CAUSAL MODELS OF CONVOLUTIONAL BINARY SPIKING NEURAL NETWORKS FOR EXPLANATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel logic-based causal approach to explaining convolutional binary spiking neural networks (C-BSNNs). Specifically, we provide an SMT (Satisfiability Modulo Theories) encoding of the binary causal model (BCM) underlying a C-BSNN and use it to compute formally correct abductive explanations for its outputs. We demonstrate that our SMT-based causal analysis extends beyond static inputs, preserving the recurrent temporal dependencies of spiking dynamics to support formal explanations of complex event-based DVS-Gesture data alongside static MNIST. To enable training of our networks with Batch Normalization while preserving the Boolean activations of neuronal units required for mapping them to BCMs and subsequent SMT encoding, we introduce a novel Batch Normalization absorption procedure. Finally, we compare our logic-based causal approach with SHAP with respect to the relevance of the elements included in an explanation. While our approach provides a formal guarantee that all such elements are relevant, SHAP provides no such guarantee.

1 INTRODUCTION

The increasing reliance on neural networks for decision making across high-stakes domains has motivated the study of *formal explainability*. Conventional attribution-based methods, such as saliency maps (Simonyan et al., 2014), SHAP (Lundberg & Lee, 2017), and LIME (Ribeiro et al., 2016), provide feature rankings or relevance scores without formal guarantees. Hence they may fail to faithfully reflect the model's output (Adebayo et al., 2018; Hedström et al., 2024) or assign high scores to completely irrelevant features (Marques-Silva & Huang, 2024). On the other hand, formal explainability addresses this challenge through notions like abductive explanations (AXps) which are subset-minimal sets of observed feature values that are sufficient to force a model's outcome (Marques-Silva & Ignatiev, 2022). For neural networks, explanations with formal guarantees can be established through constraint encodings and automated verification (Ignatiev et al., 2019; Malfa et al., 2021; Wu et al., 2023; Bassan & Katz, 2023). However, the real-valued nature of the neural networks considered in these works makes formal verification computationally challenging, due to the need to reason over continuous domains and floating-point computations.

Binarisation provides a promising route toward addressing these challenges. It has primarily been pursued for its energy efficiency, as exemplified by XNOR-Net (Rastegari et al., 2016) and BinaryNet (Hubara et al., 2016), while binary spiking neural networks (BSNNs) additionally offer temporal processing capabilities (Lu & Sengupta, 2020; Kheradpisheh et al., 2022). However, binarisation offers a further, under-explored advantage: Boolean activations and discrete weights make these networks amenable to representation as binary causal models (BCMs). We exploit this observation to map C-BSNNs to BCMs, which we represent logically using SMT (Satisfiability Modulo Theories). The SMT representation of a C-BSNN's binary causal model allows us to compute explanations for the network's outputs directly in terms of input pixels or events. We demonstrate our approach on the static MNIST and temporal DVS-Gesture classification tasks. To accommodate networks trained with Batch Normalisation, we further introduce a Batch Normalization absorption procedure that preserves the network's behaviour while restoring the Boolean structure required for their mapping to binary causal models. The novel contributions of this paper are as follows:

- A mapping of C-BSNNs to BCMs and an SMT encoding of the resulting BCMs that enables formal explanations of C-BSNN outputs (Sections 5).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- A Batch Normalization absorption procedure that enables C-BSNN training with Batch Normalization while preserving the Boolean activations required for their mapping to BCMs and subsequent SMT encoding (Section 4).
- The application of formal explanation to temporal spiking data, demonstrated on the DVS-Gesture dataset (Section 7).
- A comparison of three strategies for explaining C-BSNNs: an end-to-end strategy for finding abductive explanations (subset-minimal sufficient explanations), and more efficient layer-by-layer and n-layer decomposition strategies for finding implicants (sufficient explanations) (Section 6).

2 Related Work

Binary and spiking networks. Binary neural networks restrict weights, activations, or both to discrete values, as in XNOR-Net (Rastegari et al., 2016) and BinaryNet (Hubara et al., 2016). Straight-through estimators (Bengio et al., 2013) address non-differentiable quantisation, with subsequent work improving compact binary architectures (Tang et al., 2017; Qin et al., 2020). Spiking neural networks (SNNs), on the other hand, integrate *spikes* (0 or 1 inputs) over time and fire upon crossing a threshold. Surrogate gradients enable their efficient training with backpropagation through time (Neftci et al., 2019), despite the non-differentiable nature of the spiking mechanism. BSNNs combine spiking dynamics with quantised weights and have been studied under both temporal (Kheradpisheh et al., 2022) and rate (Lu & Sengupta, 2020) coding, including convolutional and stochastic variants (Srinivasan & Roy, 2019; Jang et al., 2021). Whereas these works emphasise efficient neuromorphic inference, Kar et al. (2026) explored the equivalence between BSNNs and symbolic causal models, but only for shallow, fully connected networks trained on static MNIST. We extend this perspective to convolutional architectures and event-based spike streams.

Causal models. Causal models are mathematical objects that describe cause-effect dependencies amongst variables through *structural equations*. They have been studied extensively in AI (Pearl, 2009), logic (Halpern, 2000; 2016), and explainable AI (Miller, 2021; Liu & Lorini, 2023). The binary causal models (BCMs) used in our paper form a special class in which all variables are restricted to Boolean values (Chockler & Halpern, 2004; Aleksandrowicz et al., 2017; Lorini, 2023; de Lima & Lorini, 2024). Their correspondence with Boolean logic makes them naturally amenable to automated causal reasoning using SAT and SMT solvers.

Logic-based explanation of neural networks. Abductive explanations (AXps) identify subset-minimal sets of observed features (Cooper & Marques-Silva, 2023; Ignatiev, 2020) that suffice to fix a model's output. They originate from abductive reasoning (Marquis, 1991) and are also known as prime-implicant explanations (Shih et al., 2018) or sufficient reasons (Darwiche & Hirth, 2020). Their dual, contrastive explanations, identify minimal sets of features whose change can flip the model's output (Ignatiev et al., 2020). Efficient methods for computing AXps exist for linear and monotone classifiers (Marques-Silva et al., 2020; Audemard et al., 2020), random forests (Izza & Marques-Silva, 2021), and boosted trees (Audemard et al., 2023).

For neural networks, the following two approaches are particularly relevant to our work. The first approach, by Shi et al. (2020), compiles binarised networks into ordered binary decision diagrams (OBDDs), providing a functional representation of the network but not explicitly representing the causal structure of its internal computation. Our approach instead explicitly represents neuron-level causal dependencies in the form of a BCM. In a similar spirit, Bertossi & León (2023) use knowledge compilation to transform binary neural networks into Boolean circuits for the efficient computation of SHAP explanation scores. While these approaches provide exact logic-based representations of Boolean networks, their objectives differ from ours. SHAP assigns attribution scores relative to a background distribution, and these scores can vary with uncertainty in this distribution (Cifuentes et al., 2024), potentially leading to misleading feature importance (Marques-Silva & Huang, 2024). In contrast, we compute AXps of the network's output by representing the network as a causal model. Bertossi (2023) provide an expository account of attribution scores and causal counterfactuals as forms of explanation, highlighting the role of logical reasoning in counterfactual analysis. Our work builds on this causal perspective by providing an explicit causal representation of a C-BSNN and using SMT-based formal reasoning to compute abductive explanations of its outputs.

A second line of work searches for formal explanations of real-valued neural networks using constraint-based methods, including MILP (Ignatiev et al., 2019) and SMT-based verification. *VeriX* (Wu et al., 2023) is a verified explainability system for deep neural networks that computes robust explanations, namely minimal sets of input features whose values are sufficient to preserve the network's prediction when the remaining features are perturbed within a bounded region. In the NLP setting, Malfa et al. (2021) similarly seek explanations whose prediction remains invariant under bounded perturbations of the omitted word tokens and their embeddings. Bassan & Katz (2023), on the other hand, relax exact minimality to make the search for explanations more scalable. Nevertheless, verification in this line of work can be affected by discrepancies between continuous, real-valued network models and their finite-precision implementations (Jia & Rinard, 2021). Our approach instead searches directly for explanations over Boolean variables, without imposing a bounded perturbation neighbourhood.

Concept-based explanations. Recent work has also investigated formal explanations at the concept level. Canizales et al. (2026) use a vision-language model such as CLIP to extract semantic concepts from a vision model's representations and compute abductive and contrastive explanations in terms of these concepts. Their causal guarantees are therefore defined over an intermediate concept representation and rely on approximate interventions on this representation. Similarly, Hoarau et al. (2026) compute minimal sufficient explanations over latent concepts at a layer followed by a linear classifier, and project these explanations back to the input space using gradient-based attribution. In contrast, our formal guarantees apply directly to the Boolean variables representing the inputs and neurons of C-BSNNs, without relying on an intermediate concept representation or approximate interventions.

