Set-Theoretical Contradiction Analysis via Object Detection and Convolutional Neural Net Image Classification with Augmented Reality Applications

Cory Ye, *Member, IEEE*, Andrew Deng, *Member, IEEE*

***Abstract*—The pervasion of DSP-driven augmented reality embedded systems/devices in modern society has changed the way we interact with technology and learn. One particular example is Photomath™, an application that utilizes the camera of a smart device to recognize the image/symbols of an algebraic equation and compute the solution of the equation in real time. In this paper, we propose a variant of this application – the Set Theoretical Contradiction Analyzer (STCA) – that analyzes set theoretic statements to identify mistakes or contradictions in mathematical arguments and proofs. To realize the objective, we will implement image recognition DSP software equipped with a convolutional neural network (CNN) to analyze and convert set theory symbolics/notation to satisfiability (SAT) on embedded systems in real time. Ideally, the amateur mathematician or logician can extract and list all set topological statements created in a closed proof (without external dependencies/statements), and apply the STCA-equipped device to either proofread direct proofs or solve proofs via contradiction/contrapositive. Moreover, we analyze the drawbacks of programming NP-complete algorithms and implementing deep neural network augmented reality image processing on a computationally-limited embedded system.**

***Index Terms*—Relatively-Finite Set Theory, Logic, Convolutional Neural Network, MATLAB, Image Recognition, DSP, Recursive DFS Implication (NP-Complete) Search Algorithms, Augmented Reality, TI LCDK Embedded System, ECE 113D**

I. Introduction

A. Context

The application of formal methodology and mathematical theory in engineering has accelerated in proportion to the advancement of computation, machine learning, data science, physics, and pure/applied mathematics in the 21^{st} century. For instance, non-linear control theory applied in the context of self-driving vehicles, drones, and spacecraft require controller optimizations via the language of reachable sets, probability/measure spaces, automata theory, and differentiable manifolds. In particular, axiomatic (ZFC) set theory is the universal language that connects and objectifies mathematical theory, permitting engineers, physicists, and mathematicians alike to rigorously communicate comprehensive theoretical ideas in the design and implementation of contemporary technology.

B. Fundamentals of Set Theory

Set Theory is the subfield of mathematical logic that studies sets, arbitrary collections of mathematical objects. [**?**] There exist abstract elements $\{\alpha, \beta, \gamma, \dots\}$ that are contained in sets $\{A, B, C, \dots\}$, denoted by the set theoretical statement syntax $\alpha \in S$ if and only if the element α is contained in the set S. Moreover, there exist multiple binary relations between sets that describe the content and/or topology of a collection of sets. In particular, we say that $A \subset B$ if and only if the set A is a subset of the set B if and only if any element contained in A is contained in B, equivalent with the logical implication $\alpha \in A \implies \alpha \in B$. In addition, $\bigcup_{i\in I} H_i$ is the union of sets, satisfying $H_i \subset \bigcup_{i\in I} H_i$, and $\bigcap_{i\in I} T_i$ is the intersection of sets, satisfying $\bigcap_{i\in I} T_i \subset T_i$.

C. Satisfiability (SAT)

Boolean Satisfiability (SAT) is a logical formulation or algorithm that transforms any solvable problem to a conjunctive normal form (CNF) string of atomic satisfiability propositions ϕ that evaluates to either TRUE or FALSE. [**?**] For instance, the atomic proposition $\phi = \{\alpha \in A\}$ evaluates as TRUE if $\alpha \in A$ and FALSE if $\alpha \notin A$, and $(\phi \vee \psi) \wedge \chi$ is TRUE if and only if $\phi \wedge \chi$ or $\psi \wedge \chi$ or $\phi \wedge \psi \wedge \chi$ is TRUE. In application, a mathematical structure $Q \models \phi$ if and only if the assumptions in Q satisfies the atomic SAT formula ϕ. Via DeMorgan's Laws, the string of satisfiability propositions can be reduced to conjunctive normal form (CNF).

$$\bigwedge_i \bigvee_j \phi_{ij}$$

Satisfiability is a useful tool to analyze the validity of any mathematical or logical argument. In particular, it suffices to decompose and translate all statements introduced in a logical argument to atomic propositions in SAT. In the general case, such a translation algorithm is difficult to implement, because many mathematical or logical statements have multi-dimensional, infinite, or abstract semantics. For example, it is not computationally feasible to translate the set topological statement: $K \subset \mathbb{R}^n$ is compact if and only if for any open cover $\mathcal{O} = \bigcup U_k \models K \subset \mathcal{O}$, there exists a finite sub-cover $K \subset \bigcup_{i=1}^{N} U_k \subset \mathcal{O}$ to SAT, because the definition of an open cover requires data corresponding to infinitely many combinatorial unions and intersections of sets, and the definition of compactness induces notions of Noetherian induction, a type of inductive reasoning that generates infinitely many sets, set unions, or set intersections. Hence, any realizable SAT translation or application pertaining to the analysis of elements and sets is limited to relatively-finite set theory, the class of set-theoretic problems that involve a finite amount of set theoretical symbols/objects and atomic SAT propositions that can be programmed and computed in an embedded system.

