Formal Verification of Long Short-Term Memory based Audio Classifiers: A Star based Approach

Neelanjana Pal
Institute for Software Integrated Systems
Vanderbilt University
Nashville, USA
neelanjana.pal@vanderbilt.edu

Taylor T Johnson
Institute for Software Integrated Systems
Vanderbilt University
Nashville, USA
taylor.johnson@vanderbilt.edu

Formally verifying audio classification systems is essential to ensure accurate signal classification across real-world applications like surveillance, automotive voice commands, and multimedia content management, preventing potential errors with serious consequences. Drawing from recent research, this study advances the utilization of star-set-based formal verification, extended through reachability analysis, tailored explicitly for Long Short-Term Memory architectures and their Convolutional variations within the audio classification domain. By conceptualizing the classification process as a sequence of set operations, the star set-based reachability approach streamlines the exploration of potential operational states attainable by the system. The paper serves as an encompassing case study, validating and verifying sequence audio classification analytics within real-world contexts. It accentuates the necessity for robustness verification to ensure precise and dependable predictions, particularly in light of the impact of noise on the accuracy of output classifications.

1 Introduction

Deep Neural Networks (DNNs) have demonstrated remarkable capabilities in addressing intricate tasks like image classification, object detection, speech recognition, natural language processing, and document analysis, at times even surpassing human performance [21,23,24]. This success has ignited a surge in exploring the viability of DNNs across diverse real-world domains, including biometric authentication, mobile facial recognition for security, and malware detection. However, given the sensitive nature of the data in these critical applications, incorporating safety, security, and robust verification into their design has become paramount.

However, studies have revealed that even slight modifications in input data can effectively mislead cutting-edge, well-trained networks, causing inaccuracies in their predictions [12, 32, 40]. The arena of network verification has primarily concentrated on image inputs, particularly emphasizing the assurance of safety and robustness in various classification neural networks [2, 7, 19, 31, 43, 44]. Previous investigations have scrutinized a range of network architectures, encompassing feed-forward neural networks (FFNNs [42]), convolutional neural networks (CNNs [44]), semantic segmentation networks (SSNs [43]), and a few using Recurrent Neural Networks (RNNs [41]) employing diverse set-based reachability tools such as Neural Network Verification (NNV [26,45]) and JuliaReach [6], among others.

Models utilizing NNs for audio classification have found application in diverse tasks, ranging from Music Genre Classification [8,10,11] and Environmental Sound Classification [4,9,13] to Audio Generation [33,36]. Therefore, formal verification of audio classification systems holds paramount importance in ensuring their reliability and safety, particularly in safety-critical applications such as autonomous vehicles [35,46], medical diagnosis [15,30], and industrial monitoring [47].

This study introduces an extension, building upon the foundations laid by two recent studies [34,41] in the domain of formal verification. The objective is to leverage set-based reachability techniques to

M. Farrell, M. Luckcuck, M. Schwammberger, and
M. Gleirscher (Eds): Fifth International Workshop on
Formal Methods for Autonomous Systems (FMAS 2023)
EPTCS 395, 2023, pp. 162–179, doi:10.4204/EPTCS.395.12

verify audio classification models based on the Long Short Term Memory (LSTM) and CNN-LSTM architectures. Drawing inspiration from [41], which highlights the star-based verification of basic vanilla RNNs, and from [34], which demonstrates the formal verification of convolutional neural networks operating on time series data, work shown in this paper amalgamates both concepts. Specifically, it employs two LSTM models and one CNN-LSTM model for these classifications, following the ones depicted in [27–29].

Contributions.

1. This paper presents a thorough case study on the formal verification of audio classification models using the LSTM and CNN-LSTM architectures with two different datasets. Our focus is to rigorously assess the robustness verification of these models within a formal verification framework, analyzing their behavior and performance against input noises. We develop our work as an extension of the NNV tool[1] to formally analyze and explore CNN-LSTM architecture verification for audio data using sound and deterministic reachability methods.
2. Building on insights from existing research [34,41], this paper extends formal verification to more complex RNN architectures. This involves addressing the challenges of the complex structure of the LSTM layers, comprehensively evaluating their behavior, and ensuring robustness compliance through formal verification. This study pushes formal verification's boundaries, embracing design complexities for heightened assurance and reliability.
3. In this assessment, we conduct a thorough and comprehensive evaluation of three distinct network architectures across diverse audio classification scenarios.
4. Finally, we develop insights on evaluating the reachability analysis on those networks and possible future direction.

Outline. The paper is organized as follows: Section 2 mentions the works already done in the literature and the inspiration works for this paper; Section 3 provides the necessary context for the background, and defines the verification properties for this work; Section 4 explains the reachability calculations for the LSTM layer; and Section 5 describes the methodology, including dataset, network models, and input perturbations. Section 6 presents the experimental results, evaluation metrics, and their implications. Finally, Section 7 summarizes the main findings and suggests future research directions.

2 Related Work

In recent times, an upsurge of methodologies and tools have arisen to confront the verification complexities inherent in intricate systems like Deep Neural Networks (DNNs), as evident from the literature [14,17,25,44]. Correspondingly, tools have emerged to tackle the robustness challenges of Convolutional Neural Networks (CNNs) [2,19,20,37–39]. Earlier undertakings in the verification of Recurrent Neural Networks (RNNs) are showcased through projects like RnnVerify [18] and RNSVerify [1]. RNSVerify employs an unrolling technique to translate RNNs into extensive Feedforward Neural Networks (FFNNs), simplifying verification through Mixed-Integer Linear Program (MILP) approaches [1]. However, this unrolling method faces scalability constraints, particularly with bounded n-step RNNs, as

[1] The code for this paper is available at https://github.com/verivital/nnv/tree/master/code/nnv/examples/Submission/FMAS2023

the verification complexity scales dramatically. Conversely, RnnVerify [18] employs invariant inference for RNN verification, bypassing unrolling. Their strategy involves crafting an FFNN with matching dimensions to over-approximate the RNN, followed by verifying RNN properties over this approximation using SMT-based methodologies. Our work gets inspiration from [41], where authors introduce a pioneering approach founded on star reachability for RNN verification, aiming to amplify the dependability and safety of RNNs and show the results based on some vanilla RNN models.

Distinction from the previous works [34,41]. While both papers share the common goal of validating the robustness of RNNs, the preceding study can be perceived as an initial step in that research trajectory. In contrast, this paper represents a more comprehensive evolution of the concepts initially introduced.