3 Architecture, Quantisation and Learning

In this section, we outline the training and architecture details for networks analysed in our causal approach.

3.1 Learning Datasets

We trained C-BSNNs on two datasets, namely the **MNIST**, consisting of static images, and **DVS-Gesture**, consisting of event streams from dynamic vision sensors (DVSs). For MNIST, we encode the 28×28 grayscale images of handwritten digits from 0 to 9 into Boolean (0 or 1) spikes using either an intensity-based *threshold* encoder that repeats binarised images over T time steps, or a classic *Poisson* rate-code based encoder. Predictions are obtained by using an *arg-max* over the time-averaged output spikes. DVS-Gesture has 11 classes representing hand gestures, with each sample comprising a time series of two-channels ON/OFF event count frames of 128×128 resolution. We threshold these event counts to obtain Boolean (0 or 1) input spikes over time steps compatible with our analysis. For the training we use a standard train, validation, and test split of $70\%, 20\%$ and 10% respectively.

3.2 Weight Quantisation

All convolutional and fully connected layers use *ternary* quantisation, mapping real-valued weights to $\{-1, 0, 1\}$. Specifically, given a real-valued weight tensor W, we compute a threshold from the distribution of the real values,

$$\hat{W}_i = \begin{cases} 1 & \text{if } W_i > \theta, \\ -1 & \text{if } W_i < -\theta, \text{ where, } \theta = \mathbb{E}(|W|) + 0.5\,\mathrm{Std}(|W|) \\ 0 & \text{otherwise.} \end{cases}$$

This is applied inside the forward pass at every training step, with a straight-through estimator (STE) passing gradients in the backward step to the underlying real-valued weights. Spiking neurons are trained with surrogate gradients (*ATan* surrogate) and backpropagation through time, with learnable firing thresholds. This quantisation strategy is useful since it yields sparse ternary filters, with kernels typically containing $60\%-70\%$ zeros, thereby reducing the computation time for explanation search.

3.3 C-BSNN ARCHITECTURES

Table 1 reports the accuracies of the best networks trained. The MNIST classifier is a sparse-ternary C-BSNN without Batch Normalization (Appendix A.1.1), while the DVS-Gesture classifier networks use Batch Normalization after every convolution layer (Appendix A.1.2). For DVS task, we trained the reduced classification models, namely the *single-class* (one-vs-rest) and the *three-class* gesture networks, each with a single convolution-pool block and two channels. We studied variants of them with low-resolution inputs: 32×32 and 16×16 as detailed in Appendix A.1.2, for studying their effect on computational cost.

Table 1: Test accuracies of the C-BSNN models. Here, K is the number of output classes and B the number of convolution and pooling blocks.

Model	K	Input	B	Test
MNIST digit classifier	10	$28^2 \times 1$	2	97.0%
DVS gesture classifier				
Single-class	1	$32^2 \times 2$	1	93.1%
Single-class	1	$16^2 \times 2$	1	90.8%
Three-class	3	$32^2 \times 2$	1	86.4%

Optimisation setup. Training minimises the mean-squared error between average firing-rate outputs and one-hot labels, with ℓ_2 regularisation on non-threshold parameters. We use AdamW with separate parameter groups for learnable thresholds and weights, cosine learning-rate scheduling with warmup, and gradient clipping.

4 BATCH NORMALIZATION ABSORPTION

Batch normalisation (BN) improves the accuracy of some of our networks, particularly those trained on DVS-Gesture. However, the normalisation

$$\mathrm{BN}(x) = \gamma \cdot \frac{x - \mu_{\text{run}}}{\sqrt{\sigma^2_{\text{run}} + \epsilon}} + \beta, \tag{1}$$

with running mean and variance $\mu_{\text{run}}, \sigma^2_{\text{run}}$, learnable scale and shift γ, β, and numerical constant ϵ, acts on the convolution output, i.e. the net synaptic input of each spiking neuron. Consequently, the firing condition effectively involves real-valued weights rather than the ternary weights required by our SMT-based encoding. We instead *absorb* BN into a *dynamic threshold*, leaving the synaptic input unchanged. From a trained BN layer, we compute a per-channel scale and bias, shared across time steps,

$$a = \frac{\gamma}{\sqrt{\sigma^2_{\text{run}} + \epsilon}}, \qquad b = \beta - \frac{\mu_{\text{run}}\,\gamma}{\sqrt{\sigma^2_{\text{run}} + \epsilon}}. \tag{2}$$

With BN, a neuron X with learned threshold τ_X and membrane time constant α receives $ax_k + b$ instead of its net synaptic input x_k at every time step k. Its membrane therefore accumulates the leaked contributions of both terms since its last firing time t' (0 before any firing), and for $a > 0$ its firing condition becomes

$$a\,\mathcal{A}(X,t) + b \sum_{j=0}^{t-t'-1} \left(1 - \tfrac{1}{\alpha}\right)^j \geq \tau_X \iff \mathcal{A}(X,t) \geq \tau_X(t),$$

$$\text{with} \quad \tau_X(t) = \frac{1}{a}\left(\tau_X - b \sum_{j=0}^{t-t'-1} \left(1 - \tfrac{1}{\alpha}\right)^j\right) \quad \text{and} \quad \mathcal{A}(X,t) = \sum_{k=t'+1}^{t} \left(1 - \tfrac{1}{\alpha}\right)^{t-k} x_k. \tag{3}$$

Since $\mathcal{A}(X,t)$ is the membrane potential of the same neuron without BN (Appendix A.2), the bias is absorbed entirely into the threshold. The threshold is reset to τ_X/a after each firing and decreases at each silent step according to the leaked bias, linearly for the integrate-and-fire neuron ($\alpha \to \infty$). Without BN, the threshold remains static, $\tau_X(t) = \tau_X$. The membrane recurrence and reset are otherwise unchanged, i.e., BN is thus removed from the synaptic input while the spikes retain their ternary weights.

5 CAUSAL MODEL OF C-BSNN DYNAMICS: AN SMT ENCODING

We give an SMT (Satisfiability Modulo Theory) representation of the binary causal model (BCM) capturing the C-BSNN spiking dynamics. The formal definition of the C-BSNN spiking dynamics is given in Appendix A.2. In particular, we rely on an SMT based on linear integer arithmetic (LIA) (Barrett et al., 2009).

Let us introduce the notion of binary causal model first. It is binary insofar as it only specifies the necessary and sufficient condition under which a variable is true.

Definition 1 (Binary causal model (BCM)). *A BCM is a triplet $\Gamma = (\mathbf{U}, \mathbf{V}, \mathcal{E})$ where (i) $\mathbf{U}$ is a finite set of* exogenous *variables, whose values are determined outside the model; (ii) $\mathbf{V}$ is a finite set of* endogenous *variables, disjoint from $\mathbf{U}$, whose values are determined within the model; and (iii) $\mathcal{E}$ is a function that associates with each endogenous variable $p \in \mathbf{V}$ a structural equation $\mathcal{E}(p)$ of the form $(p = 1) \leftrightarrow \omega_p$, where ω_p is an SMT formula over linear integer arithmetic whose variables belong to $(\mathbf{U} \cup \mathbf{V}) \setminus \{p\}$.*

The equation $(p = 1) \leftrightarrow \omega_p$ in the previous definition stipulates that p is true exactly when the condition ω_p holds.