However, satisfiability is an NP-complete problem, i.e. any algorithm that solves an arbitrary logical SAT formula executes in non-deterministic polynomial time. In particular, determining if there exists a logical contradiction in a SAT formula is NP-hard, because set-theoretic inter-dependencies between SAT formulae occlude the existence of an optimal deterministic algorithm. Necessarily, the STCA system proposed in this paper is computationally constrained by the SAT analysis algorithm that solves the set-theoretic SAT problem.

D. Convolutional Neural Nets and Image Recognition

Building a system that can correctly classify images into a large number of classes is a daunting task. Image data typically comes in the form of long vectors that represent the intensities of each individual pixel, which is very difficult to handle with heuristic classification methods; it is hard to manually program logic to check for different shapes in the image while still allowing shapes to vary by small amounts. Even the typically powerful multilayer perceptrons (MLPs) have difficulty in recognizing the small shifts in size, orientation, and location that are possible with images. However, these MLPs can be modified to be better suited for image classification.

A typical MLP consists of an input layer that takes in the input data vector followed by a series of hidden layers, ending with an output layer with one node for each classification category. Each hidden layer receives data from the previous layer transformed by a set of learnable weights, then further transforms that data using a non-linear activation function and outputs that data to the next layer. In this way, the MLP essentially learns a function for each node of the output that maps the input data vector to a probability that the input represents an item of that node's category. Important to note is that each hidden layer is fully connected to the previous layer, meaning that every value from the previous layer is taken into account for each node in the layer. This means that the number of weights to be learned is very large, making each additional layer take up enormous amounts of memory.

Convolutional Neural Networks (CNN) are a modification of MLPs that specialize them for the task of image classification. They are based on the biological concept of visual perceptive fields: study in the 1960s found that in animal visual cortexes, individual neurons responded to only small parts of the full visual field. This concept was applied to MLPs to develop the CNN. In a CNN, each hidden layer is replaced by a combination of two types of layers, a convolutional layer and a pooling layer.

The convolutional layer consists of a series of kernels, each of which is a small, square set of weights. The output of the layer is calculated by cross-correlating each kernel with the input to the layer, thus forming an output vector of the same size as the input for each kernel. Since each cell of the output is obtained from the cross-correlation the value of each cell is dependent only on a small area of the input, thus simulating the concept of the biological receptive field. The convolutional layer can be thought of as representing a set of learnable 2D filters, so cascading multiple convolutional layers is akin to applying a series of filters to an input image. The structure of the convolutional layer has two main advantages. One, since each kernel is shared across the entire input, the network is less susceptible to translational shifts in the input image. Second, the number of weights to be learned for each convolutional layer is significantly smaller than a conventional fully connected hidden layer. This helps to cut down on memory costs, while still providing classification at reasonable accuracy.

However, since each kernel produces output data of the same size as the input, the size of the data being put through the CNN rises quickly. For example, if a 100x100 image is input to a layer with 8 kernels of size 3x3, the output of the layer will be 8 100x100 data vectors. Therefore, some method to control the size of the data must be implemented. This is done with pooling layers. These layers down-sample the data by collapsing square regions into single values by choosing the maximal value in that region. This allows the network to preserve large values, while condensing data produced by each layer to manageable sizes.

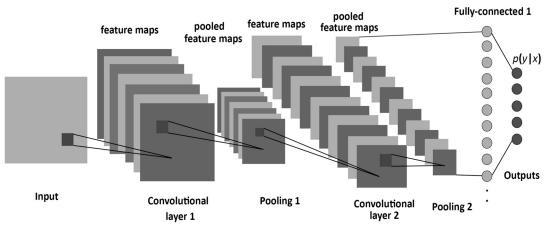


Fig. 1 – Graphical Representation of Convolutional Layers and Pooling

II. Motivation and Applications

A. Automated Set-Theoretical Computation

Mathematical computational engines like WolframAlpha and Mathematica have undeniably pervasive uses in engineering and data science. However, formulating set theory is NP-hard, blocking efficient implementations of set theoretical computers. This project introduces a particular and reduced set theoretic approach in automating the logic and computation of set theory, in the form of a contradiction analyzer.

