.

Datasets 2b and 3b benefit from a bigger window size (highly expected for dataset 3/3b due to ground truth pattern size) as shown in Fig 3(c). Accuracies are also higher with window size 6 than 3 for the peptides dataset at the cost of also higher penalties (Table 2).

The more expressive the model is i.e. the more patterns it can model, training limitations aside, the better for the performance. Of course any black box neural network with no such 1-1 rule mapping constrained architecture would reach 100% accuracy, but it is that mapping in particular that makes the model relevant, expressive and fully interpretable. Also, the best performances in accuracy for the peptides dataset ($\sim 91\%$) are comparable to the best results ($\sim 92\%$) obtained from classification with single kernels when applied to that same dataset in Nwegbu et al. (2022), our model providing an additional fully-interpretable property. The presented model is also flexible due to its logical equivalence and can be inputted into other logical layers for deeper architectures to extend the rule grammar (Beck & Fürnkranz, 2021). It can also be extended for timeseries, temporal aggregates or

multi-classification problems. Other rule grammar extensions can be inspired by Linear Temporal Logic domain and regular expression pattern mining (De Giacomo et al., 2022). However the more expressive the model is the more attention is required for training and rule complexity.

Sparsity and training strategy The importance of the model sparsity is pointed out by the experiments with different pruning strategies. First, looking at training scenarios, both on synthethic and peptide datasets, experiments with sparsity-during-training approaches reach the best model faster on average than without (lower best epoch Fig 3(d)). Then, regarding the performance in terms of accuracy, we can differentiate two cases: balanced and unbalanced datasets. Training of unbalanced datasets is more affected by the aggressive dynamic pruning strategy than balanced datasets with a drop in average of around 0.2 in accuracy for dataset 1 compared to an equivalent accuracy for balanced dataset 1 for example (Fig 3(e)). The pruning strategy starting after 30 epochs is preferred in both cases. Average accuracies with a pruning strategy not starting immediately (30 epochs) are comparable to the ones obtain without pruning for balanced datasets. In terms of rule complexity, penalty values are lower with pruning and even lower when starting after 30 epochs in most cases (Fig 3(f)).

With our pruning strategy (Eq 10), we make the assumption that lower positive loc values are associated to overfitting or redundancy by taking into account that values closer to 0, i.e. on the sigmoid slope, are more likely to shift thus less 'certain'. As pointed out in early work by Prechelt (1997), the dynamic pruning strategy helps to overcome possible lower generalization ability compared to a fixed pruning which could explain cases of better performance (peptide dataset local model window size of 3 for example). Prechelt proposed a different pruning strategy based on a generalization loss to characterize the amount of overfitting. When this strategy is relevant in more general cases and can be applied to many different networks, our strategy is tailored for minimizing positive trainable parameter values.

Sparsity of the model is also induced via the regularization term Π in the loss function $\mathcal{L}$ (Eq 7). While this method is parameterized with a relative importance of sparsity for training optimization and provides an uncontrolled target sparsity, a dynamic pruning strategy is easier to control for both target sparsity and accuracy but is highly dependent on the pruning schedule (Hoefler et al., 2021).

An interesting point is made by Hoefler et al. (2021) about the convolutional operator that 'can be seen as a sparse version of fully-connected layers'. That level of forced sparsity in our model is therefore defined by the fixed window size model parameter with respect to the maximum sequence length. The ideal sparser window size would be the size of the maximum temporal hidden pattern in distribution that can only be approximated with external or expert knowledge and/or tuned with trial and error.

With or without a dynamic pruning strategy, for highly unbalanced datasets (2 and 3), experiments have shown that the training strategy of the model is not suitable. Indeed most of them, label everything with the majority class (50% balanced accuracy). It corresponds to the specific case of learning an empty rule (penalty=0) (Fig 3(a,c,e)). For unbalanced datasets 1 and 4, their best models do not reach on average the same accuracies as in their balanced versions.