1. The work in [41] focused on the Vanilla RNN, while this paper delves into models of significantly greater intricacy, such as the LSTM and CNN-LSTM architectures. Vanilla RNNs and LSTMs are both types of recurrent neural networks. However, Vanilla RNNs are simpler in terms of architecture and have fewer parameters, whereas LSTMs are more complex due to their gated units and larger parameters.
    (a) Vanilla RNNs handle input sequences in a sequential manner, updating a hidden state at each step. In contrast, LSTMs also maintain a hidden state, but their structure is more complex, featuring multiple gates (input, forget, and output gates) that regulate the information flow.
    (b) Vanilla RNNs face challenges capturing long-term dependencies within sequences due to the vanishing gradient problem. This problem limits their ability to learn connections between distant time steps. LSTMs, on the other hand, were specifically designed to tackle the vanishing gradient problem and excel at capturing long-term dependencies, rendering them better suited for tasks involving intricate temporal relationships.
    (c) Vanilla RNNs possess a constrained memory capacity, often rapidly discarding information from earlier time steps. This limitation can hinder their performance in tasks with extended sequences. In contrast, LSTMs feature an improved memory mechanism that allows them to retain or discard information from prior time steps selectively. This capability equips them to handle longer sequences and capture complex patterns effectively.
2. The study conducted in [34] focused on examining time-series regression models in the Prognostics and Health Management domain. Drawing inspiration from this work, our study extends the investigation to encompass the time domain's influence, specifically concerning sequential audio noise and Japanese vowel audio samples. This basic experiment provides a foundation for understanding the robustness and reliability of audio classification systems. They offer insights that can be directly applied to real-world scenarios, making them valuable for a broad audience in the field of audio classification.
    (a) Utilizing real-world datasets, this experiment can yield practical insights into audio classification system performance, benefiting fields such as speech recognition, audio surveillance, and multimedia content management by offering real-world applicability.
    (b) This experiment can provide valuable insights into the robustness of audio classifiers when exposed to different noise levels and perturbations, offering crucial implications for applications where audio data is frequently affected by a noise like voice commands in automobiles or audio analysis in noisy settings.
    (c) While this work concentrates on two particular datasets, the verification methodologies showcased can be extended to diverse audio classification endeavors, allowing readers engaged in

various audio classification challenges to customize and apply the methodologies to their specific contexts.

(d) The metrics used in this paper can be potential for real-world applications to evaluate and enhance the reliability and efficiency of audio classification systems.

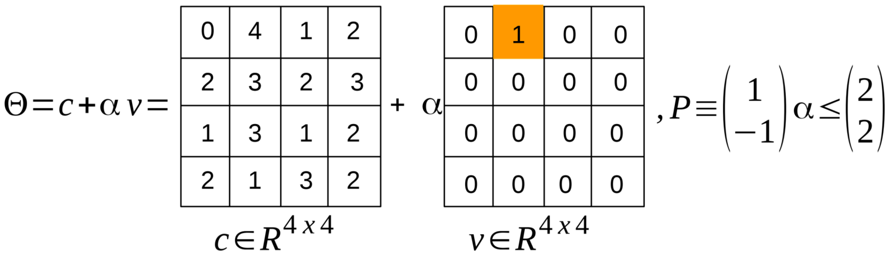
3 Preliminaries

This section introduces some basic definitions and descriptions necessary to understand the progression of this paper and the necessary evaluations on audio classification models.

3.1 Neural Network Verification Tool and Star Sets

The Neural Network Verification (NNV) tool constitutes a framework designed to verify the safety and robustness of neural networks [26,45]. This tool meticulously scrutinizes neural network behavior across diverse input conditions, warranting secure and accurate functionality across all scenarios. NNV employs reachability algorithms, including the exact and over-approximate star set methodologies [42, 44], to compute reachable sets for each neural network layer. These sets encapsulate all feasible network states for a given input, thereby facilitating the verification of specific safety properties.

NNV holds particular significance in safety-critical domains like autonomous vehicles and medical devices, where the trustworthiness and reliability of neural networks are paramount. NNV bolsters public confidence in these applications by ensuring consistent performance across all conditions. In this paper, we extend the capabilities of the NNV tool to implement our work, utilizing the star-based reachability analysis to ascertain the reachable sets of neural networks at their outputs.



$$\Theta = c + \alpha v = \underbrace{\begin{bmatrix} 0 & 4 & 1 & 2 \\ 2 & 3 & 2 & 3 \\ 1 & 3 & 1 & 2 \\ 2 & 1 & 3 & 2 \end{bmatrix}}_{c \in R^{4\,x\,4}} + \alpha \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{v \in R^{4\,x\,4}}, P \equiv \begin{pmatrix} 1 \\ -1 \end{pmatrix} \alpha \leq \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Figure 1: Star for a sequence input data with four Feature Values (rows) with four time-steps (columns)

Definition 3.1 **A generalized star set** *(or simply star)* Θ *is a tuple* $\langle c, V, P \rangle$ *where* $c \in \mathbb{R}^n$ *is the center,* $V = \{v_1, v_2, \cdots, v_m\}$ *is a set of m vectors in* $\mathbb{R}^n$ *called basis vectors, and* $P : \mathbb{R}^m \to \{\top, \bot\}$ *is a predicate. The basis vectors are arranged to form the star's n × m basis matrix. The set of states represented by the star is given as:*

$$[\![\Theta]\!] = \{x \mid x = c + \Sigma_{i=1}^{m}(\alpha_i v_i) \text{ and } P(\alpha_1, \cdots, \alpha_m) = \top\}. \tag{1}$$

In this work, we restrict the predicates to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ *where, for p linear constraints,* $C \in \mathbb{R}^{p \times m}$*,* α *is the vector of m-variables, i.e.,* $\alpha = [\alpha_1, \cdots, \alpha_m]^T$*, and* $d \in \mathbb{R}^{p \times 1}$*.*

3.2 Network Architecture Specifics

3.2.1 Long Short Term Memory (LSTM) Layer

An LSTM layer, a subtype of the Recurrent Neural Network (RNN) layer, excels at capturing long-term dependencies in time series and sequential data [16]. It comprises two critical elements: the hidden state (h_t, also called the output state) and the cell state (c_t). At each time step 't,' the hidden state captures the layer's output for that instance, while the cell state accumulates insights from preceding time steps.