A C-BSNN S is mapped onto a BCM $\Gamma_S = (\mathbf{U}_S, \mathbf{V}_S, \mathcal{E}_S)$ as follows. Let $\mathbf{I}, \mathbf{L}$ and $\mathbf{M}$ be the set of input neurons, the set of hidden convolutional neurons, and the set of max-pooling units of the C-BSNN. We assign a Boolean variable $p_{X,t}$ to every neuron $X \in \mathbf{N}$, where $\mathbf{N} = \mathbf{I} \cup \mathbf{L} \cup \mathbf{M}$ at every time step $t \in \{0, \ldots, \mathtt{t}_{end}\}$, where $\mathtt{t}_{end}$ is the last time step at which the network receives incoming spikes. The equations of the causal model reflect that $p_{X,t}$ is true iff X fires at time t. The *exogenous* variables are those associated with the input neurons, $\mathbf{U}_S = \{p_{X,t} : X \in \mathbf{I} \text{ and } 0 \leq t \leq \mathtt{t}_{end}\}$, since their values are fixed by the presented stimulus and not by the network; the *endogenous* variables are those of all remaining neurons, $\mathbf{V}_S = \{p_{X,t} : X \in \mathbf{L} \cup \mathbf{M} \text{ and } 0 \leq t \leq \mathtt{t}_{end}\}$, whose values are determined by the C-BSNN dynamics. The equation of an endogenous variable $p_{X,t}$ captures whether a neuron fires depending on the incoming spikes. Let $\Sigma_{X,t} := \sum_{X' \in \mathcal{R}^+(X)} p_{X',t} - \sum_{X'' \in \mathcal{R}^-(X)} p_{X'',t}$ denote the net synaptic input to X at time t. We have the following equation for a convolutional neuron $X \in \mathbf{L}$ and time $0 \leq t \leq \mathtt{t}_{end}$:

$$\begin{aligned}
&\mathcal{E}_S(p_{X,0}) = (p_{X,0} = 0),\\
&\mathcal{E}_S(p_{X,t}) = \Big((p_{X,t} = 1) \leftrightarrow \big[\,\big(p_{X,t-1} = 0 \rightarrow \Sigma_{X,t} + \mathcal{A}(X, t-1)\big(1 - \tfrac{1}{\alpha}\big) \geq \tau_X(t)\big)\\
&\qquad\qquad \wedge \big(p_{X,t-1} = 1 \rightarrow \Sigma_{X,t} \geq \tau_X(t)\big)\big]\Big). \qquad (4)
\end{aligned}$$

In the previous equation, $\mathcal{R}(X)$ is the set of parent neurons of X in the C-BSNN architecture. If $X' \in \mathcal{R}(X)$ then $\mathcal{W}(X, X')$ is the weight of the connection between X' (the parent neuron) and X (the successor neuron). As pointed out in Section 3.2 we consider C-BSNNs with ternary weights, so $\mathcal{W}(X, X') \in \{-1, 0, 1\}$. $\mathcal{R}^+(X) = \{X' \in \mathcal{R}(X) : \mathcal{W}(X, X') = 1\}$ and $\mathcal{R}^-(X) = \{X' \in \mathcal{R}(X) : \mathcal{W}(X, X') = -1\}$ are, respectively, the set of parent neurons of X with excitatory connections to X and inhibitory connections to X. The component $\mathcal{A}(X, t-1)$ is the membrane potential of the spiking neuron X at the previous time step $t-1$, and $\tau_X(t)$ is its firing threshold at time t. See Appendix A.2 for the formal model of the C-BSNN spiking dynamics in which all components are precisely defined. We note that a finite time constant $\alpha > 1$ models Leaky-Integrate-and-Fire (LIF) spiking neuron, while the leak-free limit $\alpha \to \infty$ corresponds to an Integrate-and-Fire (IF) neuron. Thus, the causal model is independent of the underlying spiking neuron model, as activation, leak and threshold are computed externally and treated as external parameters of the causal model. Equation 4 captures the membrane-potential recurrence and reset dynamics of the spiking neuron through the general leak parameter α. When the neuron does not spike at the previous time step ($p_{X,t-1} = 0$), its membrane potential from the previous state contributes to the net incoming spikes, with the leak determined by α. If the neuron spiked at the previous time step ($p_{X,t-1} = 1$), the membrane is reset, and the firing condition depends only on the net incoming spikes at the current time step.

For a max-pool unit $X \in \mathbf{M}$, and time $0 \leq t \leq \mathtt{t}_{end}$, we have the following equation:

$$\mathcal{E}_S(p_{X,t}) = \Big((p_{X,t} = 1) \leftrightarrow \bigvee_{X' \in \mathcal{T}(X)} (p_{X',t} = 1)\Big).$$

In the previous equation $\mathcal{T}(X)$ is the set of parent neurons of the max-pool unit X. Notice, unlike convolutional neurons, connections to max-pool units have no weights.

6 Explanation Strategies

Using the SMT representation presented in Section 5, we compute abductive explanations (AXps), i.e., minimally sufficient sets of input features that determine the network's output. The explanation (the *explanans*) fixes input features to their observed values, while the full network output to be explained constitutes the *explanandum*. Specifically, the explanandum and the explanans are formally represented as terms, with a term being a conjunction of SMT theory atoms from the network's BCM, as defined in Section 5, in which each variable occurs at most once, and a theory atom being of the form $p_{X,t} = 0$ or $p_{X,t} = 1$. Terms are denoted by $\lambda, \lambda', \ldots$ We use a deletion-based search: starting from the complete set of observed input feature values, we systematically remove elements from the candidate solution whenever the remaining set is still sufficient to force the explanandum. The formal definition of AXp, in terms of its sufficiency and minimality requirement, along with the AXp search algorithm are given in Appendix A.3. The AXp search is computationally expensive, primarily due to the large resolution of the input, the depth of the network, and the number of output heads we need to fix. This is reflected in the large number of equations in the network's BCM.

To address this scalability challenge, we study iterative explanation search strategies that involve partitioning the end-to-end causal model into sequential, layer-wise sub-models. While computationally more tractable, these decompositions introduce a trade-off regarding the minimality of the resulting explanations. The strategy, called the *n-layer decomposition* search strategy, merges several consecutive layers of the network into bundles of a fixed size and explains the last layer in a bundle using only neurons of the first layer in the bundle. Specifically, suppose that the network has k layers, with the k-th layer being the output layer, and n is the size of the bundles, with $k \bmod n = 0$. At step 1, we search for an abductive explanation of the network's full output in terms of the neurons at layer $k - n$. After finding an abductive explanation at step 1, we search at step 2 for an abductive explanation of the explanation found at step 1 in terms of the neurons at layer $k - 2n$. The procedure continues in this way until the input layer is reached and used to explain the explanation found at step k/n. At each step of the procedure, we only need to use a strict subset of the equations in the binary causal model of the network, namely those capturing the dependencies between the layers in the bundle involved at that step. Thus, instead of reasoning over the entire end-to-end causal model, each abductive explanation is computed over a smaller sub-model, thereby providing a computational advantage. The *layer-by-layer decomposition* search strategy is the special case with $n = 1$: it consists in computing abductive explanations step by step, proceeding backward from the output to the input layer, one layer at a time.

Interestingly, the n-layer decomposition strategy guarantees that the resulting explanation of the network's observed output, expressed in terms of the input neurons, satisfies the sufficiency requirement for an abductive explanation. Thus, the resulting explanation is guaranteed to be an implicant, although it is not necessarily minimal and therefore not necessarily an abductive explanation.

Theorem 1. *Let λ_0 be the term representing the observed output of the C-BSNN S to be explained, and let λ be a term representing the explanation of λ_0 obtained through the application of the n-layer decomposition search strategy, for any n dividing k. Then, λ is sufficient for λ_0 relative to the binary causal model Γ_S of S.*

The proof of this theorem is given in Appendix A.4.1 in the Supplementary Material.

7 Experiments and Results

We compare the two decomposition search strategies introduced in Section 6, n-layer and layer-by-layer against our end-to-end AXp search strategy, on MNIST and DVS-Gesture. In all our experiments, we compute explanations at a selected time step $t^\star$, i.e., λ and λ_0 contain input and output atoms, respectively, at $t^\star$. The causal analysis also applies when λ and λ_0 contain atoms from multiple time steps, although this increases the computation time. λ_{init} denotes the initial implicant in the search algorithm presented in Appendix A.3.

We use Z3 for SMT verification and PyTorch for network training.

7.1 MNIST CLASSIFICATION TASK

Table 2 compares the strategies over the $784 = 28 \times 28$ MNIST input pixels. *Retained* denotes the fraction of input atoms corresponding to input pixels that are kept in the explanation, i.e. $|\lambda|/|\lambda_{init}|$. *Present retained* and *absent retained* denote the fractions of present ($p_{X,t} = 1$) and absent ($p_{X,t} = 0$) input atoms that are kept from the initial implicant λ_{init}, respectively. *Per-call* denotes the mean solver time required to test whether an input atom can be removed from the explanation. Figure 1 visualises the retained present and absent input pixels for an instance of digit 1.

Table 2: MNIST pixel-level explanation search with $|\lambda_{init}| = 28 \times 28 = 784$ and $t^\star = 1$.