Overall this training strategy is both the key and the main limitation of our approach: it can provide a sharp concise rule with minimal redundancy and simplified logical expression but it is highly dependent on numerous model, training and pruning parameters and is not suited as is for highly unbalanced datasets.

7 Conclusion

To conclude, we presented a 1D-convolutional neural architecture to discover local and global patterns in sequential data while learning binary classification rules. This architecture is fully differentiable, interpretable and requires sparsity that is enforced dynamically. One main limitation is its dependence to the window size and sparsity scheduler parameters. Further work will consist in integrating this block into more complex architectures to augment the expressivity of the learned rules as well as extending it for multi-classification.

REFERENCES

Charu C. Aggarwal. On effective classification of strings with wavelets. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pp. 163–172, New York, NY, USA, July 2002. Association for Computing Machinery. ISBN 978-1-58113-567-1. doi: 10.1145/775047.775071.

Florian Beck and Johannes Fürnkranz. Beyond DNF: First Steps towards Deep Rule Learning. In Broňa Brejová, Lucie Ciencialová, Martin Holeňa, František Mráz, Dana Pardubská, Martin Plátek, and Tomáš Vinař (eds.), *Proceedings of the 21st Conference Information Technologies – Applications and Theory (ITAT 2021)*, volume 2962 of *CEUR Workshop Proceedings*, pp. 61–68, Hotel Heľpa, Nízke Tatry and Muránska planina, September 2021. CEUR.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Taylor & Francis, January 1984. ISBN 978-0-412-04841-8.

Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956. Publisher: IEEE.

William W. Cohen. Fast Effective Rule Induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123. Morgan Kaufmann, 1995.

Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean Decision Rules via Column Generation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Giuseppe De Giacomo, Marco Favorito, Jianwen Li, Moshe Y. Vardi, Shengping Xiao, and Shufang Zhu. LTLf Synthesis as AND-OR Graph Search: Knowledge Compilation at Work. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2591–2598, Vienna, Austria, July 2022. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-1-956792-00-3. doi: 10.24963/ijcai.2022/359.

Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2017.

Elias Egho, Dominique Gay, Marc Boulle, Nicolas Voisine, and Fabrice Clerot. A Parameter-Free Approach for Mining Robust Sequential Classification Rules. In *2015 IEEE International Conference on Data Mining*, pp. 745–750, Atlantic City, NJ, November 2015. IEEE. ISBN 978-1-4673-9504-5. doi: 10.1109/ICDM.2015.87.

Lukas Geiger and Plumerai Team. Larq: An open-source library for training binarized neural networks. *Journal of Open Source Software*, 5(45):1746, January 2020. doi: 10.21105/joss.01746.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, March 2010.

Francesca Grisoni, Claudia S. Neuhaus, Miyabi Hishinuma, Gisela Gabernet, Jan A. Hiss, Masaaki Kotera, and Gisbert Schneider. De novo design of anticancer peptides by ensemble artificial neural networks. *Journal of Molecular Modeling*, 25(5):112, April 2019. ISSN 0948-5023. doi: 10.1007/s00894-019-4007-6.

Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Pierre Holat, Marc Plantevit, Chedy Raissi, Nadi Tomeh, Thierry Charnois, and Bruno Cremilleux. Sequence Classification Based on Delta-Free Sequential Patterns. In *2014 IEEE International Conference on Data Mining*, pp. 170–179, Shenzhen, China, December 2014. IEEE. ISBN 978-1-4799-4302-9 978-1-4799-4303-6. doi: 10.1109/ICDM.2014.154.

Stefan Kramer. A Brief History of Learning Symbolic Higher-Level Representations from Data (And a Curious Look Forward). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 4868–4876, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/678.

Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, and Shubham Gupta. Differentiable Rule Induction with Learned Relational Features. *arXiv:2201.06515 [cs, stat]*, January 2022.

Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1675–1684, San Francisco California USA, August 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939874.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*, 2018.