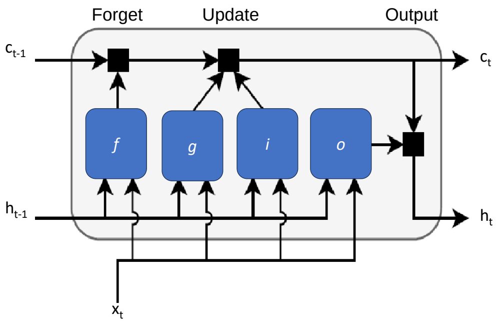


Figure 2: The flow of data at time step t in an LSTM layer

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \sigma_c(c_t) \end{aligned} \tag{2}$$

During each time step, the layer refines the cell state by incorporating or omitting information. This process is steered by distinct gates that control these adjustments, as shown in Fig. 2.

$$\begin{aligned} i_t &= \sigma_g(W_i x_t + R_i h_{t-1} + b_i) \\ f_t &= \sigma_g(W_f x_t + R_f h_{t-1} + b_f) \\ g_t &= \sigma_c(W_g x_t + R_g h_{t-1} + b_g) \\ o_t &= \sigma_g(W_o x_t + R_o h_{t-1} + b_o) \end{aligned} \tag{3}$$

In these equations, $\odot$ represents the Hadamard product (element-wise multiplication), σ_c denotes the activation function applied element-wise to the cell state c_t and to the cell state gate g_t; σ_g denotes the activation function applied element-wise to the hidden state gates. Here, W, R, and b are, respectively, hidden state weights, recurrent weights, and biases for each of the gates.

3.2.2 Convolutional Neural Network + Long Short Term Memory (CNN+LSTM) Architecture

When processing sequences, a CNN uses sliding convolutional filters over the input, extracting information from spatial and temporal dimensions. Conversely, an LSTM network progresses through time steps, capturing lasting connections between them. The synergy of CNN and LSTM layers, as seen in CNN+LSTM architectures [49], harnesses the strengths of both convolutional and LSTM units for insightful data analysis.

The convolutional component forms the foundation for acquiring local feature modules that grasp both local and hierarchical correlations. This fusion enables the identification of intricate data relationships. Additionally, the inclusion of an LSTM layer enhances the network's capacity to capture prolonged dependencies by leveraging information from these localized features.

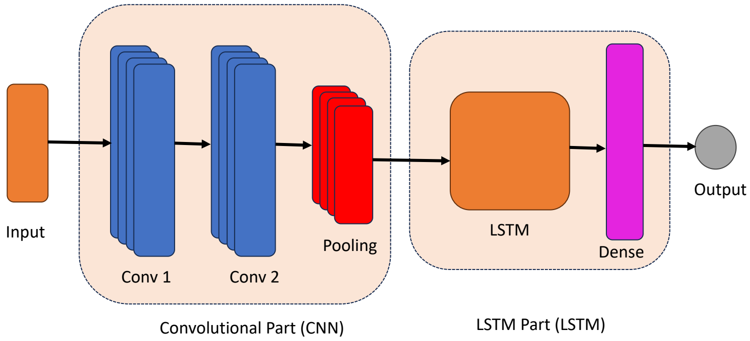


Figure 3: Layers of a demo CNN+LSTM Architecture model

3.3 Reachability Analysis Computation

This section describes how the reachability of an NN layer and the NN as a whole is computed for this study.

In this context, we adopt an alternative technique for defining a Star set. This method involves utilizing the input's upper and lower bounds with noise, subsequently aligning them around the original input. We establish a comprehensive array of constraints by incorporating these bounds for each input parameter alongside predicates. These constraints are then presented to the optimizer for a solution, ultimately yielding the initial set of states.

Definition 3.2 *A **layer** L of a NN is a function $h: \ u \in \mathbb{R}^j \rightarrow v \in \mathbb{R}^p$, defined as follows*

$$v = h(u) \tag{4}$$

where the function h is determined by parameters θ, typically defined as a tuple $\theta = \langle \sigma, W, b \rangle$ for fully-connected and convolutional layers, where W is the weight matrix, b is the bias vector, and activation function is σ. For CNN layers, θ may include parameters like the filter size, padding, or dilation factor.

Definition 3.3 *Let $h: \ u \in \mathbb{R}^j \rightarrow v \in \mathbb{R}^p$, be an NN layer as described in Eq. 4. The **reachable set** $\mathscr{R}_h$, with input, $\mathscr{I} \in \mathbb{R}^n$ is defined as*

$$\mathscr{R}_h \triangleq \{v \mid v = h(u), \ u \in \mathscr{I}\} \tag{5}$$

Reachability analysis (or, shortly, reach) of an NN f on Star input set $\mathscr{I}$ is similar to the reachable set calculations for CNN [44] or FFNN [42].

$$Reach(f, \mathscr{I}): \ \mathscr{I} \rightarrow \mathscr{R}_{ts} \tag{6}$$

We call $\mathscr{R}_{ts}(I)$ the *output reachable set* of the NN corresponding to the input set $\mathscr{I}$.

For an NN, the output reachable set can be calculated as a step-by-step process of constructing the reachable sets for each network layer.

$$\begin{aligned} \mathscr{R}_{L_1} &\triangleq \{v_1 \mid v_1 = h_1(x), \ x \in \mathscr{I}\}, \\ \mathscr{R}_{L_2} &\triangleq \{v_2 \mid v_2 = h_2(v_1), \ v_1 \in \mathscr{R}_{L_1}\}, \\ &\vdots \\ \mathscr{R}_{ts} = \mathscr{R}_{L_k} &\triangleq \{v_k \mid v_k = h_k(v_{k-1}), \ v_{k-1} \in \mathscr{R}_{L_{k-1}}\}, \end{aligned} \tag{7}$$

where h_k is the function represented by the k^{th} layer L_k. The reachable set $\mathscr{R}_{L_k}$ contains all outputs of the neural network corresponding to all input vectors x in the input set $\mathscr{I}$.

3.4 Adversarial Perturbation

An audio classification system may face real-world scenarios involving elements like background noise, interference, or distortions. While potentially perceptible, these factors remain within the scope of challenges that practical systems are designed to address. However, this paper exclusively used l-infinity perturbations, focusing on assessing how audio classification models respond to variations within specific constraints.

Considering an input sequence characterized by t_s time instances and n_f features, various perturbation types (l_∞ norm) [34] arise based on their distribution across the sequence. These adversarial perturbation categories can be delineated as follows:

1. **Single Feature Single-instance (SFSI):** This entails perturbing the value of a specific feature solely at a particular instance (t), deviating by a certain percentage from the actual value:

$$s^{perturb} = g_{\varepsilon, s^{perturb}}(s) = s + \varepsilon_t \cdot s_t^{perturb} \tag{8}$$

2. **Single Feature All-instances (SFAI):** In this scenario, a particular feature across all time instances undergoes perturbation by a certain percentage relative to its original values:

$$s^{perturb} = g_{\varepsilon, s^{perturb}}(s) = s + \sum_{i=1}^{n} \varepsilon_i \cdot s_i^{perturb} \tag{9}$$

3. **Multifeature Single-instance (MFSI):** All feature values experience perturbation, but solely at a specific instance (t), following the principle outlined in Eq. 8 for each feature.
4. **Multifeature All-instance (MFAI):** Perturbation affects all feature values across all instances, aligning with the approach delineated in Eq. 9 for every feature.

