Improving the Reliability and Safety of Systems
Toward Scalable Deep Neural Network Verification

ThanhVu (Vu) Nguyen

CEC P&T Seminar, Nov 12 2023
My Background

Academic

- 2013: PhD in CS, Univ of New Mexico–Albuquerque
- 2014: Postdoc, Univ of Maryland-College Park
- 2016: Assistant Prof., Univ of Nebraska-Lincoln
- 2021: Assistant Prof., George Mason University

Govt and Industry

- 2007: Lockheed Martin, New Jersey
My Research

Software Engineering, Formal Methods, Programming Languages

- **Invariant Generation and Automatic Program Repair**
  - since ’08, during PhD study

- **Highly-Configurable and Build System Analysis**
  - since ’15, during postdoc

- **AI Verification**
  - since ’22, new research direction
My Research

Software Engineering, Formal Methods, Programming Languages

- **Invariant Generation and Automatic Program Repair**
  - since ’08, during PhD study
- **Highly-Configurable and Build System Analysis**
  - since ’15, during postdoc
- **AI Verification**
  - since ’22, new research direction

Sponsor

- NSF (4x): CRII’20, Med Collab. ’21, CAREER’23, FMIT’23
- Defense (1x): Army Research ’18
- Industry (2x): Facebook’23 and Amazon’23
- Internal (1x): UNL Seed’20
DynaROARS
dynaroars.cs.gmu.edu

Didier  Linhan  Hai  Guolong
PhD’22 at UNL

TENURE FOR DADDY!
Outline

AI Safety Verification
Highly Configurable and Build Systems
Invariant Generation and Program Repair
DNN EVERYWHERE
DNN Problems

Amazon Rekognition FALSE MATCHES

28 current members of Congress
Black person with hand-held thermometer = firearm. Asian person with hand-held thermometer = electronic device.

Computer vision is so utterly broken it should probably be started over from scratch.
GOOGLE SELF-DRIVING CAR GETS INTO AN ACCIDENT INVOLVING INJURIES

#NEWS

GOOGLE SELF DRIVING CAR CRASHES INTO A BUS

EXCLUSIVE INVESTIGATION FOCUSED ON TESLA AUTOPILOT

DEADLY CRASH WITH SELF-DRIVING UBER

ARIZONA 11:01 64° TAKING ACTION DRIVERLESS UBER CAR INVOLVED IN CRASH IN TEMPE POLICE SAY OTHER DRIVER FAILED TO YIELD
∀i ∈ {0...|X| − 1}. X_i − Y_i ≤ 0.1 ⇒ class(X) ≡ class(Y) (1)
Robustness Properties

\[ \forall i \in \{0 \ldots |X| - 1\}. \quad X_i - Y_i \leq 0.1 \Rightarrow \text{class}(X) \equiv \text{class}(Y) \quad (1) \]

if corresponding pixels of two images \( X \) and \( Y \) are not different by more than 0.1, then \( X \) and \( Y \) should have the same classification
Safety Properties

ACAS X: Whole Airspace Protection

If intruder is distant and significantly slower than us, then we do nothing (i.e., below a certain threshold)
Safety Properties

**ACAS**: air traffic collision system, detects intruder and decides action.

\[
d_{\text{intru}} \geq 55947 \land v_{\text{own}} \geq 1145 \land v_{\text{intru}} \leq 60 \Rightarrow r_{\text{nothing}} \leq \tau
\]

*if intruder is distant and significantly slower than us, then we do nothing (i.e., below a certain threshold)*
Well-trained, e.g., 97% accuracy, DNNs are fine for most tasks

- But not enough for mission-critical tasks, e.g., self-driving cars, air traffic collision control

Testing can find counterexamples (e.g., adversarial attacks)

- Testing shows the existence of errors, not its absence (Dijkstra)
Well-trained, e.g., 97% accuracy, DNNs are fine for most tasks
  - But not enough for mission-critical tasks, e.g., self-driving cars, air traffic collision control
Testing can find counterexamples (e.g., adversarial attacks)
  - Testing shows the existence of errors, not its absence (Dijkstra)

Formal Verification Can Help!
Software Verification

- Provide formal guarantee that a system really has no specific type of errors
- Mature field in CS/Logics with lots of powerful techniques and tools
  - Automated Theorem Proving
  - Constraint Solving (e.g., SAT/SMT solving)
  - Model Checking
  - Abstract Interpretation, ...
- Employed in mission-critical systems, e.g., avionics, medical devices, Windows, Clouds system (AWS)
The problem of Deep Neural Network verification

**Question:** Given a network $N$ and a property $p$, does $N$ have $p$?

- $p$ often has the form $P \Rightarrow Q$ (precondition $P$, postcondition $Q$)

**Answer:** Yes / No
The problem of Deep Neural Network verification

Question: Given a network $N$ and a property $p$, does $N$ have $p$?