Nnanyelugo Nwegbu, Santosh Tirunagari, and David Windridge. A novel kernel based approach to arbitrary length symbolic data with application to type 2 diabetes risk. *Scientific Reports*, 12: 4985, March 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-08757-1.

Lutz Prechelt. Connection pruning with static and adaptive pruning schedules. *Neurocomputing*, 16 (1):49–61, July 1997. ISSN 0925-2312. doi: 10.1016/S0925-2312(96)00054-9.

Litao Qiao, Weijia Wang, and Bill Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4303–4311, 2021.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical Neural Networks, June 2020.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x.

Guolong Su, Dennis Wei, Kush R. Varshney, and Dmitry M. Malioutov. Interpretable Two-level Boolean Rule Learning for Classification, June 2016.

Cheng Zhou, Boris Cule, and Bart Goethals. Itemset Based Sequence Classification. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Camille Salinesi, Moira C. Norrie, and Óscar Pastor (eds.), *Advanced Information Systems Engineering*, volume 7908, pp. 353–368. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-38708-1 978-3-642-38709-8. doi: 10.1007/978-3-642-40988-2_23.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, November 2017. URL `http://arxiv.org/abs/1710.01878`. arXiv:1710.01878 [cs, stat].

A Context-Free Grammar

Context-free grammar (Chomsky, 1956) A context free-grammar is a 4-tuple $G = (V_T, V_N, S, R)$ where

- V_T, is finite set of terminals or terminal elements in the language that form the alphabet of the language.
- V_N, disjoint from V_T, is a finite set of non terminal elements (variables) that define a sub-language of L. We note $V = V_N \cup V_T$, the vocabulary of the grammar.
- $S \in V_N$, is the start symbol or variable that defines the whole sentence.
- R is a finite set of rules or production rules of the form $A \rightarrow w$ with $A \in V_N$ and $w \in V^*$

In the following, we have 3 different types of terminal elements on the syntax level:

- reserved words, distinguished with the following style : *reserved*
- signs, such as for example $-, *, ...$
- other terminal elements that are defined prior to the grammar, distinguished with the following style: <u>terminal</u>

Production rules presented in the paper (Eq 3, Eq 4, Eq 5 and Eq 6) define grammars when associated to values for V_T, V_N and S.

Here is an example for the base rule model grammar with production rules in Eq 3.

$$\begin{array}{lcl} V_T & = & \{\textit{if}, \textit{then class = 1 else class = 0}, \wedge, \vee, =\} \cup \left\{\underline{x_1}, \underline{x_2}, \dots\right\} \cup \\ & & \left\{\underline{x_{c_1} = \alpha_0^1}, \dots, \underline{x_{c_1} = \alpha_{n_1}^1}, \dots, \underline{x_{c_k} = \alpha_{n_k}^k}\right\} \\ V_N & = & \{\text{rule, base expression, conjunction, predicate, categorical expression,} \\ & & \text{categorical literal, literal}\} \\ S & = & \{\text{rule}\} \end{array} \tag{11}$$

B HARD CONCRETE DISTRIBUTION (LOUIZOS ET AL., 2018)

Parameters are set as follows: $\beta = 2/3$, $\zeta = 1.1$ and $\gamma = -0.1$.

$$u \sim U(0,1), \quad s = \sigma((log(u) - log(1-u) + loc)/\beta), \quad \hat{s} = s * (\zeta - \gamma) + \gamma \tag{12}$$

$$W = \min(\max(\hat{s}, 0), 1) \tag{13}$$

C PEPTIDES DATASET

UCI anticancer peptides dataset (Grisoni et al., 2019) (Available on Dua & Graff (2017)) is composed of one-letter amino acid sequences (of variable length) and each sequence is labeled with its anticancer activity on breast cancer cell lines. The dataset provides 4 classes with the following distribution: 83 inactive-exp, 750 inactive-virtual, 98 moderately active and 18 very active. Sequences lengths range from 5 to 38 letters (Mean: 17 $\pm$ 5.5). We transform this multi-classification problem into a binary classification problem (as done in Nwegbu et al. (2022)). Class 'inactive-virtual' is the positive class (750) and all the other are combined as the negative class (199). No other processing of the data is necessary and we leave it as is.