3.5 Robustness Verification Properties

Robustness. Robustness pertains to the capacity of a system or model to sustain its performance and functionality amid diverse challenging conditions, uncertainties, or perturbations. This highly desirable trait ensures the system's dependability, resilience, and adaptability in the presence of altering or unfavorable circumstances. To formally articulate the concept of robustness for quantifying the desired classification task, the following formulation can be employed:

$$||x' - x||\infty < \delta \implies f(x') == f(x) \tag{10}$$

Here, x signifies the original input from the input space $R^{n_f \times t_s}$, x' represents the perturbed input, $f(x')$ and $f(x)$ correspond to the classifiers' outputs for x' and x, respectively. δ stands for the maximum magnitude of the introduced perturbation ($\delta \in \mathbf{R} > 0$). By disregarding the softmax and classification layers within the models and focusing on the output of the layer immediately preceding the softmax, the formulation for robustness simplifies as follows:

$$||x' - x||_\infty < \delta \implies \text{maxID}(g(x')) == \text{maxID}(g(x)) \tag{11}$$

In this context, the function g symbolizes the operation performed by the neural network classifier model until the softmax layer, and maxID denotes the function responsible for identifying the class with the highest value in the output.

Verification Properties. Verification properties can be broadly classified into two distinct categories: local and global. A local property must be valid for specific predefined inputs, while a global property [48] is established across the entire input space $R^{n_f \times t_s}$ of the network model, holding true for all inputs without exceptions.

Local Robustness. Given a sequence classifier f and an input sequence S, the network is called **locally robust** to any perturbation $\mathscr{A}$ if and only if: reachable bounds of the desired class will be max compared to the bounds of the other classes, even in the presence of any perturbation.

Robustness Value (RV) of a sequence S is a binary variable, which indicates the local robustness of the system. RV is 1 when the reachable output range of the desired class is greater than the reachable bounds of other classes, making it locally robust; otherwise, RV is 0.

$RV = 1 \iff LB_{desired} \geq UB_{other}$ else, RV = 0

where $LB_{desired}$ and UB_{other} are the lower reachable bound of the desired class and UB_{other} are the upper bounds of all other classes.

Percentage Robustness (PR). We apply the concept of Percentage Robustness (PR), previously utilized in image-based classification or segmentation neural networks [43], to the context of sequence audio inputs. The PR for a sequence classifier, corresponding to any adversarial perturbation, is defined as:

$$PR = \frac{N_{robust}}{N_{total}} \times 100 \tag{12}$$

where N_{robust} represents the total number of robust sequences, and N_{total} is the overall count of sequences in the test dataset. Percentage robustness can be used as an indicator of **global robustness [48]** with respect to various types of perturbations.

4 Reachability of a Long Short Term Memory Layer

To compute the reachability of an LSTM layer in relation to a star input set S_t, a series of stepwise reachability computations are necessary to ultimately determine the reachable set of the LSTM layer's output, as depicted in Eq. 2-3. Ensuring accurate results relies on verifying the validity of specific conditions, which are crucial for this process to be sound and accurate:

1. **Affine Mapping Validity.** The transformation of a star set through an affine mapping using a given weight and bias must result in another valid star set [42].
2. **Star Set Summation.** Combining two star sets through Minkowski summation should lead to the formation of yet another valid star set [5].
3. **Activation Function Application.** Upon applying the activation function to a star set, the output should also result in a star set(s). The outcome could manifest as a single star set or a composition of multiple star sets, contingent on factors such as the activation functions employed and the specific reachability technique utilized [42–45].
4. **Hadamard Product Validity.** The Hadamard Product of two star sets should yield another valid star set.

While the validity of the first three conditions for star sets has been established in prior research, this current study aims to extend that validation to include the fourth condition as well.

Definition 4.1 (Hadamard product of two star sets) *Given two star-sets* $\Theta_1 = \langle c_1, V_1, P_1 \rangle$ *and* $\Theta_2 = \langle c_2, V_2, P_2 \rangle$*, the Hadamard product of them* $\bar{\Theta} = \Theta_1 \odot \Theta_2 = \{y \mid y = x_1 \odot x_2,\ x_1 \in \Theta_1,\ x_2 \in \Theta_2\}$ *is another star with the following characteristics.*

$$\bar{\Theta} = \langle \bar{c}, \bar{V}, \bar{P} \rangle,\ \bar{c} = c_1 \odot c_2,\ \bar{V} = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix},\ \bar{P} \equiv \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}$$

Therefore we can conclude that for a given input set S_t and an LSTM layer, the output is also a star set.

5 Experimental Setup

5.1 Hardware Used:

The actual experimental results shown in this paper are conducted in a Windows-10 computer with the 64-bit operating system, Intel(R) Core(TM) i7-8850H processor, and 16 GB RAM.

5.2 Dataset Description

For evaluation, we consider two different audio datasets for noise classification and Japanese vowel classification.

Audio Noise Data: To curate this dataset, we generated a collection of 1000 white noise signals, 1000 brown noise signals, and 1000 pink noise signals using MATLAB. Each signal corresponds to a 0.5-second duration and adheres to a 44.1 kHz sample rate. From this pool of 1000 signals, a training set is fashioned, comprising 800 white noise signals, 800 brown noise signals, and 800 pink noise signals. Given the multidimensionality inherent in audio data, often containing redundant information, a dimensionality reduction strategy is employed. We begin by extracting features and subsequently training the model using only two extracted features. These features are generated from the centroid and slope of the mel spectrum over time.

Japanese Vowel [3, 22]: This dataset is collected from [3] from the University of Irvine Machine Learning Repository. Two Japanese vowels were sequentially pronounced by nine male speakers. A 12-degree linear prediction analysis was subjected to each instance of utterances. Each speaker's utterance constitutes a time series ranging from 7 to 29 points in length, with each point featuring 12 coefficients. For 9 classes (i.e., vowels), the dataset has a total of 640 time series. Among these, 270 time series were designated for training purposes, while the remaining 370 were allocated for testing.

5.3 Network Description

Audio Noise Data: The network architecture used for training the audio noise dataset, partially adopted from [27], is an LSTM network. The network has two input features which correspond to one noise type at the output. Following 11, the network for this dataset can be represented as:

$$\begin{aligned} f:\ x \in \mathbb{R}^{2 \times l_s} &\rightarrow y \in \mathbb{R}^3 \\ noise\hat{C}lass &= maxID(g(x)) \end{aligned} \tag{13}$$

Japanese Vowel: Here we have trained two different classifiers for the Japanese Vowel dataset. The LSTM architecture is partially adopted from [28] and the CNN+LSTM is partially adopted from [29]. Both the networks have twelve input features which correspond to one vowel at the output. Therefore, the networks for this dataset can be represented as:

$$\begin{gathered} f:\ x \in \mathbb{R}^{12\times l_s} \rightarrow y \in \mathbb{R}^{9} \\ vowe\hat{l}Class = maxID(g(x)) \end{gathered} \tag{14}$$

Here l_s is the audio sequence length and the function *maxID* provides the class with the maximum value.