B. Computer Vision and Augmented Reality

Computer vision (CV) and augmented reality (AR) technology have changed the way humans and technology interact with the environment. Computer vision permits technology to retrieve visual information in increasingly precise and intelligent ways, while augmented reality introduces the capability to process and output "augmentative" information as a function of the input and environment in real-time to improve human capabilities. In particular, the STCA is a software that utilizes computer vision to interpret a wide variety of set-theoretic syntax and automatically analyze the logic of set-theoretical structures via interactive controls that resemble the way humans interpret/analyze reality in real-time. For instance, any engineering or mathematics student researching or attending a conference on topics that utilize set theory can point their mobile devices equipped with an STCA to an image (written, displayed, or digital) containing set-theoretical

structures and expediently analyze the set-theoretic content of the information in the input image. We hope to realize such applications of the STCA in the near-future as edge/cloud or quantum computers break classical computational limits and constraints, regardless of the fact that set theory is NP-hard.

C. Academic Motivation

Academically, the STCA project introduces design problems in DSP and software algorithms that are difficult to solve. Precise computer vision techniques demand high-dimensional convolutional neural networks that are memory-intensive and computationally-inefficient to train and execute. Moreover, extracting necessary features within images is no easy feat, requiring careful consideration of feature characteristics, image recognition assumptions/trade-offs, and the medium or environment that supports/surrounds the image. Augmented reality succeeds concepts in computer vision, extending the application of CV to real-time and challenging us to constrain the latency of our hardware and software. Though not exposited in this paper, the translation/reduction and analysis of set theory in terms of satisfiability (SAT) is complex, involving encoding maps and recursive/refinement search algorithms that run in factorial $\mathcal{O}(n!)$ or super-exponential $\mathcal{O}(n^n)$ time. Constructing a robust automaton – the STCA – to analyze a set theoretical structure for contradiction is a computer-aided verification (CAV) problem that has extensive applications in mathematics, information theory, large-network analysis, and control.

III. Implementation

A. System Overview

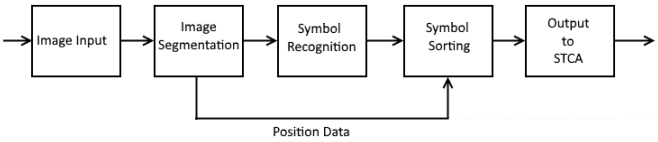


Fig. 2 – Block Diagram of the System

The system as implemented can be broken down into three different sections, the image processing and segmentation section, the character recognition section, the satisfiability (SAT) translation, and the STCA. In the image processing section, the image was segmented into different characters and resized for input to the classification system. The positions of each character were recorded as well. In the classification section, the images of each character were passed through the CNN and output in an intermediate symbol encoding. In the satisfiability translation section, the recognized symbols were sorted using the positions recorded from the processing section and translated to the SAT format of the STCA algorithm.

The input to the system was assumed to be image data of set-theoretical statements, a vector of intensity values ranging from 0-255. Higher intensities (white light) were assumed to be higher values (nearer 255) and lower intensities were assumed to be nearer 0. The image was also assumed to be in grayscale, with no additional color information. The width and height of the image were also assumed to be provided. The output of the system is a Boolean that reports if there exists a set-theoretic contradiction in the input image, and a translation of the relatively-finite set-theoretic statements to SAT.

B. Image Retrieval, Segmentation, and Processing

To retrieve and process the image, the embedded system is equipped with an input from a 1 MP camera (approximate image resolution of 1280×768 pixels) that captures an image and sends the RGB image pixel matrix to the DSP module on the embedded system. To reduce the computational complexity in processing the image, the RG in RGB are deleted from the pixel matrix, because the DSP software detects/utilizes the shape/form of the set-theoretic symbols to classify the set theory syntax necessary for the STCA.

To extract the symbols/features in the image to classify, there exist many image segmentation and clustering algorithms, like principal component analysis (PCA), k-means clustering, k-nearest neighbors clustering, the generalized Hough transform, density-based heat distributions, kernel methods, spectral clustering, and particle filtering. One type of clustering method is region-growing, initializing a collection of similar-characteristic regions in an image and combining regions with sufficient feature intersection. In particular, we propose a simplified region-growing feature extraction algorithm that extends a square cover over features in the image via proximity of the defining characteristics of the feature.

