Robustness Verification of Video Classification Neural Networks

Samuel Sasaki
Department of Computer Science
Vanderbilt University
Nashville, USA
samuel.sasaki@vanderbilt.edu

Diego Manzanas Lopez
Institute for Software Integrated Systems
Vanderbilt University
Nashville, USA
diego.manzanas.lopez@vanderbilt.edu

Preston K. Robinette
Department of Computer Science
Vanderbilt University
Nashville, USA
preston.k.robinette@vanderbilt.edu

Taylor T. Johnson
Department of Computer Science
Vanderbilt University
Nashville, USA
taylor.johnson@vanderbilt.edu

***Abstract*—Deep neural networks (DNNs) have shown impressive performance in computer vision tasks, driven by the availability of large datasets and advanced deep learning techniques. This success has prompted the formal methods community to explore ways of evaluating and improving the reliability and correctness of these models. These verification efforts, however, have primarily focused on neural networks that process image or feature data, overlooking architectures designed for video data. This oversight hinders the deployment of advanced computer vision techniques for safety-critical tasks such as video surveillance, self-driving, and industrial automation, where reliable performance is required. In this work, we develop a formal verification approach for video classification tasks. This includes a novel abstract convex set definition that can represent video data (image data with a temporal component), as well as new reachability methods for layers commonly used by video classification neural networks such as three-dimensional convolutional and three-dimensional pooling layers. Additionally, we construct 3 datasets: Zoom In MNIST, Zoom Out MNIST, and GTRSB Video. These are built upon common datasets from image classification (MNIST, GTSRB). These three datasets and the ST-MNIST dataset are used to train a video classification neural network and then formally verify their robustness against L_∞ norm perturbations. Our results show the scalability of our approach to different input sizes and where the main challenges in this domain are.**

***Index Terms*—video classification, robustness, perturbation**

I. Introduction

The success of deep neural networks (DNNs) in performing image classification tasks has led to their widespread adoption for use cases such as autonomous driving, air traffic collision avoidance systems, and facial recognition systems [1], [2]. However, these classifiers are susceptible to adversarial attacks that can cause failure, leading to catastrophic consequences if they are a component of a safety-critical system [3]. Notably, recent incidents in autonomous driving, e.g., Tesla [4] and Cruise [5], highlight the urgent need for techniques and tools to formally verify the safety and robustness of neural networks (NNs) before deploying them in safety-critical applications.

To evaluate such vulnerabilities, the formal verification community has developed many techniques for verifying various properties of neural networks [6]–[10]. Robustness verification methods, in particular, evaluate the sensitivity of a NN to very slight modifications of an input by checking for consistency amongst the outputs under these changes. Some approaches for robustness verification of NNs include: SMT/SAT-based techniques [6], [11]–[13], optimization [14]–[18], and reachability analysis [19]–[25] among others such as branch and bound methods [26], [27]. However, much of the existing works for robustness verification have targeted image classifiers with comparatively little effort going towards video classifiers. This is a concern because the improvement of deep learning techniques on video data has lead to innovation in video surveillance technologies and integration into the perceptive component of autonomous vehicles [28], [29].

Video classification convolutional neural networks (VCCNNs) enable applications in action recognition, trajectory prediction, and video understanding by leveraging three-dimensional convolutional operations. By using three-dimensional instead of two-dimensional convolutions, the NN effectively learns complex relationships between local patches of pixels as they evolve with the input video over time. This, however, introduces new robustness challenges compared to the two-dimensional convolutional case [30].

To address these existing limitations and as the main contributions of this work, we propose a new abstract convex set representation that we call VideoStars, to represent an infinite number of videos and that enables reachability analysis of VCCNNs. We additionally show reachability properties of VideoStars when propagated through three-dimensional convolutional and pooling layer types that are fundamental to VCCNNs. We implement the set representation in a NN verification tool called NNV [10] and apply it to a combination of existing and custom video datasets.

Related Works. Safety verification and robustness certifi-

Fig. 1: A sample from each of the MNIST Zoom In, MNIST Zoom Out, GTSRB Video, and ST-MNIST datasets used to train video classifiers and to generate L_∞ perturbed input sets for robustness verification. For each sample, the frames of the video are ordered in chronological order from left to right.

cation of DNNs have garnered considerable attention from various communities, e.g., machine learning [7], [20], [31]–[36], formal methods [11], [14], [37]–[42], and security [19], [35], [43]. While there exists a substantial body of research concentrating on the formal verification of feed-forward neural networks (FFNNs), convolutional neural networks (CNNs), and some on semantic segmentation networks (SSNs) and recurrent neural networks (RNNs) using image-based datasets [21], [44]–[53], there appears to be a gap in the literature when it comes to formal verification of these same architectures when using video-based datasets. To the best of the authors' knowledge, there is only one other work on formal verification of video classifiers. In [54], the authors employ an optimization-based approach within a game-theoretic framework to evaluate robustness bounds on adversarial samples generated from the optical flow—the connection between image frames in a video. Their work focuses on a specific type of video classifier that utilizes two-dimensional convolutional and LSTM layers. In contrast, our work performs reachability analysis on a neural network with three-dimensional convolutional layers and evaluates robustness over an infinite set of adversarial inputs generated from the original sample, significantly expanding upon the restricted perturbations of optical flow.

The most similar approach to our proposed method is the ImageStar [47], which is an efficient abstract set representation for reachability analysis of two-dimensional convolutional and other layer types. However, its definition does not support convolution across three dimensions, making it unsuitable for verifying VCCNNs. To address this limitation, we extend this prior work by defining an abstract convex set capable of representing an infinite set of videos. We demonstrate that this representation supports reachability analysis through three-dimensional convolutional and other layer types across a variety of video datasets.

II. Video Classification with Three-Dimensional Convolutional Neural Networks

CNNs are known for their impact on computer vision, particularly for image classification, due to their ability to capture spatial features in the image via convolution. A turning point in computer vision concerning video classification came with the advent of the three-dimensional convolutional layer, which could learn *spatiotemporal* features. By performing convolution across the spatial and temporal dimensions, the NN could consider the evolution of the frames across a video when making its prediction. In the remainder of this section, we introduce formal definitions for the video classification task and VCCNNs, so that we can speak more precisely about three-dimensional convolutional layers and the C3D architecture [55], for which we evaluate robustness.

A. Image and Video Classification

Image classification is the task of taking an input image $x \in \mathbb{R}^{C\times H\times W}$ and mapping it to one of all possible values in a finite set of labels $K = \{1, \ldots, k\}$ where $C, H, W \in \mathbb{R}$ are the number of color channels, height, and width, respectively and k is the number of possible classes. Video classification is much like the image classification task except the input is now a video and so it has the added time dimension. Therefore, we define video classification as the task of taking an input video $v \in \mathbb{R}^{C\times T\times H\times W}$ and mapping it to one of the possible values in a finite set of classes $K = \{1, \ldots, k\}$ where $C, T, H, W \in \mathbb{R}$ are the number of color channels, number of frames, height, and width, respectively and k is the number of possible classes. In the next section, we formally define VCCNNs and introduce three-dimensional convolutions.

B. Video Classification Convolutional Neural Networks

A VCCNN is a CNN used for a video classification task. More formally, a VCCNN $\mathcal{N}$ is a CNN that maps an input video $v \in \mathbb{R}^{C\times T\times H\times W}$ to one of $k \in K$ defined classes, i.e. $\mathcal{N} : v \to K$. Given $\mathcal{N}$ with n layers and an input video v, the class into which $\mathcal{N}$ classifies v is determined by $\mathcal{N}(v) = \arg\max_{c\in K} \mathcal{N}_n(v)$ where $\mathcal{N}_n(v)$ is the output from propagating v through all layers of $\mathcal{N}$.