- $p$ often has the form $P \Rightarrow Q$ (precondition $P$, postcondition $Q$)

Answer: Yes / No

Simple DNN with ReLU

![Diagram of a simple DNN with ReLU]

- E.g., $x_3 = \max(-1x_1 - 0.5x_2, 0)$
The problem of Deep Neural Network verification

**Question**: Given a network $N$ and a property $p$, does $N$ have $p$?

- $p$ often has the form $P \Rightarrow Q$ (precondition $P$, postcondition $Q$)

**Answer**: Yes / No

**Simple DNN with ReLU**

\[
\begin{align*}
    x_1 & \to -1.0 & x_3 \\
    x_2 & \to -0.5 \to 1.0 & x_4 \\
    x_3 & \to -1.0 \to 1.0 & x_5 \\
    x_4 & \to -1.0 \to 1.0 & x_5
\end{align*}
\]

- E.g., $x_3 = \max(-1x_1 - 0.5x_2, 0)$
- Valid: $x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 \leq 0$
- Invalid: $x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 > 0$
Constraint Solving Techniques

Verification Query

Input Space | Neural Network | Output Space
--- | --- | ---
$P$ | $\begin{array}{c}
x_1 \\
x_2 \\
x_3 \\
x_4 \\ \end{array}$ | $\begin{array}{c}
y_1 \\
y_2 \\
y_3 \\ \end{array}$ |

SAT (+ counter example) | UNSAT

Verification

SAT: is a property of
UNSAT: is not a property of

$N$ is the number of neurons

Solve the constraint, e.g., using MILP solvers
Transform DNN verification into a constraint (satisfiability) problem

Figure 1: The neural network verification process.
Transform DNN verification into a constraint (satisfiability) problem

- **UNSAT**: \( p \) is a property of \( N \)
- **SAT**: \( p \) is not a property of \( N \) (also provide counterexamples)
- **TIMEOUT**
To illustrate the promise of utilizing deep RL for system design, consider the following scenarios:

- **Scenarios**
  - **Scenario 1:** A DNN classification approach (and, following the rise in popularity of DNNs, the verification community has sought approaches for quality assurance, such as testing and simulation tools. Despite this, verification technology is rapidly improving and, as demonstrated by us and by others, existing tools can verify certain systems and apply Verily to determine whether these are "satisfiable" (similar to acceptance tests for traditional software), and, if not, generate counter examples. Our preliminary evaluation results expose several problems in the tested verification schemes. We formulate natural requirements for each of these systems and apply Verily to determine whether these are satisfiable. Importantly, when Verily determines that the system does not meet the requirements, it can provide formal guarantees about the behavior of the system for inputs that were not part of its training or validation.

- **Scenarios 2 and 3:** As captured by a reward function that represents the client's playback buffer occupancy and the download rate, Pensieve learns to map observables to bitrate selections, thus enabling different bitrate selections for performance. The agent learns to produce for input $x_1$, $x_2$, $x_3$, $x_4$. The parameter $\gamma$ is the discount factor, and the reward $r_t$ is the immediate reward received after selecting an action $a_t$. The discounted return $R_t$ is the sum of all future rewards discounted by the factor $\gamma$.

- **Constraint Solving Techniques**
  - **Transform DNN verification into a constraint (satisfiability) problem**
    - **UNSAT**: $p$ is a property of $N$
    - **SAT**: $p$ is not a property of $N$ (also provide counterexamples)
    - **TIMEOUT**
  - **Solve the constraint, e.g., using MILP solvers**
  - **Scalability** is a **Huge** problem (many TIMEOUTs)
    - Complexity $O(2^N)$, where $N$ is the number of neurons
Abstraction Techniques

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval, zonotopes, polytopes

![Diagrams of Interval, Zonotope, Polytope]
Abstraction Techniques

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval, zonotopes, polytopes

- Scale well, but loose precision (producing spurious cex’s)
  - Claiming a property is violated when it is not
NeuralSAT: Our DNN Constraint Solver

To prove $N \Rightarrow (P \Rightarrow Q)$

- Call NeuralSAT($N \land P \land \neg Q$)
- Return UNSAT or SAT (and counterexample)