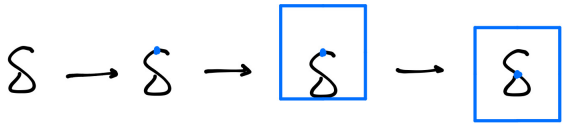


Fig. 3 – Square-Extension Feature Extraction Algorithm

1) Image Thresholding: Apply a pixel filter that classifies the pixels of an image as a feature pixel or irrelevant, i.e. if a pixel is a component of a feature in the image or not a component of a feature in the image. In particular, to extract dark set-theoretical symbols over a white background, we applied a pixel saturation threshold (fixed at 128) to differentiate between set theory syntax and blank space.

2) Feature Search, Centering, and Extraction: Taking the pixels that represent features in the image, it suffices to segment/cluster the pixels into groups corresponding to individual features/symbols and extract the pixel clusters to segment the features of the image. Looping through all feature-relevant pixels in the image matrix, the square-extension and extraction algorithm constructs and extends the dimensions of a square pixel cover centered at a feature pixel. Tuned to a particular image resolution, the square pixel cover extension terminates if proximal/boundary pixels to the square cover are consistently irrelevant or blank/null. Subsequently, the algorithm computes the average position of all the pixels contained in the square cover and re-centers the square cover to the pixel whose coordinates are approximately equivalent to the average pixel position coordinates. Extracting the square cover of the feature in the image to the output buffer and deleting the feature in the processed image, the feature extraction algorithm continues to search for feature pixels, terminating when no more feature pixels (or features) exist in the processed image.

3) Noise Filtering: Consequently, there exist noisy pixels that persist through the image feature thresholding filter and are mistaken in the feature extraction algorithm as isolated/sparse features. To counteract and filter the noise, we inject a secondary filter that measures and thresholds the dimensions of the square cover of the features. Extracted features in the image with (small) dimensions below the threshold are interpreted as noise and deleted along with their position and characteristics, while sufficiently large/non-trivial features are sent to the CNN stage for STCA classification.

4) Set Theoretical Statement Reconstruction: Via the square-extension feature extraction algorithm, the set-theoretical symbols extracted from the image and classified by the CNN have corresponding positions (x, y) determined by the average coordinates of the feature pixels in the square cover of the extracted feature. In order to translate the set theoretic symbols to set-theoretical SAT, we have to sort/arrange and process the symbols into set-theoretical statements equivalent to the set-theoretical statements displayed in the input image. To reconstruct the set-theoretical statements, the sorting algorithm searches for the approximate top-left positioned symbol extracted from the image. It initially prioritizes the minimal feature y-coordinate modulo a distance interval of uncertainty determined by the average of the dimensions of the current and previous processed symbols in the image. Note that the compared features are within the y-interval of uncertainty if and only if the symbol features are approximately positioned on the same row in the image if and only if the symbols belong to a particular set-theoretical statement, in which case the sorting algorithm assigns secondary priority to the minimal feature x-coordinate to order the symbols in the set-theoretical statement. The ordered sequence/buffer of CNN-classified set-theoretical symbols (with 0 separating set-theoretical statements) is passed to the SAT translator.

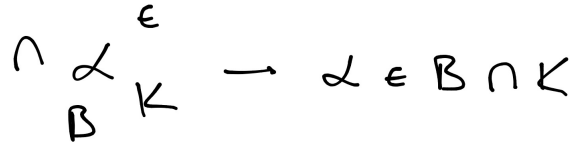


Fig. 4 – Set Theory Reconstruction via Positional Sorting

C. Image Compression for CNN

The data provided by the image segmentation algorithm is a set of vectors of image intensity values from 0-255, each representing a square surrounding a single symbol. This set of vectors is put into a single long vector, delimited by -1 values. Each symbol's vector is of varying size, since the image segmentation algorithm dynamically creates bounding boxes for the characters. To reorganize the input data into a format to be read by the CNN, symbols are sequentially resized and read into a separate array of size 10,000 (a 100x100 image).

The vector for each symbol is read sequentially, using the delimiters to detect when the end of a symbol's vector is reached. The length of the vector is recorded as well. Using the length, and the knowledge that each vector represents a square image, the vector is resized using the simple Nearest Neighbor image resizing algorithm. For upsampling, this algorithm creates new values at a particular location in the image by cloning the data values nearest that location in the smaller image. For downsampling, the coordinates of pixels in the new, smaller image are scaled to the larger image, then rounded to integer values. The image value in the rounded location is then taken as the value of the downsampled image in that location. This algorithm does not smoothly interpolate in the case of upsampling, but since the intensity values are thresholded and are all either 0 or 255 already, this is deemed acceptable. The upsampled or downsampled image is put into the 100x100 array, ready for the CNN.