Table 1: Performances of different networks used in this paper

Networks	*Accuracy*(%)
audio_noise_lstm	100
japanese_vowel_lstm	93.51
japanese_vowel_cnnlstm	96.49

6 Evaluation

6.1 Robustness Verification of Audio Noise Classifier

To conduct robustness verification on the audio noise dataset, we encompass all four categories of perturbations, following [34]. First, we curate 100 sequences each of white, brown, and pink noise as test datasets. Then, we generate adversarial sequences centered around the original ones by applying l_∞ norms. This involves utilizing 5 distinct percentage values for perturbation (ε), specifically 50%, 60%, 70%, 80%, and 90% of the mean (μ) value. These newly created adversarial inputs are subjected to assessment through the exact-star reachability analysis [Sec. 4] to determine their robustness. Notably, in the case of Single Feature Single-instance Noise (SFSI) and Single Feature All-instances Noise (SFAI), we opt for random selection of feature 1 for input perturbation.

Table 2: Global Robustness: percentage robustness (PR) and total verification runtime (sumRT in seconds) for 100 test audio noise sequences

noise	PR_{SFSI}	PR_{SFAI}	PR_{MFSI}	PR_{MFAI}	$sumRT_{SFSI}$	$sumRT_{SFAI}$	$sumRT_{MFSI}$	$sumRT_{MFAI}$
50	98	80.33	98	80.33	0.3071	0.2626	0.3018	0.2625
60	96	71.67	96	71.67	0.3034	0.2571	0.3018	0.2578
70	94	25.67	94	25.67	0.3039	0.2637	0.3070	0.2663
80	85.33	12.33	85.33	12.33	0.3073	0.2559	0.3111	0.2556
90	63.33	8.33	63.33	8.33	0.3060	0.2504	0.3093	0.2537

Observations and Analysis. Table 2 and Fig. 4 present the network's overall performance, i.e., the percentage robustness measures, PR [Sec. 3.5], and total verification runtime (sumRT) in seconds, with respect to each adversarial perturbation. The observations derived from both the table and the figure provide the following insights:

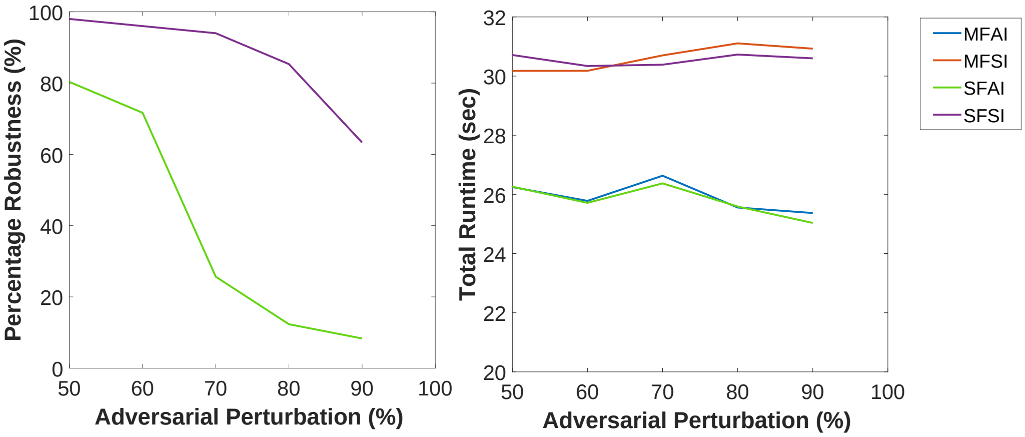


Figure 4: Percentage Robustness and Runtime plots w.r.t increasing perturbations

1. **Trend of Percentage Robustness (*PR*).** As the adversary level increases from 50 to 90, we observe a consistent decrease in PR values for all perturbation scenarios (SFSI, SFAI, MFSI, MFAI), which aligns with the concept of the robustness verification property. This decrease in PR signifies a reduction in the system's ability to maintain its classification accuracy in the presence of higher adversary levels.

2. **Comparative Analysis of Perturbation Scenarios.** Within each noise level, comparing PR values across different perturbation scenarios (SFSI, SFAI, MFSI, MFAI), it's evident that PR values for SFSI and MFSI are generally higher than those for SFAI and MFAI. This finding indicates that perturbing features at a single instance or all features at a single instance generally leads to better robustness against varying noise levels.

3. **Similar PR Values for Different Perturbation Scenarios.** Another notable observation is the similarity in robustness matrices between SFSI and MFSI scenarios, accompanied by closely comparable computation times for their respective verification processes. This parallelism is also evident for SFAI and MFAI perturbations as well. This pattern could be ascribed to the dataset's limited feature set of only two dimensions, where the foremost feature likely holds paramount importance in influencing the class determination in the presence of noise. Consequently, when single-instance perturbations target the first feature, perturbing both features results in an effect akin to perturbing the first feature alone. This interpretation is applicable to both MFAI and SFAI scenarios as well.

6.2 Robustness Verification of Japanese Vowel Classifiers

To verify the robustness of both the LSTM and the CNN+LSTM models in the context of the Japanese vowel classifier, we extend the evaluation to encompass all four perturbation categories, mirroring the approach undertaken for the audio noise classifier. During this procedure, we focus on the complete set of correctly classified test sequences. Subsequently, we create adversarial inputs centered around the original sequences by applying l_∞ norms to evaluate their robustness. This perturbation process involves applying 5 distinct percentage values (ε) for perturbation: specifically, 50%, 60%, 70%, 80%, and 90% of the mean (μ) value. The resulting set of adversarial inputs is then assessed using the exact-star reachability analysis for both the classifiers to ascertain their robustness. Like the earlier scenario, for SFSI and SFAI, feature 1 is chosen for perturbation.