	Explanation size			Search cost	
Search Strategy	Retained ($\|\lambda\|/\|\lambda_{init}\|$)	Present Retained	Absent Retained	Per-Call Time	Total Time
End-to-end	**45.2%**	**86.0%**	**40.1%**	12.5 m	6.79 d
n-layer	56.6%	91.9%	52.3%	1.7 s	**15.4 h**
layer-by-layer	57.5%	100%	52.3%	**0.9 s**	23.8 h

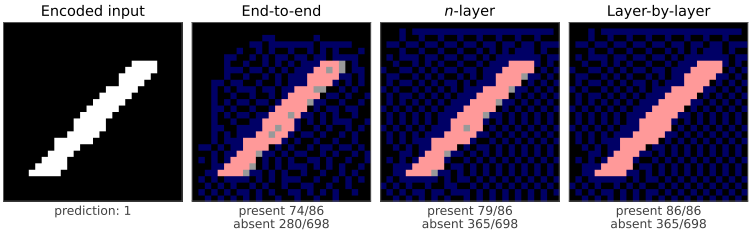

Figure 1: The encoded input (*left*) of the MNIST sample in Table 2, followed by the explanations found by end-to-end, n-layer and layer-by-layer search, each overlaid on the encoded input. Red and blue mark the retained *present* and *absent* pixels, respectively. The numbers of retained present and absent pixels are given below each panel.

The end-to-end strategy yields the smallest explanation but requires substantially more computation. The decomposition strategies reduce computation time at the cost of retaining more pixels. Although layer-by-layer search has lower per-call cost, it requires more solver invocations across layers, resulting in a higher overall search time than the n-layer strategy.

7.2 DVS-GESTURE CLASSIFICATION TASK

The DVS-Gesture task involves both spatial and temporal processing. Table 3 compares the networks described in Section 3.3 at the input-event level.

Table 3: Search for explanation on reduced DVS-Gesture models at the input spikes (events) level ($|\lambda_{init}| = 32 \times 32 \times 2 = 2{,}048$, $t^\star = 1$)

Search Strategy	Retained ($\|\lambda\|/\|\lambda_{init}\|$)	Present Retained	Absent Retained	Per-Call Time	Total Time
Single-class classifier ($K = 1$)					
End-to-end	**15.2%**	24.0%	**14.1%**	0.58 s	19.9 m
n-layer	16.7%	20.7%	16.1%	0.18 s	**6.6 m**
layer-by-layer	16.7%	20.7%	16.1%	**0.14 s**	34.9 m
Three-class classifier ($K = 3$)					
n-layer	**76.0%**	**63.2%**	**77.5%**	0.35 s	**12.8 m**
layer-by-layer	76.9%	70.5%	77.6%	**0.20** s	55.5 m

For the single-class classifier, end-to-end search again retains the fewest input atoms in the resulting explanation, while n-layer decomposition search is fastest overall despite the lower per-call solver times of layer-by-layer decomposition search, similar to the MNIST results. End-to-end results are omitted for the three-class classifier due to its substantially higher computation time. The bottom

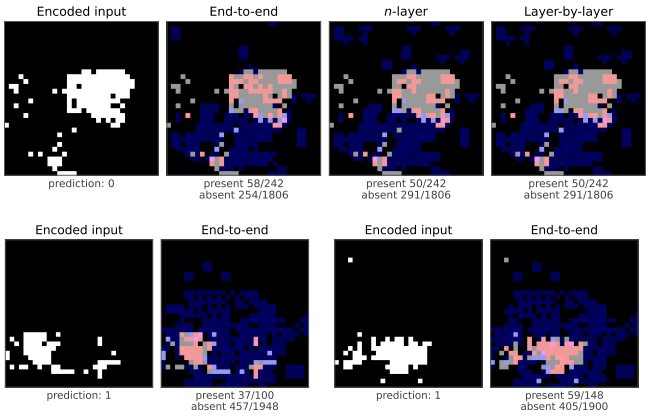

Figure 2: Explanations of outputs from the single-class gesture classifier at $t^\star = 1$. Similar to Figure 1, red marks *present* events, while blue marks the retained *absent* events in the DVS frames. The explanations are overlaid on the frame at $t^\star$ for the following gestures: left hand wave (top), forearm roll backward (bottom left), and guitar (bottom right).

panels of Figure 2 show explanations for two misclassified instances, both *false positives*. In these cases, our approach provides a certified set of input events that provably forces the erroneous firing of the output neuron.

7.3 Discussion: explanation search cost

Table 4 summarises the results from the end-to-end searches at $t^\star = 1$ for some of the trained networks across different input resolutions. The results illustrate how the computational cost of explanation search varies with the resolution and architecture of the underlying network.

Table 4: AXp found from the end-to-end search strategy at $t^\star = 1$. Retained fractions range over the evaluated samples; per-call and wall times are those of the fastest run. K and B are as in Table 1.

						Best run	
Network	K	Input	B	$\lvert\lambda_{\text{init}}\rvert$	$\lvert\lambda\rvert/\lvert\lambda_{\text{init}}\rvert$	Per-call	Wall time
DVS single-class	1	$16^2\times2$	1	512	13–16%	0.04 s	22.5 s
DVS single-class	1	$32^2\times2$	1	2,048	15–24%	0.58 s	19.9 m
MNIST	10	28^2	2	784	45.2%	12.5 m	6.79 d

For the DVS-Gesture models in the first two rows of Table 4, increasing the input resolution increases the SMT encoding from 5,634 to 22,530 variables and the search time from seconds to minutes, while the retained fraction remains comparable. MNIST is substantially more expensive despite its simpler classification task, reflecting its deeper two-block architecture, larger encoding (94,884 variables), and the need to fix all $K = 10$ output heads.

8 Comparison with SHAP

We compare our explanations with Kernel SHAP on MNIST. Kernel SHAP estimates feature attribution scores by running the network on perturbed versions of the input defined by different feature coalitions and using the resulting outputs to estimate each feature's contribution. The sampling size determines the number of such perturbed inputs, while the SHAP kernel assigns a weight to each coalition and its corresponding output. Unlike our approach, SHAP is model-agnostic, assigns relevance scores rather than identifying sufficient feature sets, and provides no formal guarantees (Huang & Marques-Silva, 2024; Letoffe et al., 2024; Huang & Marques-Silva, 2023a;b). Here, the features are input-level pixels. We investigate whether SHAP assigns high relevance scores to

disconnected pixels by thresholding its scores and measuring how many selected pixels are disconnected from the output.

Moreover, we present the following theorem, which guarantees that pixels mentioned in explanations computed from our SMT-based causal approach have non-zero connection weights with successor neurons. This stands in contrast to SHAP, for which there is no such formal guarantee.

Theorem 2. *Let λ_0 be the term representing the observed output of the C-BSNN S to be explained, and let λ be a term representing the explanation of λ_0 obtained through any of the two search strategies described in Section 6 (end-to-end and n-layer decomposition). Then, for every input neuron $X \in \mathbf{I}$ of S and theory atom $p_{X,t} = 0$ or $p_{X,t} = 1$ occurring in λ, there exists a convolutional neuron $Y \in \mathbf{L}$ which is connected to X in the network (i.e., $X \in \mathcal{R}^+(Y) \cup \mathcal{R}^-(Y)$).*

The proof is provided in Appendix A.4.2.

We identify 46 pixels that are *disconnected* from the predicted output head: every path from such a pixel to the corresponding output neuron passes through at least one zero-valued weight. These pixels therefore cannot influence the output and are irrelevant by construction. Theorem 2 guarantees that they never appear in an AXp. We use a relevance threshold $\delta = 8 \cdot 10^{-3}$ to convert SHAP attribution scores into a feature set, including a pixel if the absolute value of its SHAP score is at least δ. Table 5 reports how often Kernel SHAP selects disconnected pixels as relevant over 25 independent draws at each sampling size.

Table 5: Disconnected pixels selected as relevant by Kernel SHAP at $\delta = 8 \cdot 10^{-3}$, over 25 draws per sampling size. The last column gives the largest fraction of the 46 disconnected pixels selected in a single draw.

Sample Size	Time per Draw	Draws selecting a disconnected pixel	Maximum fraction of disconnected pixels selected
10^3	5.3 s	25 of 25	84.8%
10^4	67 s	25 of 25	52.2%
10^5	6.0 m	4 of 25	2.2%

At 10^3 and 10^4 samples, every draw selects at least one disconnected pixel as relevant. At 10^5 samples, this occurs in only 4 of 25 draws, while the maximum fraction of the 46 disconnected pixels selected in a single draw decreases from 84.8% to 2.2%. Thus, increasing the sampling size substantially reduces both the frequency and the extent of such spurious selections, but does not eliminate them.

9 Conclusion

In this paper, we presented an SMT-based encoding that maps convolutional binary spiking neural networks (C-BSNNs) with ternary weights $(-1, 0, 1)$ to binary causal models (BCMs), providing an exact representation of the causal dependencies between spiking neurons, with or without leak. We exploit this causal structure to compute abductive explanations (AXps) of network outputs. The Batch Normalization absorption introduced in Section 4 further extends our causal analysis to networks trained with batch normalization. However, exact AXp search can become computationally expensive for large input spaces and deeper architectures. We therefore introduced layer-by-layer and n-layer decomposition strategies that reduce the computational cost by reasoning over smaller causal sub-models, providing faster sufficient explanations at the cost of global minimality.