D. Convolutional Neural Net Topology

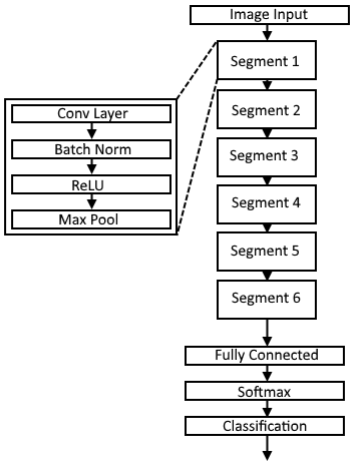


Fig. 5 – Overview of CNN Architecture

The input to the CNN is the resized image data from the pre-processing segment. Each time the CNN is run, it is run on a 100x100 array of data corresponding to a single symbol. The data is put through 27 different layers, though most of these layers are simple operations. They are divided into 7 types, an initial normalization layer, the convolutional layer, the batch normalization layer, the Rectified Linear Unit (ReLU) layer, the max pooling layer, a fully connected layer, and a softmax application layer.

The initial layer of the CNN normalizes the input data. A 100x100 array of learned normalization factors are subtracted from the input data. This operation is performed using the TI DSPLIB weighted vector sum function, making use of the LCDK's optimized implementation of vector sums.

The convolutional layer is the layer in which square kernels are cross-correlated with the input data. For each input image to the layer (initially there is only one), this layer applies the 2D cross-correlation operation using each of its learned kernels. In the case that there are multiple input images to the layer, which is true for every layer after the first, it tunes separate weights for each input image as well, integrating the results from each set of weights for a given kernel. For example, in convolutional layer 2 (layer 6), the previous convolutional layer generated 8 output images from its 8 kernels. These 8 images propagated through the next 3 layers, so 8 images of size 50x50 are input to convolutional layer 2. This

layer is set to have 16 kernels, so it must output 16 transformed images. For kernel 1, there are 8 3x3 squares of weights, one for each input image. Each square of weights is cross-correlated with the appropriate input image to produce 8 50x50 intermediate images, then these are summed to produce the output image for kernel 1. This is repeated for the remaining 15 kernels. In its current state of implementation, there are no optimizations in place for this layer type. The planned implementation of optimization using DSPLIb is discussed in the discussion section below.

The batch normalization layer is a layer that normalizes the output of the convolutional layer so that the data values do not grow out of control. This layer performs a simple linear operation using learned weights, shown below.

The ReLU layer performs another simple operation on the data. It looks for any negative entries and sets them to 0, leaving other values unchanged. This serves as an approximation of the softplus activation function, $f(x) = \ln(1 + x^x)$, as shown below. It takes very few resources to apply.

The max pooling layer reduces the size of each input image by a factor of 2 in each dimension by taking the maximum value in each 2x2 square of the image and using it as a single pixel value. This keeps the amount of data being propagated through the network under control by dividing it in 4 before each convolutional layer.

The fully connected layer is a typical layer as used in simpler artificial neural networks. It flattens the data from the previous layer into a single vector, applies the weights corresponding to each output node, and adds a bias. The number of output nodes is determined by the number of classes that the network needs to recognize. In this implementation, there are 38 output nodes for the 38 characters to be recognized. This implementation is optimized using the TI DSPLIB functions for matrix multiplication and vector sums; the weights are applied using the matrix multiplication function and the bias is applied using the weighted vector sum function with a weight of 1. The result is a 38 element vector.

The soft-max normalization layer applies the *soft-max function* to the output from the fully connected layer. This places all of the output node values between 0 and 1, and they represent the probability that the image input is an element of that class of symbol.

$$f(x)_i = \frac{e^{x^T w_i}}{\Sigma_{k=1}^{N} e^{x^T w_k}}$$

Eq. 1 – Soft-Max Normalization Function

E. Satisfiability Translation for Relatively-Finite Set Theory

To translate a collection of relatively-finite set theoretical statements to SAT, it suffices to construct a sufficient collection of atomic SAT propositions that combine to represent all possible set-theoretic statements consisting of elements, sets, element inclusion $\in$, set containment $\subset$, set union $\cup$, and set intersection $\cap$ in ZFC set theory. Indeed, suppose that there exist relatively-finitely many elements $\mathfrak{E} = \{\alpha, \beta, \ldots, \zeta\}$ and relatively-finitely many sets $\mathfrak{S} = \{A, B, \ldots, Z\}$ in a mathematical structure, argument, or proof. Define the SAT encoding Σ that maps an element γ and set T to a unique integer in $\mathbb{Z}$.