A key feature of VCCNNs is the use of three-dimensional convolutional layers which capture spatiotemporal features from the video data. Three-dimensional convolutional layers operate just like two-dimensional convolutional layers in that they use a kernel to capture relationships between features of the input data. Three-dimensional convolution, however, uses a three-dimensional kernel and learns spatiotemporal features. To do this, the convolutional layer applies an element-wise multiplication of the kernel to local regions of the input. Figure 2 shows an example of the first iteration of a convolution of an input video.

C. The C3D Architecture

In the same paper that proposed the three-dimensional convolutional layer, the C3D architecture was introduced—a formulation that was crucial to the growth of the video

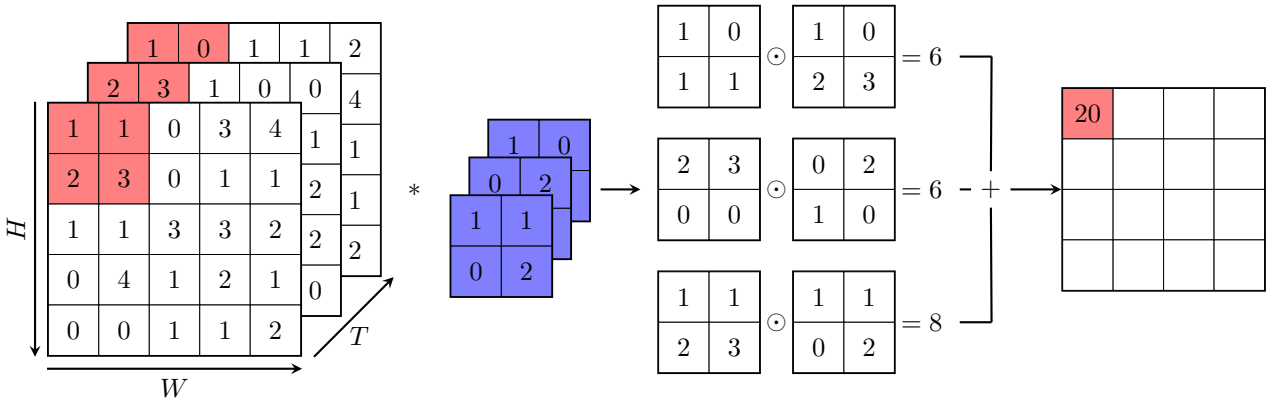


Fig. 2: Visualization of a $(1 \times 3 \times 2 \times 2)$ three-dimensional kernel applied across a $(1 \times 3 \times 5 \times 5)$ dimensional video. The operator $*$ denotes the convolution operation summarized by element-wise multiplications followed by summation over a localized region. We denote the element-wise multiplication with $\odot$.

classification domain [55]. The C3D architecture is made up of 8 convolutional blocks (with a three-dimensional convolutional layer and three-dimensional max pooling layer per block), two fully-connected layers, dropout layers, and ReLU activation layers following each of the convolutional blocks. The C3D architecture is the inspiration for the VCCNNs trained and verified in this work. However, due to the complexity of computing the output reachable set of a VCCNN, we use a pruned version of the architecture that still leverages a convolution block-like structure. This modification allows us to compare performance across models of similar sizes (e.g., verification of image classification neural network) while highlighting the significant gap that remains before formal verification becomes practical for neural network-based video classifiers.

Outline. In the remainder of the paper we will: 1) define reachability analysis and robustness verification for VCCNNs, 2) introduce a novel abstract convex set representation called VideoStars that enable reachability analysis of layer types such as three-dimensional convolutional and pooling layers and formally prove the reachability properties of them, 3) highlight the custom video datasets as well as other components used for implementing and evaluating the proposed method and provide details on the controlled variables and 4) discuss the results of the various experiments and consider future works in formal verification of VCCNNs.

III. Problem Formulation

This section formally defines the robustness verification problem for VCCNNs. We first define reachability analysis for VCCNNs and then show how this method helps us evaluate whether or not a VCCNN is robust against a set of samples within a threshold determined by some $\epsilon \in \mathbb{R}$. We extend the definitions of generalized star sets and the reachability of the convolutional, average pooling, and fully-connected layers established by Tran et al. [22], [47] to define VideoStars and the reachability of three-dimensional convolutional, three-dimensional average pooling, and fully-connected layers with the VideoStar input set.

A. Reachability Analysis

Reachability analysis of VCCNNs is the process of propagating an input or set of inputs through its layers in order to construct a set containing all possible outputs of the NN with respect to its input(s). We consider the reachability of a VCCNN $\mathcal{N}$ that consists of a series of layers of which at least one layer must be a three-dimensional convolutional layer and other potential layers include fully-connected, three-dimensional average pooling, and ReLU activation layers. Mathematically, we define a VCCNN with n layers as $\mathcal{N} = \{\mathcal{N}_i\}$ for $i = 1, 2, \ldots, n$.

Definition 1 (Reachable set of a VCCNN)**.** An (output) reachable set $\mathcal{R}_{\mathcal{N}}$ of a VCCNN $\mathcal{N}$ corresponding to a linear input set $\mathcal{I}$ is defined incrementally as:

$$\begin{aligned}
\mathcal{R}_{\mathcal{N}_1} &\triangleq \{y_1 \mid y_1 = \mathcal{N}_1(x),\, x \in \mathcal{I}\},\\
\mathcal{R}_{\mathcal{N}_2} &\triangleq \{y_2 \mid y_2 = \mathcal{N}_2(y_1),\, y_1 \in \mathcal{R}_{\mathcal{N}_1}\},\\
&\vdots\\
\mathcal{R}_{\mathcal{N}} = \mathcal{R}_{\mathcal{N}_n} &\triangleq \{y_n \mid y_n = \mathcal{N}_n(y_{n-1}),\, y_{n-1} \in \mathcal{R}_{\mathcal{N}_{n-1}}\}
\end{aligned}$$

where $\mathcal{N}$ has n layers and $\mathcal{N}_i(\cdot)$ is a function representing the operation of the i^{th} layer.

The definition shows that the reachable set of the VCCNN $\mathcal{N}$ can be constructed *layer-by-layer*. We compute the reachable set of $\mathcal{N}$ by first constructing the reachable set of each subsequent layer $\mathcal{N}_i$ defined by a specific operation. As convolution, average pooling, and linear operations are fundamental to many VCCNN architectures such as C3D [55], I3D [56], and ResNet3D [57], we later show the reachability of the output set under operations from these layer types.

B. Robustness Verification

This work considers the robustness verification of VCCNNs against L_∞ perturbations. Before defining the L_∞ perturbations, we introduce notation for referencing individual pixel values of the video. Given a video $v \in \mathbb{R}^{C \times T \times H \times W}$, we

denote indexing into the video to reference the particular pixel at color channel c, frame t, row i, and column j with $v_{(c,t,i,j)}$. L_∞ perturbations are subject to the infinity-norm

$$\| x \|_\infty = \max_i |x_i|$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. Additionally, a limit of ϵ is imposed on the difference between any given pixel x_i of an input and its counterpart:

$$|x_i - x_i'| \leq \epsilon, \quad \forall i,$$

where x_i' represents the perturbed pixel values.