While the present study focuses on abductive explanations (AXps), the causal representation developed here opens the way to a broader family of formal explanations. In particular, we aim to investigate contrastive and counterfactual explanations, as well as notions of actual cause (Halpern & Pearl, 2005; de Lima & Lorini, 2024; Halpern, 2015) and NESS (Necessary Element of a Sufficient Set) cause (Wright, 1988; Beckers, 2021; Halpern, 2008). The same causal structure could also serve as a basis for deriving higher-level, concept-based explanations, by abstracting over groups of causally related neural variables. Beyond visual tasks, applying this framework to language-based BSNNs (Bal & Sengupta, 2024) and multimodal spiking models would further test its ability to provide formal explanations across different modalities.

AI USE STATEMENT

We used generative AI assistants for certain tasks in the preparation of this work. LLMs were used for rephrasing sentences for readability and checking grammar. No section was generated completely from a prompt, and every sentence in the submitted text has been read, revised and approved by the authors. We used an LLM coding assistant while implementing, refactoring and debugging the SMT encoding for C-BSNN, the AXp search, and the plotting and analysis utilities released with this paper. All such code was reviewed by the authors, and none of it was produced solely using these coding assistants. We used an LLM assistant to suggest certain candidate related work. Every suggestion was then located, read and assessed by the authors before being cited. None of the references in this paper was included solely based on the basis of an AI-generated summary.

We did not use generative AI for research ideation or for the technical content of the paper. The formalisation of C-BSNNs as binary causal models, the SMT encoding, the Batch Normalization absorption procedure, the two decomposition search strategies, and the statements and proofs of Theorems 1 and 2 have been formulated by the authors. Generative AI was not used to produce, select or alter any experimental result. Every number reported in Section 7 and Section 8 comes from the experiments described. Furthermore, the Z3 solver provides mathematical guarantees that are not generated by LLM agents.

We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, its claims and the artifacts released with it.

ETHICS STATEMENT

Both datasets are public benchmarks. MNIST contains no personal data, and DVS-Gesture was recorded by its authors from consenting volunteers. A dynamic vision sensor records brightness changes rather than appearance, and we further downsample its frames to binary event maps, so the inputs we analyse carry no identifying information. We collected no new data and involved no human subjects.

REPRODUCIBILITY STATEMENT

The code for SMT encoding, AXp search, along with the network training scripts, is included in the supplementary material, with a README that maps each table and figure to the exact command that produces it. The causal analysis is reproducible with the code provided in the zip folder. Since AXps are not necessarily unique, multiple subset-minimal explanations may exist for a given instance. To ensure deterministic and reproducible results, we fix the literal-deletion order in the AXp search, eliminating literals with observed value 0 before those with observed value 1. However, any other order can equivalently produce a subset-minimal AXp. Both datasets are public: MNIST is downloaded by the supplied scripts, and DVS-Gesture is obtained from the link provided in the README, which also contains the event-count thresholding and downsampling that we apply.

REFERENCES

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pp. 9525–9536, 2018.

G. Aleksandrowicz, H. Chockler, J. Y. Halpern, and A. Ivrii. The computational complexity of structure-based causality. *Journal of Artificial Intelligence Research*, 58:431–451, 2017.

G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, pp. 838–849, 2020.

G. Audemard, S. Bellart, J.-M. Lagniez, and P. Marquis. Computing abductive explanations for boosted regression trees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pp. 3432–3441. ijcai.org, 2023.

M. Bal and A. Sengupta. SpikingBERT: Distilling BERT to train spiking language models using implicit differentiation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*, pp. 10998–11006. AAAI Press, 2024.

C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pp. 825–885. IOS Press, 2009.

S. Bassan and G. Katz. Towards formal XAI: Formally approximate minimal explanations of neural networks. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2023)*, volume 13993 of *Lecture Notes in Computer Science*, pp. 187–207. Springer, 2023. doi: 10.1007/978-3-031-30823-9_10.

S. Beckers. The counterfactual NESS definition of causation. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, pp. 6210–6217. AAAI Press, 2021.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

L. Bertossi. Attribution-scores and causal counterfactuals as explanations in artificial intelligence. In *Reasoning Web. Causality, Explanations and Declarative Knowledge (RW 2022 Tutorial Lectures)*, volume 13759 of *Lecture Notes in Computer Science*, pp. 1–23. Springer, 2023.

L. Bertossi and J. E. León. Efficient computation of Shap explanation scores for neural network classifiers via knowledge compilation. In *Logics in Artificial Intelligence (JELIA 2023)*, volume 14281 of *Lecture Notes in Computer Science*, pp. 49–64. Springer, 2023.

R. Canizales, D. Gopinath, C. Păsăreanu, and R. Mangal. Concept-based abductive and contrastive explanations for behaviors of vision models. *CoRR*, abs/2605.06640, 2026.

H. Chockler and J. Y. Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004.

S. Cifuentes, L. Bertossi, N. Pardal, S. Abriola, M. Vanina Martinez, and M. Romero. The distributional uncertainty of the SHAP score in explainable machine learning. In *Proc. 27th European Conference on Artificial Intelligence (ECAI 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 971–978. IOS Press, 2024. doi: 10.3233/FAIA240586.

M. C. Cooper and J. Marques-Silva. Tractability of explaining classifier decisions. *Artificial Intelligence*, 316:103841, 2023.

A. Darwiche and A. Hirth. On the reasons behind decisions. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, pp. 712–720. IOS Press, 2020.

T. de Lima and E. Lorini. Model checking causality. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024)*. ijcai.org, 2024.

J. Y. Halpern. Axiomatizing causal reasoning. *Journal of Artificial Intelligence Research*, 12:317–337, 2000.

J. Y. Halpern. Defaults and normality in causal structures. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, pp. 198–208. AAAI Press, 2008.

J. Y. Halpern. A modification of the Halpern-Pearl definition of causality. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 3022–3033. AAAI Press, 2015.

J. Y. Halpern. *Actual Causality*. MIT Press, 2016.

J. Y. Halpern and J. Pearl. Causes and explanations: a structural-model approach. Part I: Causes. *British Journal for the Philosophy of Science*, 56(4):843–887, 2005.

A. Hedström, L. Weber, S. Lapuschkin, and M. M.-C. Höhne. A fresh look at sanity checks for saliency maps. In *Explainable Artificial Intelligence (xAI 2024)*, Communications in Computer and Information Science, pp. 403–420, Cham, 2024. Springer Nature Switzerland.

A. Hoarau, C. Zhu, and V. L. Nguyen. OPTIMUS-Prime: Minimal and sufficient concept explanations for deep vision models. *CoRR*, abs/2606.07180, 2026.

X. Huang and J. Marques-Silva. A refutation of Shapley values for explainability. *CoRR*, abs/2309.03041, 2023a.

X. Huang and J. Marques-Silva. Refutation of Shapley values for XAI – additional evidence. *CoRR*, abs/2310.00416, 2023b.

X. Huang and J. Marques-Silva. On the failings of Shapley values for explainability. *International Journal of Approximate Reasoning*, 171:109112, 2024.

I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems 29 (NeurIPS 2016)*, pp. 4107–4115. Curran Associates, Inc., 2016.

A. Ignatiev. Towards trustable explainable AI. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI 2020)*, pp. 5154–5158. ijcai.org, 2020.

A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pp. 1511–1519. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33011511.

A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. From contrastive to abductive explanations and back again. In *Proceedings of the XIXth International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020)*, Lecture Notes in Computer Science. Springer, 2020.

Y. Izza and J. Marques-Silva. On explaining random forests with SAT. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pp. 2584–2591, 2021.

H. Jang, N. Skatchkovsky, and O. Simeone. BiSNN: Training spiking neural networks with binary weights via bayesian learning. In *2021 IEEE Data Science and Learning Workshop (DSLW)*, pp. 1–6. IEEE, 2021.

K. Jia and M. Rinard. Exploiting verified neural networks via floating point numerical error. In *Static Analysis: 28th International Symposium (SAS 2021)*, volume 12913 of *Lecture Notes in Computer Science*, pp. 191–205. Springer, 2021.

A. Kar, E. Lorini, and T. Masquelier. Binary spiking neural networks as causal models. *Logics for New-Generation AI*, pp. 51, 2026.

S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier. BS4NN: Binarized spiking neural networks with temporal coding and learning. *Neural Processing Letters*, 54(2):1255–1273, 2022. doi: 10.1007/s11063-021-10680-x.