$$\Sigma \equiv \begin{array}{c|cccc} & \text{A} & \text{B} & \cdots & \text{Z} \\ \hline \alpha & 1 & 2 & \cdots & 26 \\ \beta & 27 & 28 & \cdots & 52 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \zeta & 651 & 652 & \cdots & 676 \end{array} : \mathfrak{E} \times \mathfrak{S} \to \mathbb{Z}$$

Table 1 – Set-Theoretic Satisfiability Encoding Map

To utilize the mapping, we interpret the integers in **Table 1** as atomic SAT propositions ϕ, i.e. $\alpha \in B$ is represented in SAT as the integer $2 \in \mathbb{Z}$ and $\zeta \notin Z$ is represented in SAT as the integer $-676 \in \mathbb{Z}$. Extending the encoding of the SAT translation, it follows that $\left[\tilde{\Sigma} : \mathfrak{E} \times \mathfrak{S} \times \{\pm 1\} \to \mathbb{Z}\right] \equiv \pm\Sigma$ is in bijective correspondence to the SAT translation $\tilde{\Sigma} : \mathfrak{E} \times \mathfrak{S} \times \{\in, \notin\} \to \text{SAT}$ defined via the map $(\gamma, T, \{\in, \notin\}) \mapsto \{\gamma \square T \mid \gamma \in \mathfrak{E}, T \in \mathfrak{S}, \square \in \{\in, \notin\}\}$ in SAT. Applying Boolean algebra and DeMorgan's Laws, we can convert any relatively-finite set-theoretical statement to a sequence of atomic SAT propositions in conjunctive normal form (CNF), in which we represent logical AND as 0 and logical OR as $\varnothing$ in the encoded SAT sequence.

$$\alpha \in A \cap B \iff \{\alpha \in A\} \wedge \{\alpha \in B\} \xrightarrow{\Sigma} \ldots, 0, 1, 0, 2, 0, \ldots$$
$$\beta \notin A - Z \iff \{(\beta \notin A) \vee (\beta \in Z)\} \xrightarrow{\Sigma} \ldots, 0, -27, 52, 0 \ldots$$
$$\cdots \wedge \phi \wedge (\psi \vee \zeta) \wedge \cdots \mapsto \ldots, 0, \Sigma(\phi), 0, \Sigma(\psi), \Sigma(\zeta), 0, \ldots$$

To translate set relations like $A \subset B$, it is necessary to process the assumed information related to A and B. That is, the implication $\alpha \in A \implies \alpha \in B$ and contrapositive $\alpha \notin B \implies \alpha \notin A$ are introduced to the encoded SAT sequence via a persistent and dynamic SAT set relation filtering algorithm that injects or deletes necessary atomic SAT propositions/constraints determined by the aforementioned implications for all elements α satisfying $\alpha \in A$ or $\alpha \notin B$. Element relations precede set relations to characterize the content of all sets prior to the SAT translation of relatively-finite set relations $\{\subset, \supset, \cup, \cap, =, \emptyset, \ldots\}$. Set relations are implemented via set translation procedures in accordance to the implications of set theoretic relations on non-disjunctive atomic propositions ϕ in the encoded SAT sequence dynamically, i.e. if the disjunctive reduction step of the STCA algorithm refines a disjunctive term $\bigvee_j \phi_{kj}$ in $\bigwedge_i \bigvee_j \phi_{ij}$ to the atomic SAT proposition ϕ_k, we execute all set translation procedures correlated to the proposition ϕ_k to update the encoded SAT sequence relative to the set relations that depend on ϕ_k to preserve logical satisfiability. Such a filter refines the encoded SAT sequence, removing all possible contradictions to the introduced implications or atomic propositions in the sequence, and is the basis of an algorithm that dynamically searches for contradiction in the encoded SAT sequence as the sequence is constructed.

F. Set-Theoretical Contradiction Analyzer (STCA)

Utilizing the SAT translator, we propose a programmable recursive algorithm that dynamically updates, refines, and analyzes an encoded SAT sequence of atomic SAT propositions in CNF to search for logical set-theoretic contradictions – the Set-Theoretical Contradiction Analyzer (STCA).