Definition 2 (L_∞ Perturbation)**.** Given some value $\epsilon \in \mathbb{R}$ and a video $v \in \mathbb{R}^{C \times T \times H \times W}$, the L_∞ perturbations of v compose the set

$$L_\epsilon = \{\tilde{v}, \text{ if and only if } | v_{(c,t,i,j)} - \tilde{v}_{(c,t,i,j)} | \leq \epsilon\} \quad (1)$$

for all $c \in C$, $t \in T$, $i \in H$, and $j \in W$ and some $\epsilon \in \mathbb{R}$.

Equipped with the definition of a perturbation, we can now formalize what it means for a VCCNN to be robust to an ϵ-valued L_∞ perturbation for an input.

Definition 3. Given a VCCNN $\mathcal{N}$, a video v, and an epsilon perturbation ϵ, the CNN $\mathcal{N}$ is said to be robust to its input v for some ϵ if and only if: $\forall\ \tilde{v} \in L_\epsilon, \mathcal{N}(\tilde{v}) = \mathcal{N}(v)$. If $\exists\ \tilde{v} \in L_\epsilon$ such that $\mathcal{N}(\tilde{v}) \neq \mathcal{N}(v)$, then the video is called non-robust.

Building on the definition of robustness for VCCNNs, we can state the robustness verification problem for VCCNNs as follows: given a VCCNN $\mathcal{N}$, a video v, and a set of L_∞-perturbed videos L_ϵ, show that the video v is robust or non-robust.

IV. Reachability with VideoStars

Introduced by Tran et al. [47], *ImageStars* are an optimized abstract set representation for reachability analysis of two-dimensional convolutional, two-dimensional average pooling, and fully-connected layers. However, their abstraction for images is not sufficient for representing time-dependent video data. Therefore, we extend the notion of an *ImageStar* to create *VideoStar*, a representation based on generalized star sets for the efficient computation of output reachable sets of three-dimensional convolutional, three-dimensional average pooling and fully-connected layers. In the remainder of this section, we formally define *VideoStars* and show their reachability as propagated through three-dimensional convolutional, three-dimensional average pooling, and fully connected layers. We note that $\top$ and $\bot$ refer to true and false values, respectively.

Definition 4. A VideoStar Θ is a tuple (c, V, P, l, u) where $c \in \mathbb{R}^{C \times T \times H \times W}$ is the anchor video, $V = \{v_1, v_2, \ldots, v_m\}$ is a set of m videos in $\mathbb{R}^{C \times T \times H \times W}$ called generator videos, $P : \mathbb{R}^m \to \{\top, \bot\}$ is a predicate, and C, T, H, W are the number of channels, time (in number of frames), height, and width of the videos respectively. The generator videos are arranged to form the VideoStar's $C \times T \times H \times W \times m$ basis array. The set of videos represented by the VideoStar is given as:

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

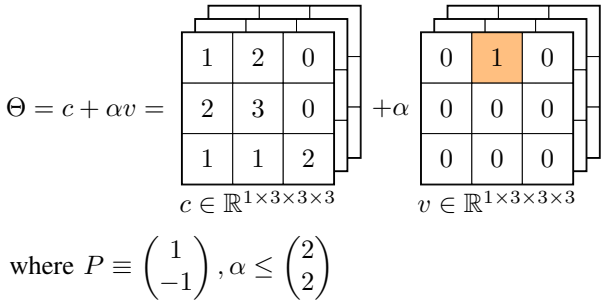


Fig. 3: An example of a VideoStar that is 3-pixels wide (W), 3-pixels high (H), with 1 color channel (grayscale) and 3 time steps (T). The VideoStar describes an infinite set of videos that can be constructed such that pixel $x_{1,1,2} \in \Theta$ can take on any value between $0 \leq x_{1,1,2} \leq 4$.

Sometimes we will refer to both the tuple Θ and the set of states $[\![\Theta]\!]$ as Θ. 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., $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, and $d \in \mathbb{R}^{p \times 1}$. A VideoStar is an empty set if and only if $P(\alpha)$ is empty. Figure 3 is an example visualization of the VideoStar created when trying to perturb a single pixel (highlighted) of a $3 \times 3 \times 3$ video by ± 2 where P, α define our set of linear constraints. The VideoStar defines an infinite set of possible videos that fall in the range where the perturbed pixel is in the range $[0, 4]$.

Given the definition of a VideoStar, we next state a critical lemma used to develop reachability: that VideoStars are closed under affine mappings. That is, given a VideoStar, applying an affine mapping to it gives us another VideoStar by transforming the anchor, generators, and predicate.

Proposition 1 (Affine mapping of a VideoStar)**.** *An affine mapping of a VideoStar $\Theta = (c, V, P, l, u)$ with a scale factor γ and an offset video β is another VideoStar $\Theta' = (c', V', P', l', u')$ in which the new anchor, generators, and predicate are as follows:*

$$c' = \gamma \cdot c + \beta, \ \ V' = \gamma \cdot V, \ \ P' = P, \ \ l' = l, \ \ u' = u.$$

Proposition 2 (Intersection of a VideoStar and a Half-space)**.** *The intersection of a VideoStar $\Theta = (c, V, P, l, u)$ and a half-space $\mathcal{H} \triangleq \{x \mid Hx \leq g\}$ is another VideoStar Θ' with the following characteristics*

$$c' = c, \ \ V' = V, \ \ P' = P \wedge \bar{P}, \ \ l' = l, \ \ u' = u.$$

such that $\bar{P}(\alpha) \triangleq (H \times V_m)\alpha \leq g - H \times c$ with $V_m = [v_1\ v_2\ \ldots\ v_m]$.

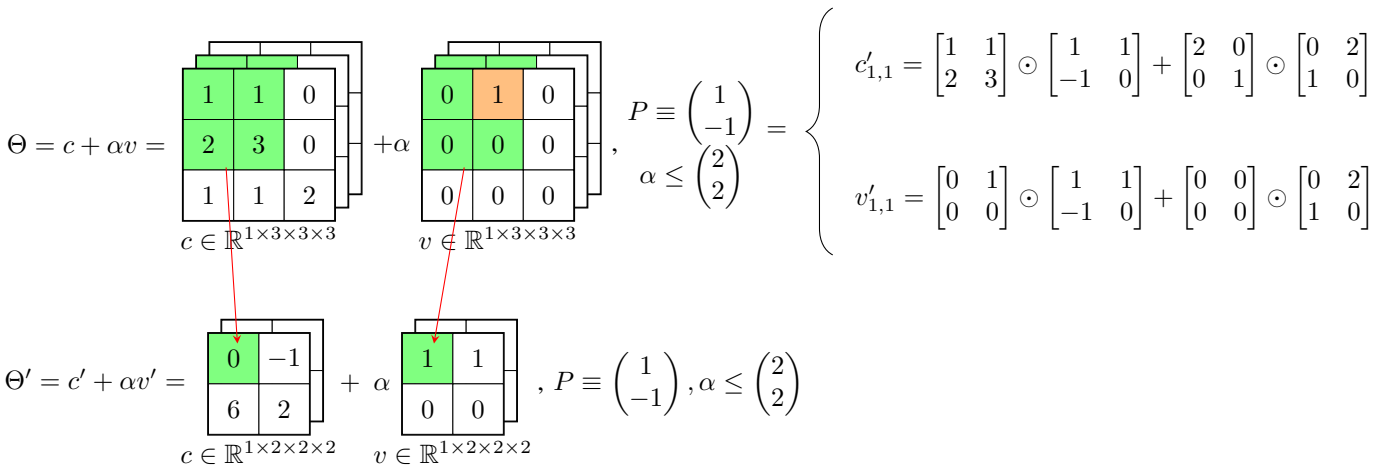


Fig. 4: An example of the output reachable set (VideoStar) of a convolutional layer with a single $1 \times 2 \times 2 \times 2$ (grayscale) filter where the weights and the bias of the filter are $W = \left[\begin{bmatrix}1&1\\-1&0\end{bmatrix}, \begin{bmatrix}0&2\\1&0\end{bmatrix}\right]$, $b = 0$, respectively, the stride is $\mathcal{S} = \begin{bmatrix}1&1&1\end{bmatrix}$, the padding size is $\mathcal{P} = \begin{bmatrix}0&0&0&0&0&0\end{bmatrix}$, and the dilation factor is $\begin{bmatrix}1&1&1\end{bmatrix}$.