O. Letoffe, X. Huang, N. Asher, and J. Marques-Silva. From SHAP scores to feature importance scores. *CoRR*, abs/2405.11766, 2024.

X. Liu and E. Lorini. A unified logical framework for explanations in classifier systems. *Journal of Logic and Computation*, 33(2):485–515, 2023.

E. Lorini. A rule-based modal view of causal reasoning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pp. 3286–3295, 2023.

S. Lu and A. Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 2020. doi: 10.3389/fnins.2020.00535.

S. M. Lundberg and S. I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 4765–4774. Curran Associates, Inc., 2017.

E. La Malfa, A. Zbrzezny, R. Michelmore, N. Paoletti, and M. Kwiatkowska. On guaranteed optimal robust explanations for NLP models. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pp. 2658–2665, 2021.

J. Marques-Silva and X. Huang. Explainability is Not a game. *Communications of the ACM*, 67(7): 66–75, 2024. doi: 10.1145/3635301.

J. Marques-Silva and A. Ignatiev. Delivering trustworthy AI through formal XAI. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, pp. 12342–12350. AAAI Press, 2022. doi: 10.1609/aaai.v36i11.21499.

J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explaining Naive Bayes and other linear classifiers with polynomial time and delay. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

P. Marquis. Extending abduction from propositional to first-order logic. In *Proceedings of the International Workshop on Fundamentals of Artificial Intelligence Research (FAIR'91)*, Lecture Notes in Computer Science, pp. 141–155. Springer, 1991.

T. Miller. Contrastive explanation: a structural-model approach. *The Knowledge Engineering Review*, 36, 2021.

E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *Computer Vision – ECCV 2016*, pp. 525–542, Cham, 2016. Springer International Publishing.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*, pp. 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778.

W. Shi, A. Shih, A. Darwiche, and A. Choi. On tractable representations of binary neural networks. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, pp. 882–892, 2020.

A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining Bayesian network classifiers. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pp. 5103–5111, 2018.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at the 2nd International Conference on Learning Representations (ICLR 2014)*, 2014.

G. Srinivasan and K. Roy. ReStoCNet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing. *Frontiers in Neuroscience*, 13, 2019. doi: 10.3389/fnins.2019.00189.

W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 2625–2631. AAAI Press, 2017.

R. W. Wright. Causation, responsibility, risk, probability, naked statistics, and proof: Pruning the bramble bush by clarifying the concepts. *Iowa Law Review*, 73:1001–1077, 1988.

M. Wu, H. Wu, and C. Barrett. VeriX: Towards verified explainability of deep neural networks. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pp. 22247–22268, 2023.

A Appendix

A.1 C-BSNN Architecture Details

The network architecture and layer structures of the models described in Section 3 are given below; their accuracies are reported in Table 1.

A.1.1 MNIST C-BSNN (without Batch Normalization)

The MNIST classifier is a sparse-ternary CNN without Batch Normalization and with Integrate-and-Fire (IF) neurons:

$$\begin{aligned}&\text{Input } (1 \times 28 \times 28) \to \text{Conv}(1 \to 32, 3 \times 3, \text{pad} = 1) \to \\ &\text{IF} \to \text{MaxPool}(2) \\ &\to \text{Conv}(32 \to 64, 3 \times 3, \text{pad} = 1) \to \text{IF} \to \text{MaxPool}(2) \\ &\to \text{FC}(64{\cdot}7{\cdot}7 \to 10) \to \text{IF}.\end{aligned}$$

A.1.2 DVS-Gesture C-BSNN classifiers (with Batch Normalization)

For the DVS-Gesture dataset, we trained classifiers on spatially downsampled $32 \times 32 \times 2$ event frames using a single convolutional block with Batch Normalization:

$$\begin{aligned}&\text{Input } (T \times 2 \times 32 \times 32) \\ &\to [\text{Conv}(2 \to 8,\ 3 \times 3,\ \text{pad} = 1) \to \text{BN} \to \text{LIF} \to \text{Pool}] \\ &\to \text{FC}(8{\cdot}16{\cdot}16 \to K) \to \text{LIF}.\end{aligned}$$

with $K = 1$ for the *single-class* (one-vs-rest) classifier and $K = 3$ for the *three-class* model restricted to the gesture classes $\{0, 1, 2\}$. Both classifiers use the same training pipeline. To study the effect of input resolution on the computational cost, we additionally trained a $16 \times 16 \times 2$ version of the single-class classifier, identical in all other respects. Leaky-Integrate-and-Fire (LIF) neurons were used to exploit a forgetting mechanism on the event frames across time steps.

A.2 Formal Model of C-BSNN

In this section, we present the formal model of a convolutional BSNN (C-BSNN) with ternary weights. For simplicity, we refer to it simply as a convolutional BSNN, since the use of ternary weights is assumed throughout our analysis.

In our notation, we denote the spatial dimension of the input of a convolution layer by d, the dimension of its kernel by m, the padding and stride dimensions by p and s, along with the input and output channel dimensions denoted by $\mathsf{c}_{in}, \mathsf{c}_{out}$ respectively. Notice that d, m, p, s, $\mathsf{c}_{in}, \mathsf{c}_{out}$ are integer dimensions. We write m_M, p_M and s_M for the kernel, padding and stride dimensions of the pooling operation; these play the same role as m, p and s do for convolution, and generally take different values.

A convolution operation with these parameters produces an output of spatial side length $\mathsf{d}' = \left\lfloor \frac{\mathsf{d}-\mathsf{m}+2\mathsf{p}}{\mathsf{s}} \right\rfloor + 1$. Every neuron is identified by a tuple (x, y, c, ℓ) denoting its spatial position (x, y), its channel c, and a tag $\ell \in \{\mathtt{I}, \mathtt{L}, \mathtt{M}\}$ signifying the layer type it belongs to. However for notational convenience, we simply use X to denote a generic neuron of any type: input, convolutional or max-pool.

Before stating the definition, we describe the different neuron types in our network. First, $\mathbf{I}$ is the set of *input neurons*, representing the spatial array of Boolean spike-encoded input pixels as presented to the network at each time step. Secondly, $\mathbf{L}$ is the set of *hidden convolutional neurons*, each of which is either an integrate-and-fire (**IF**) or a leaky **IF** unit for the MNIST and DVS Gesture case respectively with its own firing threshold $\tau_X(t)$. Thirdly, $\mathbf{M}$ represents the set of *max-pooling units* which are stateless, threshold-free units that subsample the convolutional layer channel by channel, reporting whether at least one neuron in its $\mathsf{m}_\mathsf{M} \times \mathsf{m}_\mathsf{M}$ pooling window fired.

A deep C-BSNN generally stacks several such convolution-pooling blocks. The final fully connected classification layer is the degenerate convolution whose receptive field is the entire preceding layer

($\mathsf{m} = \mathsf{d}$, $\mathsf{p} = 0$, $\mathsf{s} = 1$); we write $\mathbf{C} \subseteq \mathbf{L}$ for its neurons, the *classification (output) neurons*, one per class, which have no successors. For readability, we state the definition for a single block; the multi-block case iterates it, with block-specific values of these parameters.

Definition 2 (C-BSNN architecture)**.** *The architecture of a convolutional BSNN (C-BSNN) is a tuple* $S = \big(\mathbf{I}, \mathbf{L}, \mathbf{M}, \mathcal{R}, \mathcal{W}, \mathcal{T}, (\tau_X(t))_{X\in\mathbf{L},\, t\in\mathbb{N}}\big)$ *where:*

- $\mathbf{I} = \big\{(x, y, c, \mathtt{I}) : 0 \le x, y < \mathsf{d},\ 1 \le c \le \mathsf{c}_{in}\big\}$ *is the set of input neurons;*
- $\mathbf{L} = \big\{(x, y, c, \mathtt{L}) : 0 \le x, y < \mathsf{d}',\ 1 \le c \le \mathsf{c}_{out}\big\}$*, with* $\mathsf{d}' = \big\lfloor \frac{\mathsf{d}-\mathsf{m}+2\mathsf{p}}{\mathsf{s}} \big\rfloor + 1$*, is the set of hidden convolutional neurons;*
- $\mathbf{M} = \big\{(x, y, c, \mathtt{M}) : 0 \le x, y < \mathsf{d}'',\ 1 \le c \le \mathsf{c}_{out}\big\}$*, with* $\mathsf{d}'' = \big\lfloor \frac{\mathsf{d}'-\mathsf{m}_\mathsf{M}+2\mathsf{p}_\mathsf{M}}{\mathsf{s}_\mathsf{M}} \big\rfloor + 1$*, is the set of max-pooling units, one per* $\mathsf{m}_\mathsf{M} \times \mathsf{m}_\mathsf{M}$ *window of* $\mathbf{L}$ *and per output channel;*
- $\mathcal{R} \subseteq \mathbf{L} \times \mathbf{N}$*, with* $\mathbf{N} = \mathbf{I} \cup \mathbf{L} \cup \mathbf{M}$*, is the receptive-field relation such that, for every* $(x, y, c, \mathtt{L}) \in \mathbf{L}$*,*

$$\mathcal{R}(x, y, c, \mathtt{L}) = \big\{(x', y', c', \ell) \in \mathbf{N} : \ell \in \{\mathtt{I}, \mathtt{L}, \mathtt{M}\}, x' = \mathsf{s}x + k - \mathsf{p}, y' = \mathsf{s}y + h - \mathsf{p}, \\ 0 \le k, h < \mathsf{m}, 1 \le c' \le \mathsf{c}_{in}\big\}; \tag{5}$$