Suppose there exists an encoded SAT sequence $s_n \subset \mathbb{Z}$ that is constructed via SAT translation of a relatively-finite set-theoretical mathematical structure/model. Without loss of generality, assume that the sequence $s_n \subset \mathbb{Z}$ is CNF, taking the form $s_n \cong \bigwedge_{i \in I} \bigvee_{j \in J} \phi_{ij}$. Note that if any disjunction term $\bigvee_{j \in J} \phi_{ij}$ evaluates as FALSE, it follows that $\bigwedge_{i \in I} \bigvee_{j \in J} \phi_{ij}$ evaluates as FALSE. A set-theoretical contradiction exists if and only if there exists a collection of atomic SAT propositions ϕ_{ij} represented in s_n that contradicts some disjunctive term $\bigvee_{j \in J} \phi_{ij}$ represented in s_n. To search for contradictions, it suffices to refine the encoded SAT sequence for all injected disjunctive terms $\bigvee_{j \in J} \phi_{ij}$ as the sequence of atomic propositions is constructed. In particular, assume that the disjunctive term $\bigvee \psi_j$ is translated via Σ and appended to s_n. The STCA algorithm executes on the subsequence $s_n - \Sigma\left(\bigvee \psi_j\right)$.

1) Disjunctive Reduction: Assume that ψ_j is TRUE for all ψ_j in $\bigvee \psi_j$. Check if there exists a contradiction to ψ_j in s_n. If a contradiction is determined, delete ψ_j from $\bigvee \psi_j$, because ψ_j cannot be TRUE without inducing contradiction. If no contradiction is found, preserve the atomic SAT proposition ψ_j continue the disjunctive reduction. If there exists at least one atomic SAT proposition that remains in $\bigvee \psi_j$ post-disjunctive reduction, then the introduced SAT formula does not induce contradiction. Otherwise, if all atomic propositions in $\bigvee \psi_j$ induce contradiction, then $\bigvee \psi_j$ induces contradiction if TRUE. Terminate the STCA algorithm.

2) Recursive Contradiction Searching: To search for set-theoretical contradictions, assume that ψ_j in the introduced SAT formula $\bigvee \psi_j$ is TRUE and search for $-\Sigma(\psi_j)$, representing the logical negation of ψ_j via the encoded SAT translation defined previously, in the encoded SAT sequence $s_n \subset \mathbb{Z}$. Suppose that the negation $\Sigma(\gamma_k) = -\Sigma(\psi_j)$ exists in s_n, contained in a subsequence representing the disjunctive SAT formula $\bigvee_k \gamma_k \subset \Sigma^{-1}(s_n)$. Assume that γ_k is FALSE via the assumption that $\psi_j = \neg\gamma_k$ is TRUE. Delete $\Sigma(\gamma_k)$ from s_n and recursively call the contradiction search algorithm on all atomic propositions $\gamma_{l \neq k}$ in $\bigvee_{l \neq k} \gamma_l$. If γ_l induces contradiction, delete $\Sigma(\gamma_l)$ from s_n. Otherwise, continue the STCA algorithm. If all atomic propositions γ_k induce contradiction, return CONTRADICTION. Otherwise, there exists at least one atomic proposition γ_l that does not induce contradiction in the SAT sequence if and only if $\bigvee_k \gamma_k$ and ψ_j can simultaneously be TRUE if and only if no contradiction exists. Continue the recursive contradiction search algorithm in the STCA.

An implementation of the recursive contradiction search algorithm in the STCA is described in pseudo-code (**Alg. 1**) and is compatible/programmable in C. The algorithm takes as input the encoded atomic SAT proposition $\Sigma(\psi_j)$ that is assumed TRUE and performs local operations on a duplicate of the encoded SAT sequence s_n. In particular, the computational complexity of the STCA algorithm is undefined in closed-form, because the algorithm dynamically refines/reduces the encoded SAT sequence during the contradiction analysis. Taking an upper bound, the algorithm executes in factorial $\mathcal{O}(n!)$ or super-exponential $\mathcal{O}(n^n)$ time with n defined as the length of the encoded SAT sequence. Note that these are *weak* upper bounds on computational complexity, because the STCA algorithm aggressively refines/reduces the set-theoretical SAT sequence during contradiction analysis. Research in information theory pertaining to representation of set theory in SAT is necessary to improve these complexity bounds.