In the remainder of this section, we show how to compute the reachable set of several layer types that are critical in VCCNNs, assuming we are representing sets of videos using VideoStars as just defined, and for which we have shown are closed under affine mappings in **Lemma** 1.

A. Reachability of a three-dimensional convolutional layer

We begin by showing how to compute the reachable set of a three-dimensional convolutional layer. Consider a three-dimensional convolutional layer with the following parameters: the weights $\theta_{Conv3d} \in \mathbb{R}^{F\times C\times T_f\times H_f\times W_f}$, the bias $b_{Conv3d} \in \mathbb{R}^{F\times1\times1\times1}$, the padding size $\mathcal{P}$, the stride $\mathcal{S}$, and the dilation factor d where C, T_f, H_f, W_f are the number of color channels, time, height, and width of the filters in the layer respectively. Additionally, F is the number of filters. The reachability of a convolutional layer is given in the following lemma.

Lemma 1. *The reachable set of a convolutional layer with a VideoStar input set $\mathcal{I} = (c, V, P, l, u)$ is another VideoStar $\mathcal{I}' = (c', V', P', l', u')$ where $c' = Convol3D(c)$ is the three-dimensional convolution operation applied to the anchor video, $V' = \{v'_1, \ldots, v'_m\}$, and $v'_i = Convol3DZeroBias(v_i)$ is the three-dimensional convolution operation with zero bias applied to the generator videos, i.e., only using the weights of the layer.*

Proof. By **Definition** 4, any video in the VideoStar $x \in \Theta$ is a linear combination of the center and basis videos, e.g. there exists some α such that $x = c + \alpha v$. Given the formula for a three-dimensional convolution

$$(A * B)_{(h,i,j,k)} = \sum_{i'=i}^{n_i}\sum_{j'=j}^{n_j}\sum_{k'=k}^{n_k}\sum_{h'=1}^{n_h} A_{(\hat{h},\hat{i},\hat{j},\hat{k})} B_{(h',i',j',k')}$$

where $A \in \mathbb{R}^{C\times T\times H\times W}$ is the kernel matrix, $B \in \mathbb{R}^{C\times T'\times H'\times W'}$ is the input video, and the indices to the kernel matrix denoted by $\hat{h}$, for example, are computed $\hat{h} = h' - h$, we want to show that

$$(W * (c + \alpha v)) + b = [(W * c) + b] + \alpha(W * v)$$

where W is the kernel matrix of the convolutional layer and $c + \alpha v$ is the VideoStar. Note that the equation for a convolution $A * B$ is an element-wise multiplication of regions in the matrices A, B and summation of these products. The values nh, ni, nj, nk denote the sum of the initial values, e.g. h, i, j, k, and the size of the kernel in the corresponding dimension C, T, H, W such that we sum together all element-wise products of the kernel. This property is trivial as convolution is an affine transformation and we can rewrite the convolution $W * (c + \alpha v)$ into the sum

$$\begin{aligned}
(W * (c + \alpha v))_{(h,i,j,k)} &= \sum_{i'=i}^{n_i}\sum_{j'=j}^{n_j}\sum_{k'=k}^{n_k}\sum_{h'=1}^{n_h} A_{(\hat{h},\hat{i},\hat{j},\hat{k})}(c + \alpha v)_{(h',i',j',k')} \\
&= \sum_{i'=i}^{ni}\sum_{j'=j}^{nj}\sum_{k'=k}^{nk}\sum_{h'=1}^{nh} W_{(\hat{h},\hat{i},\hat{j},\hat{k})} \cdot c_{(h',i',j',k')} + \\
&\quad \alpha\sum_{i'=i}^{ni}\sum_{j'=j}^{nj}\sum_{k'=k}^{nk}\sum_{h'=1}^{nh} W_{(\hat{h},\hat{i},\hat{j},\hat{k})} \cdot v_{(h',i',j',k')} \\
&= (W * c) + \alpha(W * v)
\end{aligned}$$

Therefore, we have that

$$(W * (c + \alpha v)) + b = [(W * c) + b] + \alpha(W * v)$$

which can be represented as another VideoStar by **Lemma** 1 with $c' = (W * c) + b$ and $v' = \alpha(W * v)$. □

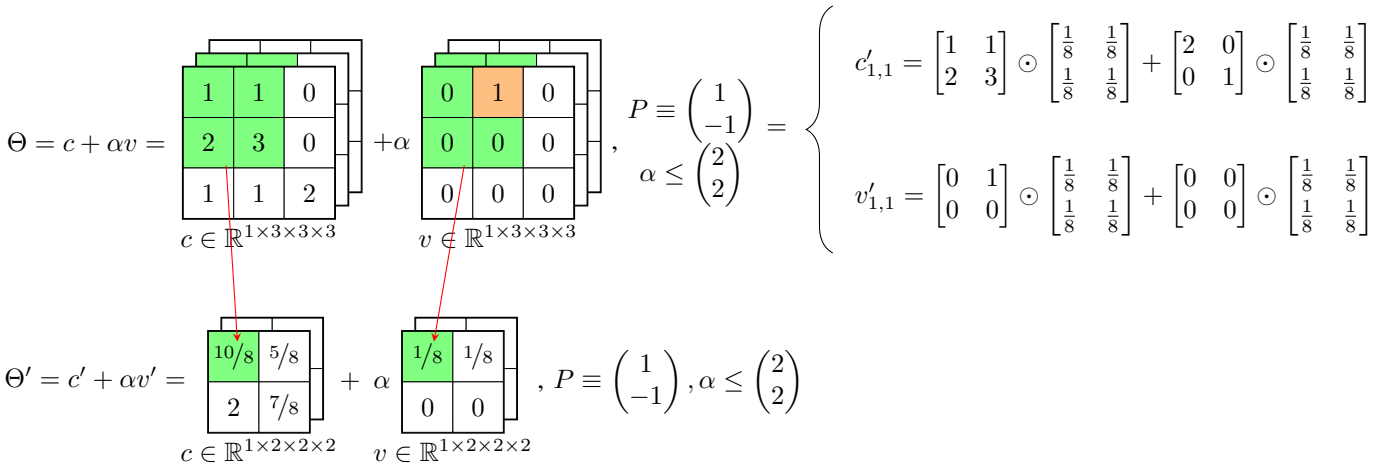


Fig. 5: An example of the output reachable set of a three-dimensional average pooling layer with a single $2 \times 2 \times 2$ filter, all weights of the filter equal to $\frac{1}{2\times2\times2}$, the bias $b = 0$, the stride is $\mathcal{S} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$, the padding size is $\mathcal{P} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$, and the dilation factor is $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$.