1. Abstract as a boolean satisfiability problem
2. Iteratively search for satisfying assignment
   - Use heuristics to make decision
   - Use propagation to communicate learned information
   - Analyze conflicts, learn conflict information, and backtrack
   - Use a theory solver to quickly deduce unsatisfiability (UNSAT)
Example: Simple DNN with ReLU activation

To prove $f : x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 \leq 0$:

- Use NeuralSAT to check if $\neg f$ is satisfiable
- NeuralSAT($\neg N \land x_1 \in [-1, 1] \land x_2 \in [-2, 2] \land x_5 > 0$)
- NeuralSAT returns UNSAT, indicating $f$ is valid
Boolean Abstraction

- Create 2 boolean variables $v_3$ and $v_4$ to represent activation status of $x_3$, $x_4$
  - $v_3 = T$ means $x_3$ is active, $-x_1 - 0.5x_2 - 1 > 0$

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
Boolean Abstraction

- Create 2 boolean variables $v_3$ and $v_4$ to represent activation status of $x_3$, $x_4$
  - $v_3 = T$ means $x_3$ is active, $-x_1 - 0.5x_2 - 1 > 0$
- Form two clauses $\{v_3 \lor \overline{v_3} ; v_4 \lor \overline{v_4}\}$
- Find boolean values for $v_3$, $v_4$ that satisfies the clauses and their implications

$x_1 \in [-1, 1]$, $x_2 \in [-2, 2]$, $x_5 > 0$
Iteration 1

- Use **abstraction** to approximate upperbound $x_5 \leq 0.55$ (from $x_1 \in [-1, 1], x_2 \in [-2, 2]$)
Iteration 1

- Use **abstraction** to approximate upperbound $x_5 \leq 0.55$ (from $x_1 \in [-1, 1], x_2 \in [-2, 2]$)
- Deduce $x_5 > 0$ might be feasible

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
Iteration 1

- Use abstraction to approximate upperbound $x_5 \leq 0.55$ (from $x_1 \in [-1, 1], x_2 \in [-2, 2]$)
- Deduce $x_5 > 0$ might be feasible
- Decide $v_3 = F$ (randomly)
  - new constraint $-x_1 - 0.5x_2 - 1 < 0$

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
Iteration 2

- **Approximate** upperbound \( x_5 \leq 0 \) (due to additional constraint from \( v_3 = F \))
- **Deduce** \( x_5 > 0 \) infeasible: **CONFLICT**

\[
x_1 \in [-1, 1], \ x_2 \in [-2, 2], \ x_5 > 0
\]
Iteration 2

- **Approximate** upperbound $x_5 \leq 0$ (due to additional constraint from $v_3 = F$)
- **Deduce** $x_5 > 0$ infeasible: CONFLICT
- **Analyze** conflict, **backtrack** and erase prev. decision $v_3 = F$
- **Learn** new clause $v_3$
  - $v_3$ will have to be $T$ in next iteration

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
**Iteration 3**

- **Decide** $v_3 = T$ (BCP, due to learned clause $v_3$)
  - new constraint $-x_1 - 0.5x_2 - 1 > 0$

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
Iteration 3

- **Decide** $v_3 = T$ (BCP, due to learned clause $v_3$)
  - new constraint $-x_1 - 0.5x_2 - 1 > 0$
- **Approximate** new upperbound for $x_5$
  (using additional constraint from $v_3 = T$)
- **Deduce** $x_5 > 0$ might be feasible
- **Decide** $v_4 = T$ (randomly)

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
After several iterations
- **Learn** clauses \{v_3, \overline{v_3} \lor v_4, \overline{v_3} \lor \overline{v_4}\}
- **Deduce** not possible to satisfy the clauses

\[
x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0
\]
After several iterations

- **Learn** clauses \( \{v_3, \overline{v_3} \lor v_4, \overline{v_3} \lor \overline{v_4} \} \)
- **Deduce** not possible to satisfy the clauses
- **Return** UNSAT