Algorithm 1: DPLL Algorithm - STCA Recursive Contradiction Search

Input: SAT-Encoded Integer Array σ
Output: Returns contradiction (False) or lack of contradiction (True).
if *all disjuctive clauses contain at least one literal assigned to True in* σ **then**
  return True
else if *there exists a disjunctive clause in which all literals are False in* σ **then**
  return False
while *there exist any disjunctive clauses* $\Delta \subset \sigma$ *such that all assigned literals contained in* Δ *are False except for precisely one unassigned literal* $\alpha \in \Delta$ **do**
  Universally assign the literal α to True and $-\alpha$ to False throughout σ.
Choose any unassigned literal λ in σ.
return DPLL-STCA($\sigma \wedge \lambda$) $\vee$ DPLL-STCA($\sigma \wedge \neg\lambda$)

IV. Conclusion

A. Results

On a test image containing 13 set-theoretic symbols placed in 3 different rows, the image segmentation algorithm managed to separate all 13 characters and record their corresponding position/coordinates. When classified using the CNN, 10 out of 13 symbols were recognized correctly. Of the mis-classified symbols, 2 out of the 3 were recognized as significantly visually similar symbols (D mistaken as set containment $\supset$, and G mistaken as set inclusion $\subset$). In the output of the CNN, the symbols were sorted with perfect accuracy, matching the order that they were written/displayed in the input image, and output in the digital reconstructed format for the STCA without any problems.

The execution time of the system was rather slow, at a few seconds per character for a total execution time of approximately one minute. The primary bottleneck was the unoptimized convolutional layer implementation of the CNN, specifically the first convolutional layer activation for each processed symbol/feature in the image. Such layer executions accounted for almost the entirety of the execution time.

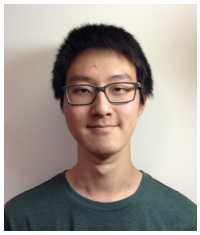
B. Discussion and Future Work

The optimization of the convolutional layer using DSPLIB would have drastically accelerated the bottleneck CNN component of the system. The plan for optimization was to use the DSPLIB FIR filter function to apply each layer's kernels efficiently. However, the DSPLIB functions demanded additional restrictions, namely extra zero padding for the input data and data arrays sized to multiples of 4 via zero padding. There exist two zero-padding possibilities, either use the memory block move functions to store each row of an input image in a temporary array with zero padding, execute the function,

and copy the data back, or integrate zero padding into the storage of the images. In particular, zero-padding injects zeros into the front and end of each row, giving the DSP function a pointer to the correct location in the array to perform the cross-correlation. Each row in the output image is formed by using DSPLIB vector addition to add the results of 3 such cross-correlations together, along with a bias term. The padding entries would have to be zeroed out after each convolution application to maintain the correct format for the next cross-correlation operation. However, this would likely be much faster than the manual implementation of the convolution, which multiplied and added each term manually. The logic for this optimization was implemented, but it was not completely correct and debugging the system proved very time intensive due to the large number of intermediate values to be analyzed.

Moreover, the STCA algorithm is far from optimized. Refinement is a clever method to dynamically reduce the complexity of the algorithm in run-time, but there does not exist intelligent programming to determine which set-theoretical statements translate to computational-analytical bottlenecks in the STCA. In particular, we did not have the time to design a combination of the SAT translator and the STCA. If combined, we can effectively *pipeline* the STCA input, i.e. for each reconstructed set-theoretical statement, we simultaneously apply the SAT translation and STCA recursive algorithm to filter the encoded SAT sequence. Such a filter manages to constrain the cardinality of the encoded SAT sequence, drastically reducing the computational latency of the STCA to precisely $\mathcal{O}(n!)$, as the STCA recursion executes in approximately $n!$ iterations and there exists a constant/finite number k of set-theoretical statements in the input image via hardware constraints in an embedded system.

Last, but not least, we consider the ramifications of our system in augmented reality (AR). Considering the breadth of the project, we lacked the time to implement our system in a real-time embedded system. That is, we were unable to connect a low-resolution, high frame-rate camera to take image input, nor were we able to program an image overlay software that detects and points out the precise set-theoretical statements in the image that induce contradiction. In terms of computer vision, the image segmentation algorithm is prone to any type of connected or organized noise (curves, shapes, feature mixtures, occlusion, etc.) and is built on many assumptions pertaining to depth, spacing, and relative structure/grammar of the features/syntax in the input image. However, given more powerful technology, it would not be difficult to embed our STCA algorithms and CNN into existing AR technology equipped with optimized/advanced input sensors, image DSP, and human-computer interfaces compatible with the necessities of the STCA system if extended to AR applications.



Cory Ye Just some geek.

Andrew Deng Just some nerd.