Figure 4 demonstrates how a VideoStar is transformed when propagated through a 3D convolutional layer to produce another VideoStar. Next, we show the reachability of a three-dimensional average pooling layer as a special application of the three-dimensional convolutional layer.

B. Reachability of a three-dimensional average pooling layer

The reachability of an average pooling layer with padding size P, and stride S is given below, with its proof similar to that of the convolutional and fully-connected layers.

Lemma 2. *The reachable set of a three-dimensional average pooling layer with a VideoStar input set $\mathcal{I} = (c, V, P, l, u)$ is another VideoStar $\mathcal{I}' = (c', V', P', l', u')$ where $c' = average(c)$, $V' = \{v'_1, \ldots, v'_m\}$, $v'_i = average(v_i)$, and $average(\cdot)$ is the average pooling operation applied to the anchor and generator videos.*

Proof. The three-dimensional average pooling layer can be considered a special application of a three-dimensional convolutional layer. By setting all of the weights of a convolutional layer equal to

$$W_{i,j} = \frac{1}{k_t \cdot k_h \cdot k_w}$$

where the convolutional layer is defined by the kernel matrix $\mathbf{K} \in \mathbb{R}^{C\times k_t \times k_h \times k_w}$ for all $i, j \in \mathbb{N}$, $i \leq C \cdot T \cdot H \cdot W$ and $j \leq C \cdot T' \cdot H' \cdot W'$ and

$$T' = \frac{T - k_t + 2 \times \mathcal{P}_t}{\mathcal{S}_t} + 1$$
$$H' = \frac{H - k_h + 2 \times \mathcal{P}_h}{\mathcal{S}_h} + 1$$
$$W' = \frac{W - k_w + 2 \times \mathcal{P}_w}{\mathcal{S}_w} + 1$$

Therefore, we can conclude that the reachable set of a three-dimensional average pooling layer with a VideoStar input set $\mathcal{I}$ is another VideoStar $\mathcal{I}'$. □

Figure 5 is the process of computing the output set of a 3D average pooling layer which takes a VideoStar as input. Having shown reachability of VideoStars under convolution operations, we turn our attention toward the fully-connected layer. Because of **Lemma** 1, it is trivial to show reachability.

C. Reachability of a fully connected layer

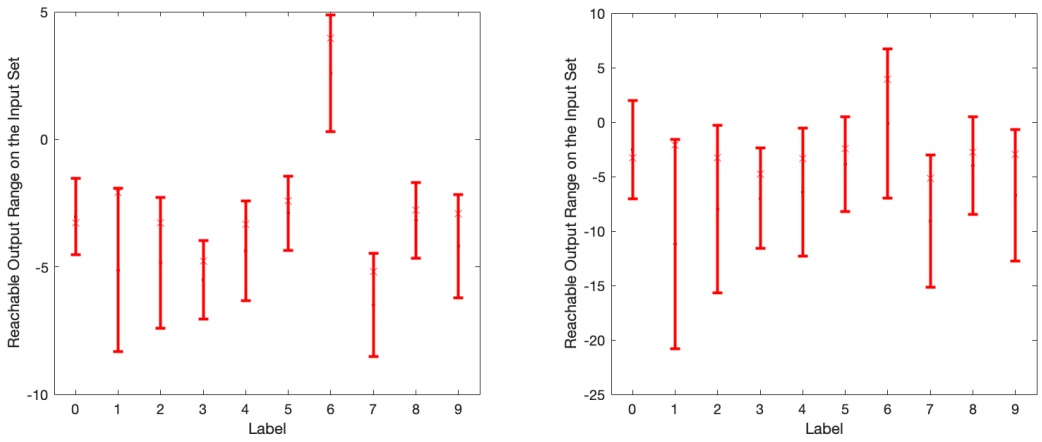
Next, we show the reachability of a fully connected layer.

Lemma 3. *Given a fully connected layer with weight $W \in \mathbb{R}^{n_{f_c} \times m_{f_c}}$, bias $b_{f_c} \in \mathbb{R}^{n_{f_c}}$, and a VideoStar input set $\mathcal{I} = (c, V, P, l, u)$, the reachable set of the layer is another VideoStar $\mathcal{I}' = (c', V', P', l', u')$ where $c' = W \cdot \overline{c} + b$, $V' = \{v'_1, \ldots, v'_m\}$, $v'_i = W \cdot \overline{v_i}$, $\overline{c}(\overline{v}) = reshape(c(v_i), [m_{f_c}, 1])$. Note that it is required for consistency between the VideoStar and the weight matrix that $m_{f_c} = c \times t \times h \times w$ where c, t, h, w are the number of channels, time, height, and width of the VideoStar.*

Proof. This proof follows from the proof for **Lemma** 1. By **Definition** 4, any video in the VideoStar $x \in \Theta$ is a linear combination of the center and basis videos, e.g. there exists some α such that $x = c + \alpha v$. Like in the proof of **Lemma** 1, we reshape the VideoStar to a column vector with $m_{f_c} = c \cdot t \cdot h \cdot w$ number of elements. The output of the linear layer, y, is defined by

$$y = W \cdot (c + \alpha v) + b$$

The fully-connected layer is performing an affine mapping of the input video $c + \alpha v$, so, by linearity of matrix multiplication,



(a) Robust Result (b) Unknown Result

Fig. 6: Visualization of the reachable outputs computed on input sets generated from a sample of the ST-MNIST dataset verified as `robust` (left) and `unknown` (right). The plots generated are on the same ST-MNIST sample with different values of ϵ. The true label for the sample is 6. In the `robust` case, we see that the minimum of the output range for the class label 6 exceeds the maximum output of all other classes, therefore we can conclude that all samples of the input set will be classified with label 6. In the `unknown` case, however, we see that there is overlap in the output ranges, therefore we cannot conclude whether or not the VCCNN is robust to all samples in the generated input set.

we can split this up into

$$y = (W \cdot c) + (W\alpha v) + b$$
$$y = [(W \cdot c) + b] + \alpha(Wv)$$

Therefore, the output y is another VideoStar with updated c' and v' such that $c' = (W\bar{c}) + b$ and $v' = \alpha(W\bar{v})$. □

V. Experimental Setup

To summarize the experimental setup, we verify the robustness of a VCCNN based on the C3D architecture on custom video datasets against L_∞ perturbations using NNV. In the following subsections, we describe each component of the experimental setup in more detail.

A. Datasets

Video classification is a very computationally expensive task because of the high number of features in the data and the increased complexity of the task. This challenge becomes even more pronounced when verifying VCCNNs, as the size of the perturbed input sets for robustness evaluation grows significantly. To address these challenges, we prioritized the creation and use of smaller video datasets for verifying VCCNNs. We developed three novel video datasets based on the MNIST and GTSRB image datasets, thereby enabling robustness verification of VCCNNs on samples with a relatively small number of features.