D SYNTHETIC DATASETS GENERATION

Balanced datasets are generated randomly with the same ground truth as unbalanced datasets. Then, they are upsampled until the minority class represents half of the goal dataset size and appropriate number of majority class are randomly removed.

E EXPERIMENTAL SETTING

The loc-parameters for weights computation are initialized with xavier uniform initialization method (Glorot & Bengio, 2010). The loss function is described in Eq 7 and depends on the MSE loss and regularization coefficient $\lambda = 10^{-5}$. The adam optimizer is used with a fixed learning rate set to 0.1 and a run consists of 200 epochs.

Experiments were run on CPU on a MacBookPro18,2 (2021) with Apple M1 Max chip, 10 Cores, 32 GB of RAM and running macOS Monterey Version 12.4.

F FULL EXPERIMENT RESULTS

Table 3: Performance metrics obtained for the different models, window size and pruning strategy on the synthetic datasets. Values are followed by the standard deviation over the 10 executions with different weights initializations. (Bal Acc: balanced accuracy, Epoch: best epoch).

Dataset	Model	Window	Pruning	Bal. Acc.	Penalty	Epoch
1	global	3	No	50.0 ± 0.0	0.0 ± 0.0	3 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	3 ± 2
			30	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
		6	No	54.9 ± 3.9	9.5 ± 8.1	109 ± 64
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
	local	3	No	100.0 ± 0.0	1.8 ± 1.5	63 ± 24
			Yes	90.0 ± 20.0	0.8 ± 0.4	56 ± 47
			30	85.0 ± 22.9	0.7 ± 0.5	41 ± 27
		6	No	100.0 ± 0.0	5.8 ± 5.6	96 ± 50
			Yes	80.9 ± 23.6	1.1 ± 1.4	26 ± 18
			30	95.0 ± 15.0	0.9 ± 0.3	79 ± 43
1b	global	3	No	70.7 ± 4.0	5.1 ± 1.6	81 ± 27
			Yes	68.5 ± 6.6	5.8 ± 2.8	47 ± 31
			30	71.6 ± 3.5	5.0 ± 1.3	60 ± 16
		6	No	76.5 ± 3.4	37.4 ± 20.3	108 ± 49
			Yes	70.2 ± 4.3	12.5 ± 5.5	29 ± 15
			30	72.7 ± 5.1	15.4 ± 6.9	53 ± 13
	local	3	No	100.0 ± 0.0	3.0 ± 3.1	70 ± 47
			Yes	100.0 ± 0.0	1.0 ± 0.0	49 ± 39
			30	100.0 ± 0.0	1.0 ± 0.0	54 ± 30
		6	No	100.0 ± 0.0	12.6 ± 10.1	116 ± 52
			Yes	100.0 ± 0.0	1.0 ± 0.0	77 ± 24
			30	100.0 ± 0.0	1.0 ± 0.0	87 ± 27