Table 3: Global Robustness: percentage robustness (PR) and total verification runtime (sumRT in seconds) for all test Japanese Vowel audio sequences

noise	PR_{SFSI}	PR_{SFAI}	PR_{MFSI}	PR_{MFAI}	$sumRT_{SFSI}$	$sumRT_{SFAI}$	$sumRT_{MFSI}$	$sumRT_{MFAI}$
50	100	68.21	100	74.86	1.1502	1.0398	1.0392	1.0442
60	100	60.40	100	50.29	0.9989	0.9992	0.9970	0.9966
70	100	50.29	100	27.17	0.9981	0.9968	0.9965	0.9952
80	100	43.93	100	13.01	0.9920	0.9930	0.9978	0.9882
90	100	39.02	100	8.38	1.0044	1.0083	1.004	0.9985

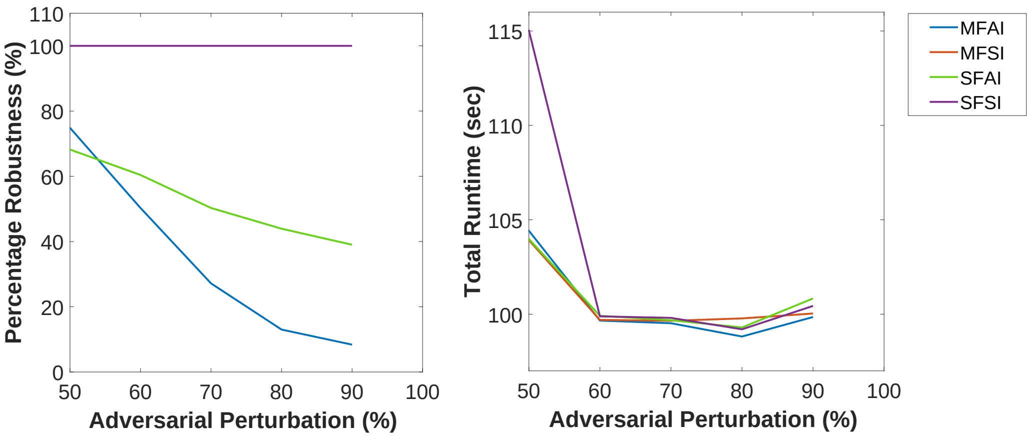


Figure 5: Percentage Robustness and Runtime plots w.r.t increasing perturbations, for LSTM architecture

Observations and Analysis: LSTM Model Table 3 and Fig. 5 present the LSTM network's overall performance, i.e., the percentage robustness measures, PR [Sec. 3.5], and total verification runtime (sumRT), with respect to each adversarial perturbation. The notable findings are outlined as follows

1. **Trend of Percentage Robustness (*PR*).** Similar to the audio noise classifier, the trends in *PR* values here also suggest that as noise levels increase, the percentage robustness tends to decrease across all scenarios. This aligns with the intuitive expectation that higher adversary levels lead to increased challenges in maintaining robustness.

   The PR_{SFSI} and PR_{MFSI} values remain consistently at 100% across all noise levels, indicating that perturbing either a single feature or all features at a specific instance does not significantly affect the robustness of the audio sequences. On the other hand, PR_{SFAI} and PR_{MFAI} show distinct trends. As adversary levels increase, PR_{SFAI} gradually decreases, suggesting that perturbing all instances but only a single feature starts impacting the robustness. Similarly, PR_{MFAI} also experiences a decline with increasing noise levels, reflecting that perturbing all instances and features has an impact on the sequences' robustness.

2. **Comparative Analysis of Perturbation Scenarios.** The comparison between single-instance perturbation scenarios (*SFSI* and *SFAI*) and multifeature perturbation scenarios (*MFSI* and *MFAI*) reveals a pattern. The former scenarios (single-instance) generally maintain higher robustness compared to the latter (multifeature) scenarios. This suggests that perturbing all features has a larger impact on robustness than perturbing just a single feature.

   The interrelation between PR_{SFSI} and PR_{MFSI} is also notable. Both scenarios exhibit identical trends, regardless of the noise level. Similarly, PR_{SFAI} and PR_{MFAI} also demonstrate similar behaviors, with both scenarios showing a decline in robustness as noise increases.

Observations and Analysis: CNN+LSTM Model Table 4 and Fig. 6 present the CNN+LSTM network's overall performance.

Table 4: Global Robustness: percentage robustness (PR) and total verification runtime (sumRT in seconds) for all correctly-classified test Japanese Vowel audio sequences

noise	PR_{SFSI}	PR_{SFAI}	PR_{MFSI}	PR_{MFAI}	$sumRT_{SFSI}$	$sumRT_{SFAI}$	$sumRT_{MFSI}$	$sumRT_{MFAI}$
50	96.82	49.13	97.39	65.89	5.5019	4.2007	4.2197	4.1148
60	96.82	40.75	97.39	43.64	4.5148	3.9238	3.9123	3.9200
70	96.82	34.68	97.10	18.78	4.6223	4.0966	4.0778	4.0626
80	96.82	30.63	97.10	3.17	4.5658	4.0998	4.0550	4.0714
90	96.82	26.58	97.10	0	4.5773	4.0785	4.0715	4.0605

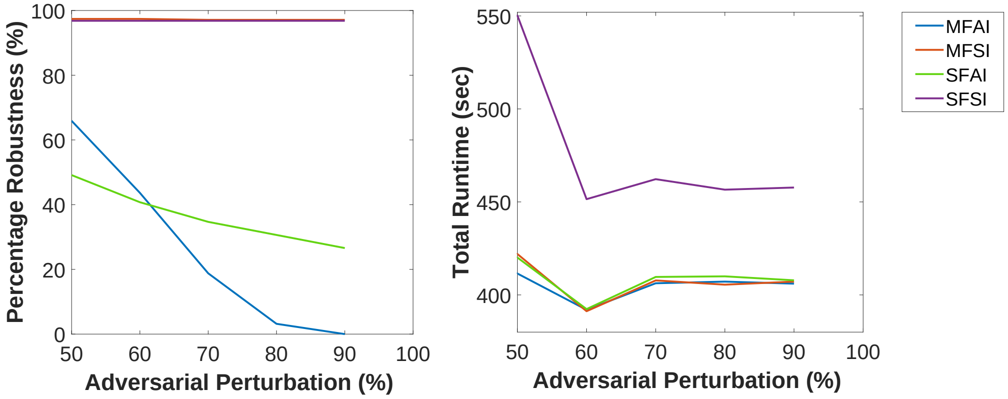


Figure 6: Percentage Robustness and Runtime plots w.r.t increasing perturbations, for LSTM architecture

Key insights gleaned from both the table and the plot include:

1. **Trend of Percentage Robustness (*PR*).** Across all perturbation levels, the PR_{SFSI} remain consistently at around 96% and the PR_{MFSI} at around 97%, indicating that the perturbations applied in these scenarios do not significantly affect the robustness of the audio sequences. For SFAI and MFAI perturbations, PR also decreases with rising noise levels, although the decline is more pronounced. PR values for SFSI and MFSI perturbations are significantly higher compared to SFAI

and MFAI perturbations at all noise levels, indicating that sequences with perturbations at a single instance are more robust to noise.