Zoom In MNIST and Zoom Out MNIST. Using the MNIST dataset, we created two video datasets for these experiments by altering the contents of the images over time to construct a sequence. The resulting datasets transform the MNIST dataset by zooming in or zooming out on the digits to create a video, thereby increasing or decreasing the digit's visibility through time. To introduce more variability in the kinds of data being trained on, we also control the length of the videos being created by altering the number of frames in each video. For each of the Zoom In MNIST and Zoom Out MNIST datasets, we create videos with 4, 8, or 16 frames. To construct these datasets, we first generated the Zoom In MNIST and Zoom Out MNIST datasets to use 16 frames for the video. To construct the 4 and 8-frame video length versions of the datasets, we subsampled specific frames from the 16-frame versions of the datasets, therefore resulting in six variations of the MNIST dataset altogether. By subsampling the frames from the 16-frame video length datasets, we maintained consistency between the videos being verified, therefore allowing us to analyze the impact of increasing the size of the temporal dimension of the video.

Each of the dataset variations has already been split into training (10,000 videos) and testing data (1,000 videos). Each grayscale video has 4, 8, or 16 frames t with a pixel resolution of 32×32, i.e., $v \in \mathbb{R}^{1\times t\times 32\times 32}$, such that $t \in \{4, 8, 16\}$. Figure 1 showcases a progression of frames for samples from each dataset.

GTSRB Video. We used a similar process as outlined previously to construct a slightly more complex video dataset based on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Previous works have shown how susceptible neural networks are to adversarial attacks, which is a major concern, particularly in the autonomous driving industry, because of their widespread use for classifying traffic signs. The ability to recognize traffic signs—the language for communicating the rules of the road—is a crucial capability of autonomous vehicles and can have dangerous consequences if not safely implemented. For this reason, we additionally created the GT-

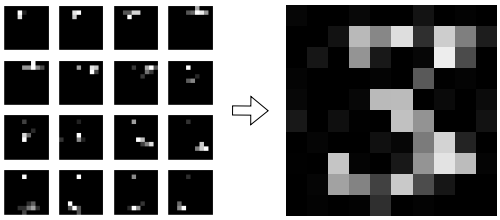


Fig. 7: An example from the ST-MNIST dataset with class **3**. On the left are the individual frames of the video and on the right is the output of overlaying all frames on top of each other. A subset of the frames shown on the left are also available in Figure 1.

TABLE I: Modified C3D model performance on Zoom Out, Zoom In, GTSRB Video, and ST-MNIST variations. Training metrics are reported from top to bottom in each table as 4-frame, 8-frame, 16-frame videos for the Zoom Out, Zoom In, and GTSRB Video datasets. ST-MNIST, on the otherhand, uses 16-frame, 32-frame, and 64-frame videos. All values reported are percentages.

Dataset	Accuracy	Precision	Recall	F1 Score
Zoom Out	84.0	71.6	66.8	68.0
	87.9	76.8	72.8	73.9
	90.0	79.2	75.7	76.7
Zoom In	86.5	73.3	69.8	70.6
	89.6	76.7	73.6	74.4
	89.8	76.5	73.9	74.5
GTSRB	87.5	80.3	80.5	79.8
	87.4	79.7	79.9	79.2
	87.5	79.9	80.3	79.5
STMNIST	73.4	68.2	68.5	65.7
	77.1	71.9	72.1	69.7
	79.9	75.4	75.9	73.3

SRB Video dataset. This dataset is made up of approximately 60,000 videos of German traffic signs from the GTSRB dataset being zoomed in on over time. Like the ZoomIn and ZoomOut MNIST datasets, the videos have height and width of 32×32, with the number of frames in the video varying between 4, 8, and 16. However, these videos are in RGB instead of grayscale, so they maintain 3 times as many features, i.e. $v \in \mathbb{R}^{3\times t\times 32\times 32}$, such that $t \in 4, 8, 16$.

ST-MNIST. Finally, we evaluated VideoStars against L_∞ perturbations on the ST-MNIST dataset [58]. This dataset contains 6953 samples of 23 participants writing a digit (0-9) 30 times each. As such, each sample records the contact location of a pen when writing a specific digit on a canvas over time. These videos are compiled into samples of dimension $2 \times t \times 10 \times 10$ where $t \in \mathbb{N}$ is the number of frames. On the left side, Figure 8 shows a breakdown of the progression of frames for a sample of the ST-MNIST dataset going from top to bottom and left to right. The right side of the figure demonstrates how the frames come together to produce a now classifiable sample where the temporal sequencing of the frames is a learnable feature that distinguishes it from other samples. Unlike in the previously used datasets, it will be virtually impossible for the VCCNN to accurately classify samples using only the spatial features because each individual frame carries so little information with it. It is the composition of the frames together that make a distinguishable sample. Therefore, this dataset is a low-cost experimentation of robustness verification for VCCNNs where the temporal component has high impact on the task.

The only caveat to this dataset is that the videos have varying length in number of frames. To remedy this, like in the custom video datasets, variations of the ST-MNIST dataset were designed to have a target number of frames. If a specific sample did not have enough frames to satisfy the requirement, the sample was padded. We use ST-MNIST samples with 16, 32, and 64 frames in our experiments, i.e. $v \in \mathbb{R}^{2\times t\times 10\times 10}$, such that $t \in 16, 32, 64$.

B. Video Classification Neural Network Architecture

The C3D neural network architecture inspired the VCCNN architecture used in these experiments because of its significance to the progression of video classification via deep learning in the last decade. In our experience, however, the result of using this architecture for the Zoom In or Zoom Out MNIST datasets was an NN with greater than 10 million parameters and a large number of ReLU activation neurons.

To reduce the complexity of the VCCNNs, we instead use only the intuition behind the C3D neural network architecture, the three-dimensional convolutional block. We use a single three-dimensional convolutional block in the architecture with three-dimensional average pooling instead of a three-dimensional max pooling layer. Additionally, we reduce the number of output channels to 8 from the 512 output channels

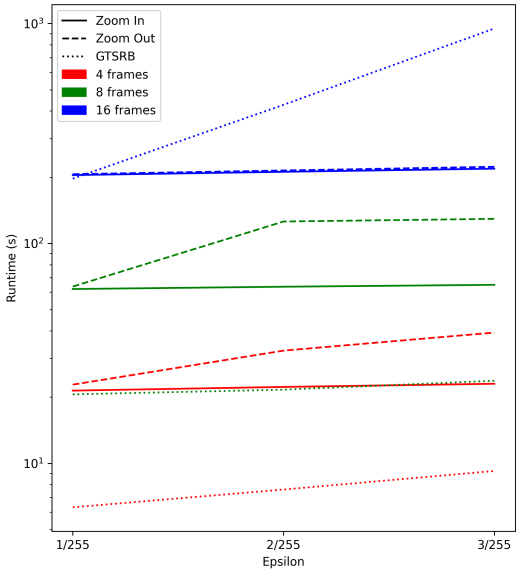


Fig. 8: A comparison of the average running times across the Zoom In and Zoom Out MNIST and GTSRB Video datasets. The average runtime is measured in seconds.

TABLE II: Robustness verification results for a VCCNN evaluated on the ZoomIn, ZoomOut, GTSRB Video, and ST-MNIST datasets against L_∞-perturbed samples represented as VideoStars and verified with the *relax* and *approx* reachability methods. The metrics reported for each dataset include PRSV and average runtime across all provided samples.