- Cannot find inputs satisfying \( x_1 \in [-1, 1], x_2 \in [-2, 2] \) that cause \( N \) to return \( x_5 > 0 \)
- Hence, \( x_5 \leq 0 \) holds (i.e., the original property is valid)

\[ x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0 \]
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Rank</th>
<th>Verifier</th>
<th>Score</th>
<th>Percent</th>
<th>Verify</th>
<th>Falsify</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACAS Xu (13K)</strong></td>
<td>1</td>
<td>NeuralSAT</td>
<td>1437</td>
<td>100.0%</td>
<td>139</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>nnenum</td>
<td>1437</td>
<td>100.0%</td>
<td>139</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>αβ-CROWN</td>
<td>1436</td>
<td>99.9%</td>
<td>139</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Marabou</td>
<td>1426</td>
<td>99.2%</td>
<td>138</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>MN-BaB</td>
<td>1097</td>
<td>76.3%</td>
<td>105</td>
<td>47</td>
</tr>
<tr>
<td><strong>MNISTFC (532K)</strong></td>
<td>1</td>
<td>αβ-CROWN</td>
<td>582</td>
<td>100.0%</td>
<td>56</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>NeuralSAT</td>
<td>573</td>
<td>98.5%</td>
<td>55</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>nnenum</td>
<td>403</td>
<td>69.2%</td>
<td>39</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>MN-BaB</td>
<td>370</td>
<td>63.6%</td>
<td>36</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Marabou</td>
<td>370</td>
<td>63.6%</td>
<td>35</td>
<td>20</td>
</tr>
<tr>
<td><strong>CIFAR2020 (2.5M)</strong></td>
<td>1</td>
<td>NeuralSAT</td>
<td>1533</td>
<td>100.0%</td>
<td>149</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>αβ-CROWN</td>
<td>1522</td>
<td>99.3%</td>
<td>148</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MN-BaB</td>
<td>1486</td>
<td>96.9%</td>
<td>145</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>nnenum</td>
<td>518</td>
<td>33.8%</td>
<td>50</td>
<td>18</td>
</tr>
<tr>
<td><strong>RESNET_AB (354K)</strong></td>
<td>1</td>
<td>NeuralSAT</td>
<td>513</td>
<td>100.0%</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>αβ-CROWN</td>
<td>513</td>
<td>100.0%</td>
<td>49</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MN-BaB</td>
<td>363</td>
<td>70.8%</td>
<td>34</td>
<td>23</td>
</tr>
<tr>
<td><strong>MNIST_GDVB (3M)</strong></td>
<td>1</td>
<td>NeuralSAT</td>
<td>480</td>
<td>100.0%</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>αβ-CROWN</td>
<td>400</td>
<td>83.3%</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MN-BaB</td>
<td>200</td>
<td>41.7%</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>1</td>
<td>NeuralSAT</td>
<td>4536</td>
<td>100.0%</td>
<td>440</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>αβ-CROWN</td>
<td>4453</td>
<td>98.2%</td>
<td>432</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MN-BaB</td>
<td>3516</td>
<td>77.5%</td>
<td>340</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>nnenum</td>
<td>2358</td>
<td>52.0%</td>
<td>228</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Marabou</td>
<td>1796</td>
<td>39.6%</td>
<td>173</td>
<td>66</td>
</tr>
</tbody>
</table>
Key Ideas

- Formalization of DNN verification
- Analyze, learn, and propagate information (significantly reduce search space)
- Dedicated DNN-specific theory solver (enable fast proving)
- *New approach; open doors to new research on heuristics, optimizations specific to DNNs*
Key Ideas

- Formalization of DNN verification
- Analyze, learn, and propagate information (significantly reduce search space)
- Dedicated DNN-specific theory solver (enable fast proving)
- *New approach; open doors to new research on heuristics, optimizations specific to DNNs*

Usability Features

- **Standard**: inputs (ONNX) and outputs (SAT/UNSAT/TIMEOUT)
- **Versatile**
  - Support Feedforward, Convolutional, Residual Networks
  - Support ReLU, Sigmoid, Tanh, Power, etc
- **Scale well** to large networks with millions of neurons
- **Active development & frequent Updates**
- **Fully automatic** (require little configurations from users)
Outline

AI Safety Verification
Highly Configurable and Build Systems
Invariant Generation and Program Repair
Modern software are highly-configurable

- Allow for customization and flexibility
- Can have misconfigurations (5th on OWASP most critical security risks)