- $\mathcal{W} : \mathcal{R} \longrightarrow \{-1, 0, 1\}$ *is the weighting function, subject to weight sharing, that is, for all* $(x, y, c, \mathtt{L}), (x', y', c, \mathtt{L}) \in \mathbf{L}$ *sharing the same output channel* c*, for all* $0 \le k, h < \mathsf{m}$*, for all* $1 \le c' \le \mathsf{c}_{in}$*, and for all* $\ell \in \{\mathtt{I}, \mathtt{L}, \mathtt{M}\}$*,*

$$\mathcal{W}\big((x, y, c, \mathtt{L}), (\mathsf{s}x + k - \mathsf{p}, \mathsf{s}y + h - \mathsf{p}, c', \ell)\big) = \\ \mathcal{W}\big((x', y', c, \mathtt{L}), (\mathsf{s}x' + k - \mathsf{p}, \mathsf{s}y' + h - \mathsf{p}, c', \ell)\big); \tag{6}$$

- $\mathcal{T} \subseteq \mathbf{M} \times \mathbf{L}$ *is the pooling-support relation such that, for every* $(x, y, c, \mathtt{M}) \in \mathbf{M}$*,*

$$\mathcal{T}(x, y, c, \mathtt{M}) = \big\{(x', y', c, \mathtt{L}) \in \mathbf{L} : x' = \mathsf{s}_\mathsf{M}x + k - \mathsf{p}_\mathsf{M}, y' = \mathsf{s}_\mathsf{M}y + h - \mathsf{p}_\mathsf{M}, \\ 0 \le k, h < \mathsf{m}_\mathsf{M}\big\}; \tag{7}$$

- $\tau_X(t) \in \mathbb{R}$ *is the firing threshold of the convolutional neuron* $X \in \mathbf{L}$ *at time* t*. It is static,* $\tau_X(t) = \tau_X$ *for all* t*, with* τ_X *being the learned threshold, for networks trained without Batch Normalization, while for networks with Batch Normalization absorbed as introduced in Section 4, it evolves according to Equation 3.*

Notice that two properties are satisfied in a convolutional architecture: a local receptive field and shared weights. This local connectivity is reflected in the definition of $\mathcal{R}$ in Equation 5, as each neuron is only connected to parent neurons within the $\mathsf{m} \times \mathsf{m}$ kernel window across all channels. The shared-weight property defined in Equation 6 forces all neurons of a channel c to apply the same connection weights $\mathcal{W}$ at every spatial position. On the other hand, max-pool units have no weights associated with their connections. They perform a max operation over the parent neurons of each channel connected to them through Equation 7.

Given the architecture of a convolutional BSNN, we define BSNN-compatible spiking dynamics.

Definition 3 (C-BSNN-compatible dynamics)**.** *Let* $S = \big(\mathbf{I}, \mathbf{L}, \mathbf{M}, \mathcal{R}, \mathcal{W}, \mathcal{T}, (\tau_X(t))_{X\in\mathbf{L}}\big)$ *be a C-BSNN architecture and* $\Delta = (F, P)$ *with* $F = (\mathcal{F}_X)_{X\in\mathbf{I}\cup\mathbf{L}}$ *a family of firing functions* $\mathcal{F}_X : \mathbb{N} \to \{0, 1\}$*, and* $P = (\mathcal{P}_X)_{X\in\mathbf{M}}$ *a family of max-pool functions* $\mathcal{P}_X : \mathbb{N} \to \{0, 1\}$*. We say* Δ *is* S*-compatible up to time* t_{end} *iff for every* $X \in \mathbf{L}$ *and* $t \le \mathsf{t}_{end}$*,*

$$\mathcal{F}_X(t) = \begin{cases} 0, & t = 0, \\ \Theta\big(\mathcal{A}(X, t) - \tau_X(t)\big), & t > 0, \end{cases} \tag{8}$$

where

$$\mathcal{A}(X, t) = \begin{cases} 0, & t = 0, \\ \mathcal{A}(X, t-1)\,\big(1 - \mathcal{F}_X(t-1)\big)\big(1 - \frac{1}{\alpha}\big) \\ \quad + \sum_{(X,X')\in\mathcal{R}} \mathcal{W}(X, X')\,\mathcal{F}_{X'}(t), & t > 0, \end{cases}$$

$\Theta(x) = 1$ *if* $x \geq 0$ *and* 0 *otherwise; and for every* $X \in \mathbf{M}$ *and* $t \leq \mathtt{t}_{end}$,

$$\mathcal{P}_X(t) = \max_{X' \in \mathbf{L}:\ (X,X') \in \mathcal{T}} \mathcal{F}_{X'}(t). \tag{9}$$

Some explanations of Definition 3 are in order. The quantity $\mathcal{A}(X,t)$ is the *membrane potential* of neuron X at time t: it accumulates, through the recursive first summand, whatever potential survived from the previous step and, through the second summand, the weighted Boolean spikes currently arriving from the receptive field of X. The surviving potential is governed by two factors. The factor $(1 - \mathcal{F}_X(t-1))$ implements the *hard reset*: if X fired at $t-1$, its potential is wiped to zero before the new inputs are integrated. The factor $(1 - \frac{1}{\alpha})$ implements the *leak* of the LIF neuron with time constant $\alpha > 1$: at every silent step a fraction $\frac{1}{\alpha}$ of the potential is lost. The IF neurons used for MNIST correspond to the leak-free limit $\alpha \to \infty$, for which this factor becomes 1. The firing function $\mathcal{F}_X$ then simply applies the Heaviside step Θ to the difference between the membrane potential and the firing threshold $\tau_X(t)$: the neuron emits a Boolean spike exactly when its potential reaches the threshold. Two further points deserve emphasis. First, the network is required to be completely silent at $t = 0$; this grounds the recursion and matches the reset state of the trained models. Second, the condition constrains $\mathcal{F}_X$ only for non-input neurons $X \in \mathbf{L}$: the firing of the input neurons is not determined by the network but by the spike-encoded stimulus presented to it, which is precisely why the input neurons will play the role of *exogenous* variables in the causal model introduced in Section 5.

A.3 Computing Explanations

In this section, we show how to use a binary causal model (BCM) defined in Definition 1 for formalising and computing explanations in the context of the C-BSNN architectures we trained for the MNIST and DVS Gesture classification tasks. Following the literature on abductive explanation (AXp), we define it to be a prime implicant that is actually true. Moreover, we define it in relation to a binary causal model. For simplicity, we assume an AXp (the *explanans*) is a term made of exogenous variables and the property to be explained (the *explanandum*) is a Boolean expression made of endogenous ones. This assumption is perfectly compatible with our application to the MNIST as well as DVS-Gesture classification task in which we want to explain the network's full output on the basis of the pixel-level features. Nonetheless, this assumption could be dropped without consequence; we would only need to assume that the explanans and the explanandum involve different variables.

Some preliminary notions are needed before defining AXp formally. We define a *term* to be a conjunction of SMT theory atoms in which a variable can occur at most once, a theory atom being of the form $p = 0$ or $p = 1$. Terms are denoted by $\lambda, \lambda', \ldots$ Given two terms λ, λ', with a slight abuse of notation, we write $\lambda' \subseteq \lambda$ (resp. $\lambda' \subset \lambda$) to mean that the set of theory atoms appearing in λ' is a subset (resp. strict subset) of the set of theory atoms appearing in λ. Given a BCM, $\Gamma = \big(\mathbf{U}, \mathbf{V}, \mathcal{E}\big)$ as defined in Definition 1 and an arbitrary set of variables $\mathbf{X} \subseteq \mathbf{U} \cup \mathbf{V}$, $Term_{\mathbf{X}}$ denotes the set of terms built from $\mathbf{X}$.