Reachability Method	Metric	Dataset	Epsilon		
			1/255	2/255	3/255
Relax	PRSV (%)	ZoomIn-4f	98	94	93
		ZoomIn-8f	100	96	93
		ZoomIn-16f	100	99	98
		ZoomOut-4f	97	96	93
		ZoomOut-8f	100	98	97
		ZoomOut-16f	100	99	98
		GTSRB-4f	93.49	79.07	64.19
		GTSRB-8f	90.70	78.14	64.65
		GTSRB-16f	91.16	74.88	57.21
		STMNIST-16f	80	46	27
		STMNIST-32f	77	37	23
		STMNIST-64f	41	12	5
	Avg. Time (s)	ZoomIn-4f	21.41	22.25	23.00
		ZoomIn-8f	62.03	63.53	64.81
		ZoomIn-16f	204.42	211.56	218.68
		ZoomOut-4f	22.79	32.54	39.31
		ZoomOut-8f	63.64	125.74	129.30
		ZoomOut-16f	206.37	214.54	223.02
		GTSRB-4f	6.31	7.60	9.23
		GTSRB-8f	20.59	21.62	23.74
		GTSRB-16f	197.03	426.74	948.58
		STMNIST-16f	22.35	27.81	35.06
		STMNIST-32f	13.06	15.05	20.27
		STMNIST-64f	6.76	7.91	300.18
Approximate	PRSV (%)	ZoomIn-4f	98	94	93
		ZoomIn-8f	100	96	93
		ZoomOut-4f	97	96	93
		ZoomOut-8f	100	98	97
		GTSRB-4f	93.49	79.07	64.19
		GTSRB-8f	90.70	78.14	64.65
		GTSRB-16f	91.16	74.88	56.74
		STMNIST-16f	80	46	27
		STMNIST-32f	77	37	23
		STMNIST-64f	41	12	5
	Avg. Time (s)	ZoomIn-4f	39.58	40.99	42.44
		ZoomIn-8f	111.36	113.87	116.58
		ZoomOut-4f	41.86	59.99	72.33
		ZoomOut-8f	115.43	227.01	234.41
		GTSRB-4f	6.99	9.01	11.41
		GTSRB-8f	20.00	21.62	24.77
		GTSRB-16f	222.43	459.89	1044.40
		STMNIST-16f	35.60	40.75	47.57
		STMNIST-32f	24.04	26.33	32.44
		STMNIST-64f	7.68	9.68	304.52

in the final convolutional block of the C3D architecture. Finally, we eliminated the dropout and the two fully connected layers. In doing so, we reduced the number of parameters to anywhere from 60,000-300,000, depending on the length of the videos used to train the neural network. Table I shows a breakdown of the performance for each neural network on the testing data.

C. Tools

We selected the NNV tool because of VideoStar's relation to ImageStar as a modification of generalized star sets. It is also a considerable benefit that three-dimensional convolutional layers and three-dimensional average pooling layers are supported by the NN framework used by the NNV tool for reachability analysis. Other tools such as nnenum [25] and CORA [23] were considered but ultimately failed to support the use of three-dimensional convolutional layers critical to the purpose

of this work.

D. Evaluation

For each of the Zoom Out, Zoom In, and ST-MNIST dataset variations, we randomly select 100 (10 per class) originally correctly classified videos from the testing datasets to verify. The input sets are generated using L_∞ perturbations for all $\epsilon \in \{1/255, 2/255, 3/255\}$ due to the normalization of the input pixel values to be between $[0, 1]$. In the case of the GTSRB Video dataset, we verify 215 samples (5 per class). We enforce a timeout of 1800s for verifying any single input set. Additionally, we use two different reachability methods to compute the output reachable sets of the neural networks. The first reachability method is the relaxed reachability method, which we will refer to as *relax* [59]. The second reachability method is an overapproximation reachability method (*approx*) [22]. The *relax* reachability method builds off of *approx* to reduce the number of linear programming problems it needs to solve and improve the time required to verify the sample, but the consequences of which are a more conservative estimate of the output reachable sets. Because we are using overapproximate methods for verification, for any given input set being verified, the possible outcomes are `robust`, `non-robust`, `unknown`, and `timeout`. Figure 6 shows the reachable outputs for two input sets generated from a sample of the ST-MNIST dataset with outcomes `robust` and `non-robust`.

All verification experiments were run on an Apple M1 Max 10-core CPU@3.20GHz×10 with 64GB of RAM using the modified NNV tool with the added support for reachability analysis of three-dimensional convolutional layers as stated in **Lemma** 1.

VI. Results

We evaluate the robustness verification of the video classification neural networks using two metrics: proportion of robust samples verified and runtime. Proportion of robust samples verified (PRSV) is the proportion of samples that are verified to be robust out of all samples being verified, e.g. if there were 80 samples verified to be robust out of 100 samples then the PRSV is 80%. Because we are using overapproximate reachability methods, it is important to note that those samples whose robustness are neither proven nor disproven do not count as robust under this definition. Additionally, we measure how long it takes to verify the sample.

Table II summarizes the PRSV and average running times for all samples verified for each of the experiments run at their different ϵ values. An important takeaway from these results is that there exists a relationship between the difficulty of the video classification task and the ease with which an input can be adversarially perturbed. Starting with the MNIST-based video datasets and increasing with complexity to the GTSRB Video dataset, which is RGB instead of grayscale, has 43 classes instead of 10, and contains complicated shape and color combinations, ultimately leads to a degradation in robustness of the VCCNN under the same architecture. Similarly, going from GTSRB Video to ST-MNIST, which contains only 10×10 features to learn from, dependence on the temporal component of the data, and high sparsity of the input videos, leads to a highly sensitive VCCNN.

Also, as expected, the increased number of features in the samples and complexity of the task and NNs makes robustness verification for VCCNNs a computationally expensive problem. From the results, we observe a clear correlation between the number of features, much more so than ϵ, and the increase in average runtime of the verification methods. It is also important to note how easily the runtime of these verification algorithms can grow as in the runtime seen from using the *relax* method on the 16-frame GTSRB Video dataset. Additionally, the runtime results of reachability on the perturbed 64-frame ST-MNIST samples shows that the runtime bottleneck in solving LP problems is even more pronounced, as expected, with video as opposed to image data.

VII. Conclusion and Future Work

State-of-the-art deep learning techniques are incredibly powerful but cannot be trusted in safety-critical systems due to their opaque nature. This work aims to bridge the gap between modern deep learning and formal verification by adapting reachability analysis to video classification tasks.

In this work, we present VideoStars, an abstract convex set representation for representing infinite sets of four-dimensional video data. We show the reachability of VideoStars under propagation through layer types fundamental to VCCNNs such as three-dimensional convolution and average pooling, and fully-connected. Additionally, we construct three new video datasets based on the MNIST and GTSRB datasets to make robustness verification of video classifiers a more accessible research direction.

In future work, we want to explore the increased complexity of possible perturbations or attacks on the video input that result from the added temporal dimension. The set of possible perturbations increases as the small changes can be distributed throughout a much greater space than when limited to only the spatial dimension. As robustness verification of NNs on high-dimensional inputs is a computationally costly problem, we plan to investigate other approaches based on abstract interpretation to improve the scalability of the presented work.

Acknowledgements

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 2220401 and 2220426 and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA or NSF.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *arXiv preprint arXiv:1810.04240*, 2018.

[3] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[4] B. Goggin, *A Tesla owner says his car's 'self-driving' technology failed to detect a moving train ahead of a crash caught on camera*, 2024. [Online]. Available: https://www.nbcnews.com/tech/tech-news/tesla-owner-says-cars-self-driving-mode-fsd-train-crash-video-rcna153345

[5] D. Shepardson, *GM's Cruise recalling 950 driverless cars after pedestrian dragged in crash*, 2023. [Online]. Available: https://www.reuters.com/business/autos-transportation/gms-cruise-recall-950-driverless-cars-after-accident-involving-pedestrian-2023-11-08/

[6] H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, P. Huang, O. Lahav, M. Wu, M. Zhang, E. Komendantskaya, G. Katz, and C. Barrett, "Marabou 2.0: A versatile formal analyzer of neural networks," 2024.