**Challenge:** huge search space ($2^{13000}$ for Linux)

### Linux/Unix Build Systems

<table>
<thead>
<tr>
<th>Network device support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network core driver support</td>
</tr>
<tr>
<td>Bonding driver support</td>
</tr>
<tr>
<td>Dummy net driver support</td>
</tr>
<tr>
<td>EQL (serial line load balancing) support</td>
</tr>
<tr>
<td>Fibre Channel driver support</td>
</tr>
<tr>
<td>Intermediate Functional Block support</td>
</tr>
<tr>
<td>Ethernet team driver support</td>
</tr>
<tr>
<td>MAC-VLAN support</td>
</tr>
<tr>
<td>MAC-VLAN based tap driver</td>
</tr>
<tr>
<td>IP-VLAN support</td>
</tr>
<tr>
<td>Virtual eXtensible Local Area Network (VXLAN)</td>
</tr>
<tr>
<td>Generic Network Virtualization Encapsulation</td>
</tr>
<tr>
<td>GPRS Tunneling Protocol datapath (GTP-U)</td>
</tr>
<tr>
<td><strong>IEEE 802.1AE MAC-level encryption (MACsec)</strong></td>
</tr>
<tr>
<td>Network console logging support</td>
</tr>
<tr>
<td>Dynamic reconfiguration of logging targets</td>
</tr>
<tr>
<td>Universal TUN/TAP device driver support</td>
</tr>
<tr>
<td>Support for cross-endian vnet headers on little endian</td>
</tr>
<tr>
<td>Virtual ethernet pair device</td>
</tr>
<tr>
<td>Virtio network driver</td>
</tr>
<tr>
<td>Virtual netlink monitoring device</td>
</tr>
<tr>
<td>Virtual Routing and Forwarding (LITE)</td>
</tr>
<tr>
<td>Virtual vsoc monitoring device</td>
</tr>
<tr>
<td>ARCnet support</td>
</tr>
</tbody>
</table>

- **elect** - Exit - Help - Save
Modern software are highly-configurable

- Allow for customization and flexibility
- Can have misconfigurations (5th on OWASP most critical security risks)

Challenge: huge search space (2\textsuperscript{13000} for Linux)

Approach: use symbolic execution to compute path conditions mapping to built files

- # of files is very small
- Solve path conds to find build issues and misconfigurations
Outline

AI Safety Verification
Highly Configurable and Build Systems
Invariant Generation and Program Repair
def intdiv(x, y):
    q = 0
    r = x
    while r ≥ y:
        a = 1
        b = y
        while r ≥ 2b:
            a = 2a
            b = 2b
        r = r - b
        q = q + a
    return q

• Discover invariant properties at certain program locations

• Answer the question “what does this program do?”

• Approach: use template and dynamic analysis
Invariant Generation (DIG)

```
def intdiv(x, y):
    q = 0
    r = x
    while r >= y:
        a = 1
        b = y
        while r >= 2b:
            a = 2a
            b = 2b
        r = r - b
        q = q + a
    return q
```

- Discover invariant properties at certain program locations
- Answer the question “what does this program do?”
- Approach: use template and dynamic analysis

Program Repair (GenProg)

```
def intdiv(x, y):
    q = 0
    r = x

    while r != y:
        a = 1
        b = y
        while r >= 2b:
            a = 2a
            b = 2b
        r = r - b
        q = q - 2a

    return q
```

- Localize errors and modify code to fix bugs
- Approach: use dynamic and static analyses to identify, create, and validate patches
Awards and Impacts

AI Verification

- NSF CAREER (’23—’28)
- Amazon Research Award ’23
- featured in SIGBED
- ranked 4th in VNN-COMP ’23 (would be 1st now)
Awards and Impacts

AI Verification
- NSF CAREER ('23—'28)
- Amazon Research Award'23
- featured in SIGBED
- ranked 4th in VNN-COMP'23 (would be 1st now)

Highly-Configurable and Build System Analysis
- NSF CISE CRII '20
- NSF Formal Methods in the Field (FMiT) '23
- Meta/Facebook unrestricted gift
- Adoption: used internally at Meta Whatsapp to analyze build issues
Awards and Impacts

AI Verification
- NSF CAREER (’23—’28)
- Amazon Research Award’23
- featured in SIGBED
- ranked 4th in VNN-COMP’23 (would be 1st now)

Highly-Configurable and Build System Analysis
- NSF CISE CRII ’20
- NSF Formal Methods in the Field (FMiT) ’23
- Meta/Facebook unrestricted gift
- Adoption: used internally at Meta Whatsapp to analyze build issues

Invariant Generation and Automatic Program Repair
- 10-year ACM SIGSOFT/IEEE TCSE Most Influential Paper Award’19
- 10-year ACM SIGEVO Most Impact Award’19
- NSF Medium Collaborative grant ’21–’25
- Army Office of Research ’18–’21
- Adoption
  - SV-COMP included benchmarks created by DIG
  - GrammaTech integrated DIG in Mnemosyne
  - Facebook and GrammaTech used GenProg in multiple projects
Future Directions