Table 3 continued from previous page

Dataset	Model	Window	Pruning	Bal. Acc.	Penalty	Epoch
2	global	3	No	50.0 ± 0.0	0.0 ± 0.0	3 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	3 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
		6	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
	local	3	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
		6	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
2b	global	3	No	90.5 ± 0.0	3.7 ± 2.1	44 ± 37
			Yes	85.7 ± 9.8	2.3 ± 0.6	39 ± 21
			30	90.5 ± 0.0	2.0 ± 0.0	51 ± 24
		6	No	94.2 ± 0.6	19.8 ± 7.7	89 ± 36
			Yes	90.5 ± 0.0	2.0 ± 0.0	71 ± 16
			30	91.0 ± 1.5	12.4 ± 13.0	61 ± 26
	local	3	No	97.9 ± 2.6	2.8 ± 1.2	58 ± 42
			Yes	97.4 ± 2.6	1.5 ± 0.5	43 ± 28
			30	98.0 ± 2.4	1.6 ± 0.5	54 ± 29
		6	No	100.0 ± 0.0	12.5 ± 7.4	105 ± 56
			Yes	100.0 ± 0.0	2.0 ± 0.0	51 ± 21
			30	100.0 ± 0.0	2.4 ± 1.2	68 ± 22
3	global	3	No	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
		6	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
	local	3	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			Yes	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
		6	No	50.0 ± 0.0	0.0 ± 0.0	1 ± 0
			Yes	50.0 ± 0.0	0.0 ± 0.0	1 ± 1
			30	50.0 ± 0.0	0.0 ± 0.0	2 ± 1
3b	global	3	No	79.8 ± 8.1	4.6 ± 2.2	98 ± 51
			Yes	70.5 ± 8.1	3.1 ± 1.4	48 ± 31
			30	80.4 ± 9.2	2.8 ± 0.6	57 ± 36
		6	No	90.2 ± 1.4	26.2 ± 15.3	120 ± 51
			Yes	77.4 ± 7.4	14.1 ± 8.0	35 ± 13
			30	87.7 ± 3.1	11.5 ± 6.6	61 ± 10
	local	3	No	89.8 ± 6.3	8.5 ± 3.9	95 ± 64
			Yes	86.2 ± 8.0	6.1 ± 8.2	50 ± 34
			30	92.5 ± 4.6	4.2 ± 1.8	64 ± 22
		6	No	97.1 ± 1.2	15.1 ± 7.9	123 ± 49
			Yes	95.7 ± 2.9	10.1 ± 12.9	43 ± 8
			30	96.8 ± 0.9	7.6 ± 2.9	69 ± 21

Table 3 continued from previous page

Dataset	Model	Window	Pruning	Bal. Acc.	Penalty	Epoch
4	global	3	No	100.0 ± 0.0	2.2 ± 0.6	42 ± 28
			Yes	90.0 ± 20.0	1.6 ± 0.8	28 ± 14
			30	95.0 ± 15.0	1.8 ± 0.6	42 ± 25
		6	No	100.0 ± 0.0	4.3 ± 3.4	72 ± 35
			Yes	86.2 ± 21.3	1.7 ± 0.9	33 ± 20
			30	100.0 ± 0.0	2.0 ± 0.0	43 ± 15
	local	3	No	98.7 ± 0.4	28.6 ± 14.5	108 ± 40
			Yes	59.1 ± 12.9	3.4 ± 5.1	30 ± 37
			30	77.1 ± 20.7	12.5 ± 8.7	58 ± 39
		6	No	99.1 ± 0.6	68.8 ± 51.0	109 ± 31
			Yes	52.1 ± 5.2	3.3 ± 5.2	11 ± 13
			30	79.3 ± 19.2	21.9 ± 17.5	57 ± 24
4b	global	3	No	100.0 ± 0.0	2.7 ± 1.1	16 ± 8
			Yes	99.2 ± 2.4	2.1 ± 0.3	29 ± 18
			30	100.0 ± 0.0	2.0 ± 0.0	42 ± 32
		6	No	100.0 ± 0.0	6.7 ± 3.8	90 ± 52
			Yes	99.2 ± 2.4	4.7 ± 8.1	50 ± 22
			30	100.0 ± 0.0	2.2 ± 0.6	69 ± 25
	local	3	No	100.0 ± 0.0	39.9 ± 17.5	76 ± 38
			Yes	99.0 ± 2.0	29.2 ± 9.4	60 ± 14
			30	98.7 ± 4.0	35.9 ± 17.3	53 ± 14
		6	No	100.0 ± 0.0	122.6 ± 100.2	107 ± 42
			Yes	88.2 ± 7.8	103.9 ± 77.3	44 ± 16
			30	98.0 ± 3.9	79.0 ± 55.9	68 ± 12