[7] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems*, 2018, pp. 4944–4953.

[8] Y. Y. Elboher, J. Gottschlich, and G. Katz, "An abstraction-based framework for neural network verification," in *Computer Aided Verification*, S. K. Lahiri and C. Wang, Eds. Cham: Springer International Publishing, 2020, pp. 43–65.

[9] D. Shriver, S. Elbaum, and M. B. Dwyer, "Dnnv: A framework for deep neural network verification," in *Computer Aided Verification*, A. Silva and K. R. M. Leino, Eds. Cham: Springer International Publishing, 2021, pp. 137–150.

[10] D. M. Lopez, S. W. Choi, H.-D. Tran, and T. T. Johnson, "Nnv 2.0: The neural network verification tool," in *Computer Aided Verification*, C. Enea and A. Lal, Eds. Cham: Springer Nature Switzerland, 2023, pp. 397–412.

[11] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[12] H. Duong, T. Nguyen, and M. Dwyer, "A dpll(t) framework for verifying deep neural networks," 2024.

[13] Z. Liu, P. Yang, L. Zhang, and X. Huang, "Deepcdcl: A cdcl-based neural network verification framework," in *Theoretical Aspects of Software Engineering*, W.-N. Chin and Z. Xu, Eds. Cham: Springer Nature Switzerland, 2024, pp. 343–355.

[14] S. Dutta, S. Jha, S. Sanakaranarayanan, and A. Tiwari, "Output range analysis for deep neural networks," *arXiv preprint arXiv:1709.09130*, 2017.

[15] K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," in *Proceedings of the Thirty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-18)*. Corvallis, Oregon: AUAI Press, 2018, pp. 162–171.

[16] W. Lin, Z. Yang, X. Chen, Q. Zhao, X. Li, Z. Liu, and J. He, "Robustness verification of classification deep neural networks via linear programming," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 410–11 419.

[17] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] H. Duong, D. Xu, T. Nguyen, and M. B. Dwyer, "Harnessing neuron stability to improve dnn verification," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3643765

[19] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 6369–6379.

[20] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 825–10 836.

[21] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3290354

[22] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *Formal Methods – The Next 30 Years*, M. H. ter Beek, A. McIver, and J. N. Oliveira, Eds. Cham: Springer International Publishing, 2019, pp. 670–686.

[23] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[24] T. Ladner and M. Althoff, "Exponent relaxation of polynomial zonotopes and its applications in formal neural network verification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 19, pp. 21 304–21 311, Mar. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/30125

[25] S. Bak, "nnenum: Verification of relu neural networks with optimized abstraction refinement," in *NASA Formal Methods Symposium*. Springer, 2021.

[26] R. Bunel, I. Turkaslan, P. H. S. Torr, M. P. Kumar, J. Lu, and P. Kohli, "Branch and bound for piecewise linear neural network verification," *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020.

[27] C. Ferrari, M. N. Mueller, N. Jovanović, and M. Vechev, "Complete verification via multi-neuron relaxation guided branch-and-bound," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=l_amHf1oaK

[28] W. Shinego, "Defense department tests ai software, advances to improve physical security posture," https://www.defense.gov/News/News-Stories/Article/Article/3946607/defense-department-tests-ai-software-advances-to-improve-physical-security-post/, 2024.

[29] M. Biparva, D. Fernández-Llorca, R. I. Gonzalo, and J. K. Tsotsos, "Video action recognition for lane-change classification and prediction of surrounding vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 569–578, 2022.

[30] M. C. Schiappa, N. Biyani, P. Kamtam, S. Vyas, H. Palangi, V. Vineet, and Y. Rawat, "Large-scale robustness analysis of video action recognition models," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[31] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.

[32] P. Kouvaros and A. Lomuscio, "Formal verification of cnn-based perception systems," *arXiv preprint arXiv:1811.11373*, 2018.

[33] M. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano, "Reachability analysis for neural agent-environment systems," in *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018.

[34] M. E. Akintunde, A. Kevorchian, A. Lomuscio, and E. Pirovano, "Verification of rnn-based neural agent-environment systems," in *Proceedings of the 33th AAAI Conference on Artificial Intelligence (AAAI19). Honolulu, HI, USA. AAAI Press. To appear*, 2019.

[35] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," *arXiv preprint arXiv:1804.10829*, 2018.

[36] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel, "Towards fast computation of certified robustness for relu networks," *arXiv preprint arXiv:1804.09699*, 2018.

[37] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 243–257.

[38] H.-D. Tran, P. Musau, D. M. Lopez, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Parallelizable reachability analysis algorithms for feed-forward neural networks," in *7th International Conference on Formal Methods in Software Engineering (FormaliSE2019), Montreal, Canada*, 2019.

[39] W. Xiang, H.-D. Tran, and T. T. Johnson, "Specification-guided safety verification for feedforward neural networks," *AAAI Spring Symposium on Verification of Neural Networks*, 2019.

[40] ——, "Reachable set computation and safety verification for neural networks with relu activations," *arXiv preprint arXiv:1712.08163*, 2017.

[41] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 3–29.

[42] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–7, 2018.

[43] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai 2: Safety and robustness certification of neural networks with abstract interpretation," in *Security and Privacy (SP), 2018 IEEE Symposium on*, 2018.

[44] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić *et al.*, "The marabou framework for verification and analysis of deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2019, pp. 443–452.

[45] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, "Global robustness evaluation of deep neural networks with provable guarantees for the l_0 norm," *arXiv preprint arXiv:1804.05805*, 2018.

[46] G. Anderson, S. Pailoor, I. Dillig, and S. Chaudhuri, "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*, ser. PLDI 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 731–744.

[47] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," in *32nd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2020.

[48] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 4939–4948.

[49] S. Dathathri, K. Dvijotham, A. Kurakin, A. Raghunathan, J. Uesato, R. Bunel, S. Shankar, J. Steinhardt, I. Goodfellow, P. Liang, and P. Kohli, "Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming," 2020.

[50] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations*, 2019.

[51] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[52] J. Mohapatra, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, "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 (CVPR)*, June 2020.

[53] E. Botoeva, P. Kouvaros, J. Kronqvist, A. Lomuscio, and R. Misener, "Efficient verification of relu-based neural networks via dependency analysis," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3291–3299, Apr. 2020.

[54] M. Wu and M. Kwiatkowska, "Robustness guarantees for deep neural networks on videos," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 308–317.

[55] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497.

[56] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4724–4733.

[57] K. Hara, H. Kataoka, and Y. Satoh, "Learning spatio-temporal features with 3d residual networks for action recognition," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 3154–3160.

[58] H. H. See, B. Lim, S. Li, H. Yao, W. Cheng, H. Soh, and B. C. K. Tee, "St-mnist – the spiking tactile mnist neuromorphic dataset," 2020. [Online]. Available: https://arxiv.org/abs/2005.04319

[59] H.-D. Tran, N. Pal, P. Musau, D. M. Lopez, N. Hamilton, X. Yang, S. Bak, and T. T. Johnson, "Robustness verification of semantic segmentation neural networks using relaxed reachability," in *Computer Aided Verification*, A. Silva and K. R. M. Leino, Eds. Cham: Springer International Publishing, 2021, pp. 263–286.