Definition 4 (Abductive explanation)**.** *Let* S *be a C-BSNN architecture,* $\Gamma_S = \big(\mathbf{U}_S, \mathbf{V}_S, \mathcal{E}_S\big)$ *the BCM induced by it,* $\mathcal{I}_{\mathbf{U}_S} : \mathbf{U}_S \longrightarrow \{0,1\}$ *a Boolean interpretation for the exogenous variables in* $\mathbf{U}_S$*,* $\lambda \in Term_{\mathbf{U}_S}$ *and* λ_0 *a term built from* $\mathbf{V}_S$*. We say that* λ *is an abductive explanation (AXp) of* λ_0 *with respect to* Γ_S *and* $\mathcal{I}_{\mathbf{U}_S}$ *if and only if the following three conditions are satisfied:*

$$i)\ \forall \tau \in \{0,1\}, \forall p = \tau \subseteq \lambda, \mathcal{I}_{\mathbf{U}_S}(p) = \tau,$$

$$ii) \models \Big(\bigwedge_{p \in \mathbf{V}_S} \mathcal{E}_S(p) \wedge \lambda \wedge \mathsf{hyp}_S \Big) \rightarrow \lambda_0,$$

$$iii)\ \forall \lambda' \subset \lambda, \not\models \Big(\bigwedge_{p \in \mathbf{V}_S} \mathcal{E}_S(p) \wedge \lambda' \wedge \mathsf{hyp}_S \Big) \rightarrow \lambda_0,$$

where the hypothesis

$$\mathsf{hyp}_S =_{def} \bigwedge_{p_{X,t} \in \mathbf{U}_S \cup \mathbf{V}_S} \big(p_{X,t} = 1 \vee p_{X,t} = 0\big)$$

captures the assumption that every variable takes a value in $\{0,1\}$*, reflecting the spiking nature of the neuron, and where* $\models \varphi$ *means that the SMT formula* φ *is valid, i.e., true under every assignment of integer values to its variables.*

Let us illustrate how Definition 4 applies to a ternary-weight C-BSNN S trained on a classification task. Given an input sequence $input : \{0, \ldots, \mathtt{t}_{end}\} \times \mathbf{I} \longrightarrow \{0, 1\}$ and an observed output sequence $out : \{0, \ldots, \mathtt{t}_{end}\} \times \mathbf{C} \longrightarrow \{0, 1\}$ for this input, where $\mathbf{C}$ is the set of classification (output) neurons introduced in Appendix A.2, we aim to abductively explain the output at a chosen time $t \in \{0, \ldots, \mathtt{t}_{end}\}$ using only variables for the input at time t. More precisely, we take the *explanandum* (i.e., λ_0) to be the term

$$\mathsf{out}_{S,t} =_{def} \bigwedge_{\mathtt{c}_z \in \mathbf{C}:\ out(t,\mathtt{c}_z)=1} p_{\mathtt{c}_z,t} = 1 \wedge \bigwedge_{\mathtt{c}_z \in \mathbf{C}:\ out(t,\mathtt{c}_z)=0} p_{\mathtt{c}_z,t} = 0. \tag{10}$$

It represents the observed output of the network at time t. Then, we search for an abductive explanation $\lambda \in Term^t_{\mathbf{U}_S}$ of $\mathsf{out}_{S,t}$ with respect to Γ_S and to the Boolean interpretation $\mathcal{I}_{\mathbf{U}_S}$ encoding the input sequence $input$ (i.e., $\mathcal{I}_{\mathbf{U}_S}(p_{X,t}) = input(t, X)$ for every $t \in \{0, \ldots, \mathtt{t}_{end}\}$ and $X \in \mathbf{I}$). The latter condition guarantees that the resulting explanation of the network's output at time t represents a portion of the actual input presented to the network at t.

To compute an abductive explanation, we rely on a standard abductive explanation search algorithm, whose pseudo code is presented in Algorithm 1. The algorithm is initialized with a complete term λ_{init} over the set of exogenous variables (i.e., $\mathbf{U}_S$), which fully represents the actual input at the selected time t. Then, literals are systematically removed from λ_{init}, and at each iteration, we check whether condition ii) in Definition 4 is still satisfied.

Algorithm 1 Computing Abductive Explanation

Input: Initial implicant λ_{init} and explanandum λ_0 that satisfy conditions i) and ii) in Def. 4
Output: Abductive explanation λ
Set $\lambda = \lambda_{init}$
for $l \in \lambda$ **do**
  if $\models \big(\bigwedge_{p \in \mathbf{V}_S} \mathcal{E}_S(p) \wedge \lambda \setminus l \wedge \mathsf{hyp}_S\big) \rightarrow \lambda_0$ **then**
    $\lambda \leftarrow \lambda \setminus l$
  end if
end for
return λ

At the end of the search algorithm we further verify the validity of condition iii) in Definition 4 for a *prime implicant* check of the resulting abductive explanation λ. Algorithm 1 issues exactly $|\mathbf{U}_S|$ validity queries, so the number of solver calls is linear in the number of input neurons. Each call is nonetheless a validity check whose cost grows with the size of the encoding, and it is this per-call cost, rather than the number of calls, that dominates in practice (Table 4).

A.4 Proofs

A.4.1 Proof of Theorem 1

Proof. λ_0 is the given term that represents the observed output to be explained. At each step of the n-layer decomposition search (Section 6), for $i = 1, \ldots, r$, we define the explanation λ_i which is obtained by explaining λ_{i-1} over the i-th bundle $V^{(i)}$. The final explanation $\lambda_r = \lambda$ is defined over the input neurons. We note that $r = \frac{k}{n}$. By condition (ii) of Definition 4, for every i,

$$\models \left(\bigwedge_{p \in V^{(i)}} \mathcal{E}_S(p) \wedge \lambda_i \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_{i-1}. \tag{11}$$

We prove by induction on $j = 1, \ldots, r$ that the explanations obtained after the first j bundles are sufficient for λ_0, i.e.,

$$\models \left(\bigwedge_{i=1}^{j} \bigwedge_{p \in V^{(i)}} \mathcal{E}_S(p) \wedge \lambda_j \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_0. \tag{12}$$

For $j = 1$, equation 12 is exactly equation 11 for $i = 1$.

Now suppose equation 12 holds for some $j < r$. By equation 11 for $i = j + 1$,

$$\models \left(\bigwedge_{p \in V^{(j+1)}} \mathcal{E}_S(p) \wedge \lambda_{j+1} \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_j.$$

Combining this implication with the induction hypothesis equation 12 by transitivity of logical implication yields

$$\models \left(\bigwedge_{i=1}^{j+1} \bigwedge_{p \in V^{(i)}} \mathcal{E}_S(p) \wedge \lambda_{j+1} \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_0, \tag{13}$$

which proves the statement for $j + 1$.

For $j = r$, since the bundles form a partition of $\mathbf{V}_S$ and $\lambda_r = \lambda$, we obtain

$$\models \left(\bigwedge_{p \in \mathbf{V}_S} \mathcal{E}_S(p) \wedge \lambda \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_0. \tag{14}$$

Thus, the final input-level explanation λ is sufficient for λ_0.

Minimality, however, is not preserved: each step removes literals that are redundant within its own bundle, whereas a literal retained at one step may become redundant when all bundles are considered jointly. The strategy therefore guarantees sufficiency, but not subset-minimality. □

A.4.2 Proof of Theorem 2

Proof. Suppose, for contradiction, that i) $\lambda = (p_{X,t} = \tau) \wedge \lambda'$, with $X \in \mathbf{I}$ and $\tau \in \{0, 1\}$, is an abductive explanation of λ_0 with respect to Γ_S and $\mathcal{I}_{\mathbf{U}_S}$, and ii) there is no $Y \in \mathbf{L}$ such that $X \in \mathcal{R}^+(Y) \cup \mathcal{R}^-(Y)$. By the construction of the equations, ii) implies that iii) for every $q \in \mathbf{V}_S$, the variable $p_{X,t}$ does not occur in $\mathcal{E}_S(q)$. Moreover, as $p_{X,t} \in \mathbf{U}_S$ and λ is a term, iv) $p_{X,t}$ does not occur in λ' and λ_0.

Thus, by the sufficiency of λ in i), λ' is also a sufficient explanation for λ_0, i.e.,

$$\models \left(\bigwedge_{q \in \mathbf{V}_S} \mathcal{E}_S(q) \wedge \lambda' \wedge \mathsf{hyp}_S \right) \rightarrow \lambda_0.$$

However, $\lambda' \subset \lambda$, contradicting the minimality condition of Definition 4. Therefore, there exists $Y \in \mathbf{L}$ such that $X \in \mathcal{R}^+(Y) \cup \mathcal{R}^-(Y)$. □