Currently

- focuses on *existing* problems (robustness, safety)
- tested with *existing* benchmarks
Future Directions

Currently

- focuses on *existing* problems (robustness, safety)
- tested with *existing* benchmarks

Challenges & Opportunities

- **new** problems
  - what properties should AI/ML have? (e.g., fairness, privacy, security)
  - how to formally define such specifications?
- **new** benchmarks (e.g., real-world, industrial data)
- **new** analyses (e.g., automatic property inference and repair for NNs)
Funding

- 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (Co-PI), 2 industry (sole-PI), 1 internal (sole-PI)
  - Total $2.65M; my share $1.5M, as PI $1.3M
  - At GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M)
  - Young Faculty: NSF CRII’20, NSF CAREER’23, Amazon Research Award’23
Funding

- 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (Co-PI), 2 industry (sole-PI), 1 internal (sole-PI)
  - Total $2.65M; my share $1.5M, as PI $1.3M
  - At GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M)
  - Young Faculty: NSF CRII’20, NSF CAREER’23, Amazon Research Award’23

Publications

- 27 journals/confs. papers since ’16 (11 since joining GMU in ’21)
  - 20 papers with students (9 with undergrad)
- Google: 3617 citations (h-index 17 i10-index 24)
- SIGSOFT MIP paper award, SIGEVO Impact paper award
Funding

- 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (Co-PI), 2 industry (sole-PI), 1 internal (sole-PI)
  - Total $2.65M; my share $1.5M, as PI $1.3M
  - At GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M)
  - Young Faculty: NSF CRII’20, NSF CAREER’23, Amazon Research Award’23

Publications

- 27 journals/confs. papers since ’16 (11 since joining GMU in ’21)
  - 20 papers with students (9 with undergrad)
- Google: 3617 citations (h-index 17 i10-index 24)
- SIGSOFT MIP paper award, SIGEVO Impact paper award

Students Mentoring

- Current: 3 Ph.D RA’s, 2 undergrads
- Graduated (at UNL): 1 PhD, 2 Masters, 11 undergrads (2 Outstanding Undergrad Research Awards)
Funding

- 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (Co-PI), 2 industry (sole-PI), 1 internal (sole-PI)
  - Total $2.65M; my share $1.5M, as PI $1.3M
  - At GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M)
  - Young Faculty: NSF CRII’20, NSF CAREER’23, Amazon Research Award’23

Publications

- 27 journals/confs. papers since ’16 (11 since joining GMU in ’21)
  - 20 papers with students (9 with undergrad)
- Google: 3617 citations (h-index 17 i10-index 24)
- SIGSOFT MIP paper award, SIGEVO Impact paper award

Students Mentoring

- Current: 3 Ph.D RA’s, 2 undergrads
- Graduated (at UNL): 1 PhD, 2 Masters, 11 undergrads (2 Outstanding Undergrad Research Awards)

Teaching

- At GMU (2 years): 1 grad (2x, required, SWE619), 1 undergrad (SWE419), 1 seminar (CS695)
- Developed online SWE619 course with Wiley (went live in Spring’23)
Funding
- 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (Co-PI), 2 industry (sole-PI), 1 internal (sole-PI)
  - Total $2.65M; my share $1.5M, as PI $1.3M
  - At GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M)
  - Young Faculty: NSF CRII’20, NSF CAREER’23, Amazon Research Award’23

Publications
- 27 journals/confs. papers since ’16 (11 since joining GMU in ’21)
  - 20 papers with students (9 with undergrad)
- Google: 3617 citations (h-index 17 i10-index 24)
- SIGSOFT MIP paper award, SIGEVO Impact paper award

Students Mentoring
- Current: 3 Ph.D RA’s, 2 undergrads
- Graduated (at UNL): 1 PhD, 2 Masters, 11 undergrads (2 Outstanding Undergrad Research Awards)

Teaching
- At GMU (2 years): 1 grad (2x, required, SWE619), 1 undergrad (SWE419), 1 seminar (CS695)
- Developed online SWE619 course with Wiley (went live in Spring’23)

Services
- Regularly serve in well-known confs/journals, 7 NSF panels in past 5 consec. yrs
- At GMU: program director of MS SWE; organize Virtual Open House; maintain CSRankings DB (GMU is ranked 32!)