2. **Trend of Verification Runtimes (*sumRT*).** Verification runtimes tend to rise with elevated noise levels across all perturbation scenarios. However, in the case of the Japanese Vowel dataset, an initial decrease is observed in the runtime trend, followed by an increase at perturbation level 70% and then again decreases at 80%, followed by another increase at 90%. It's also worth noting that contrary to the expected trend, $sumRT_{SFSI}$ exhibits a higher runtime value in comparison to $sumRT_{SFAI}$ and $sumRT_{MFAI}$.

Overall, the above tables demonstrate how different perturbation scenarios and adversary levels impact the percentage robustness of the audio noise and Japanese Vowel audio classifiers. The trends and interrelations provide insights into the varying effects of perturbations on different scenarios and noise levels, helping to understand the robustness behavior of the neural network models under different conditions.

7 Conclusion and Future Directions

This study delves into formal method-based reachability analysis for various LSTM-based neural networks (NNs) using exact and approximate Star methods, specifically in the context of audio sequence classification – a critical aspect for safety-critical applications. The investigation encompasses four distinct adversarial perturbation types, as introduced in the existing literature. The evaluation occurs across two audio sequence datasets: audio noise sequences and Japanese vowel audio sequences. The unified reachability analysis accommodates shifting features within time sequences while scrutinizing the output against the desired audio class. Robustness properties are verified for both datasets. Although real-world datasets are employed, further research is essential to strengthen the connection between practical issues and performance metrics. The evaluation can also be conducted with multiple repetitions to ensure that the reported results are not dependent on specific instances or random fluctuations, thus enhancing the overall validity and reliability of the findings. Exploring real-world scenarios encompassing a wider array of perturbation types and magnitudes will also be fascinating, potentially yielding diverse effects on system behavior. The study paves the way for exploring the impact of perturbations on the output and expanding reachability analysis to three-dimensional sequence data like videos. An intriguing direction for exploration can involve analyzing the peculiar runtime patterns observed in the plots for the Japanese Vowel audio dataset. Potential future applications can also encompass medical video analysis. Notably, this work concentrates on offline data analysis, omitting considerations for real-time stream processing and memory limitations, which offers intriguing prospects for future investigation.

Acknowledgements. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1910017, 2028001, 2220426, and 2220401, and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089 and FA8750-23-C-0518, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

References

[1] Michael E Akintunde, Andreea Kevorchian, Alessio Lomuscio & Edoardo Pirovano (2019): *Verification of RNN-Based Neural Agent-Environment Systems*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, pp. 6006–6013, doi:10.1609/aaai.v33i01.33016006.

[2] Greg Anderson, Shankara Pailoor, Isil Dillig & Swarat Chaudhuri (2019): *Optimization and abstraction: a synergistic approach for analyzing neural network robustness*. In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 731–744, doi:10.1145/3314221.3314614.

[3] Arthur Asuncion & David Newman (2007): *UCI machine learning repository*.

[4] Yusuf Aytar, Carl Vondrick & Antonio Torralba (2016): *Soundnet: Learning sound representations from unlabeled video*. *Advances in Neural Information Processing Systems 29: December 5-10, 2016, Barcelona, Spain*.

[5] Stanley Bak & Parasara Sridhar Duggirala (2017): *Simulation-equivalent reachability of large linear systems with inputs*. In: *International Conference on Computer Aided Verification*, Springer, pp. 401–420, doi:10.1007/978-3-319-63387-9_20.

[6] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin & Christian Schilling (2019): *JuliaReach: a toolbox for set-based reachability*. In: *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 39–44, doi:10.1145/3302504.3311804.

[7] Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio & Ruth Misener (2020): *Efficient verification of relu-based neural networks via dependency analysis*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, pp. 3291–3299, doi:10.1609/aaai.v34i04.5729.

[8] Keunwoo Choi, György Fazekas, Mark Sandler & Kyunghyun Cho (2017): *Convolutional recurrent neural networks for music classification*. In: *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, IEEE, pp. 2392–2396, doi:10.1109/ICASSP.2017.7952585.

[9] Fatih Demir, Daban Abdulsalam Abdullah & Abdulkadir Sengur (2020): *A new deep CNN model for environmental sound classification*. *IEEE Access* 8, pp. 66529–66537, doi:10.1109/ACCESS.2020.2984903.

[10] Mingwen Dong (2018): *Convolutional neural network achieves human-level accuracy in music genre classification*. *arXiv preprint arXiv:1802.09697*.

[11] Peace Busola Falola, Emmanuel Oluwadunsin Alabi, Folashade Titilope Ogunajo & Oluwakemi Dunsin Fasae (2022): *Music genre classification using machine and deep learning techniques: a review*. *ResearchJet J Anal Invent* 3(03), pp. 35–50.

[12] Ian J Goodfellow, Jonathon Shlens & Christian Szegedy (2015): *Explaining and harnessing adversarial examples*. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[13] Andrey Guzhov, Federico Raue, Jörn Hees & Andreas Dengel (2021): *Esresnet: Environmental sound classification based on visual domain models*. In: *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 4933–4940, doi:10.1109/ICPR48806.2021.9413035.

[14] Navid Hashemi, Bardh Hoxha, Tomoya Yamaguchi, Danil Prokhorov, Georgios Fainekos & Jyotirmoy Deshmukh (2023): *A Neurosymbolic Approach to the Verification of Temporal Logic Properties of Learning-enabled Control Systems*. In: *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, pp. 98–109, doi:10.1145/3576841.3585928.

[15] Ezz El-Din Hemdan, Walid El-Shafai & Amged Sayed (2023): *CR19: A framework for preliminary detection of COVID-19 in cough audio signals using machine learning algorithms for automated medical diagnosis applications*. *Journal of Ambient Intelligence and Humanized Computing* 14(9), pp. 11715–11727, doi:10.1007/s12652-022-03732-0.

[16] Sepp Hochreiter & Jürgen Schmidhuber (1997): *Long short-term memory*. *Neural computation* 9(8), pp. 1735–1780, doi:10.1162/neco.1997.9.8.1735.

[17] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu & Xinping Yi (2020): *A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability*. Computer Science Review 37, p. 100270, doi:10.1016/j.cosrev.2020.100270.

[18] Yuval Jacoby, Clark Barrett & Guy Katz (2020): *Verifying recurrent neural networks using invariant inference*. In: Automated Technology for Verification and Analysis: 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19–23, 2020, Proceedings 18, Springer, pp. 57–74, doi:10.1007/978-3-030-59152-6_3.

[19] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić et al. (2019): *The marabou framework for verification and analysis of deep neural networks*. In: International Conference on Computer Aided Verification, Springer, pp. 443–452, doi:10.1007/978-3-030-25540-4_26.

[20] Panagiotis Kouvaros & Alessio Lomuscio (2018): *Formal verification of cnn-based perception systems*. arXiv preprint arXiv:1811.11373.

[21] Alex Krizhevsky, Ilya Sutskever & Geoffrey E Hinton (2012): *Imagenet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems 25: December 3-6, 2012, Lake Tahoe, Nevada, United States, pp. 1106–1114.

[22] Mineichi Kudo, Jun Toyama & Masaru Shimbo (1999): *Multidimensional curve classification using passing-through regions*. Pattern Recognition Letters 20(11-13), pp. 1103–1111, doi:10.1016/S0167-8655(99)00077-X.

[23] Steve Lawrence, C Lee Giles, Ah Chung Tsoi & Andrew D Back (1997): *Face recognition: A convolutional neural-network approach*. IEEE transactions on neural networks 8(1), pp. 98–113, doi:10.1109/72.554195.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio & Patrick Haffner (1998): *Gradient-based learning applied to document recognition*. Proceedings of the IEEE 86(11), pp. 2278–2324, doi:10.1109/5.726791.

[25] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer et al. (2021): *Algorithms for verifying deep neural networks*. Foundations and Trends® in Optimization 4(3-4), pp. 244–404, doi:10.1561/2400000035.

[26] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran & Taylor T Johnson (2023): *NNV 2.0: the neural network verification tool*. In: International Conference on Computer Aided Verification, Springer, pp. 397–412, doi:10.1007/978-3-031-37703-7_19.

[27] *Classify Sound Using Deep Learning - MATLAB & Simulink — mathworks.com*. https://www.mathworks.com/help/audio/gs/classify-sound-using-deep-learning.html.

[28] *Sequence Classification Using Deep Learning - MATLAB &; Simulink — mathworks.com*. https://www.mathworks.com/help/deeplearning/ug/classify-sequence-data-using-lstm-networks.html.

[29] *Sequence Classification Using 1-D Convolutions - MATLAB &; Simulink — mathworks.com*. https://www.mathworks.com/help/deeplearning/ug/sequence-classification-using-1-d-convolutions.html.

[30] Toshio Modegi & Shun-ichi Iisaku (1997): *Application of MIDI technique for medical audio signal coding*. In: Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.'Magnificent Milestones and Emerging Opportunities in Medical Engineering'(Cat. No. 97CH36136), 4, IEEE, pp. 1417–1420, doi:10.1109/IEMBS.1997.756970.

[31] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu & Luca Daniel (2020): *Towards verifying robustness of neural networks against a family of semantic perturbations*. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 244–252, doi:10.1109/CVPR42600.2020.00032.

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi & Pascal Frossard (2016): *Deepfool: a simple and accurate method to fool deep neural networks*. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2574–2582, doi:10.1109/CVPR.2016.282.

[33] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior & Koray Kavukcuoglu (2016): *Wavenet: A generative model for raw audio*. *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, p. 125.

[34] Neelanjana Pal, Diego Manzanas Lopez & Taylor T Johnson (2023): *Robustness verification of deep neural networks using star-based reachability analysis with variable-length time series input*. In: *International Conference on Formal Methods for Industrial Critical Systems*, Springer, pp. 170–188, doi:10.1007/978-3-031-43681-9_10.

[35] Gunasekaran Raja, Senbagapriya Senthilkumar, Sivaseelan Ganesan, Rithika Edhayachandran, Geetha Vijayaraghavan & Ali Kashif Bashir (2021): *AV-CPS: audio visual cognitive processing system for critical intervention in autonomous vehicles*. In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, pp. 1–6, doi:10.1109/ICCWorkshops50388.2021.9473647.

[36] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne & Douglas Eck (2018): *A hierarchical latent vector model for learning long-term structure in music*. In: *International conference on machine learning*, PMLR, pp. 4364–4373.

[37] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening & Marta Kwiatkowska (2019): *Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the L_0 Norm*. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 5944–5952, doi:10.24963/ijcai.2019/824.

[38] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel & Martin Vechev (2018): *Fast and effective robustness certification*. In: *Advances in Neural Information Processing Systems*, pp. 10825–10836.

[39] Gagandeep Singh, Timon Gehr, Markus Püschel & Martin Vechev (2019): *An abstract domain for certifying neural networks*. *Proceedings of the ACM on Programming Languages* 3(POPL), p. 41, doi:10.1145/3291645.

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow & Rob Fergus (2014): *Intriguing properties of neural networks*. In: *2nd International Conference on Learning Representations, ICLR 2014*.

[41] Hoang Dung Tran, Sung Woo Choi, Xiaodong Yang, Tomoya Yamaguchi, Bardh Hoxha & Danil Prokhorov (2023): *Verification of Recurrent Neural Networks with Star Reachability*. In: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 1–13, doi:10.1145/3575870.3587128.

[42] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang & Taylor T Johnson (2019): *Star-based reachability analysis of deep neural networks*. In: *International Symposium on Formal Methods*, Springer, pp. 670–686, doi:10.1007/978-3-030-30942-8_39.

[43] Hoang-Dung Tran, Neelanjana Pal, Patrick Musau, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Stanley Bak & Taylor T Johnson (2021): *Robustness verification of semantic segmentation neural networks using relaxed reachability*. In: *International Conference on Computer Aided Verification*, Springer, pp. 263–286, doi:10.1007/978-3-030-81685-8_12.

[44] Hoang-Dung Tran, Weiming Xiang & Taylor T Johnson (2020): *Verification approaches for learning-enabled autonomous cyber–physical systems*. *IEEE Design & Test* 39(1), pp. 24–34, doi:10.1109/MDAT.2020.3015712.

[45] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak & Taylor T Johnson (2020): *NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems*. In: *International Conference on Computer Aided Verification*, Springer, pp. 3–17, doi:10.1007/978-3-030-53288-8_1.

[46] Finley Walden, Sagar Dasgupta, Mizanur Rahman & Mhafuzul Islam (2022): *Improving the Environmental Perception of Autonomous Vehicles using Deep Learning-based Audio Classification*. *CoRR* abs/2209.04075, doi:10.48550/arXiv.2209.04075.

[47] Avery Wang et al. (2003): *An industrial strength audio search algorithm.* In: *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, Washington, DC, pp. 7–13.

[48] Zhilu Wang, Yixuan Wang, Feisi Fu, Ruochen Jiao, Chao Huang, Wenchao Li & Qi Zhu (2022): *A Tool for Neural Network Global Robustness Certification and Training*. *CoRR* abs/2208.07289, doi:10.48550/arXiv.2208.07289.

[49] Jianfeng Zhao, Xia Mao & Lijiang Chen (2019): *Speech emotion recognition using deep 1D & 2D CNN LSTM networks*. *Biomedical signal processing and control* 47, pp. 312–323, doi:10.1016/j.bspc.2018.08.035.