Detecting Numerical **B**ugs in Neural Network **Ar**chitectures

Yuhao Zhang[1], Luyao Ren[1], Liqian Chen[2], **Yingfei Xiong**[1], Shing-Chi Cheung[3], Tao Xie[1]

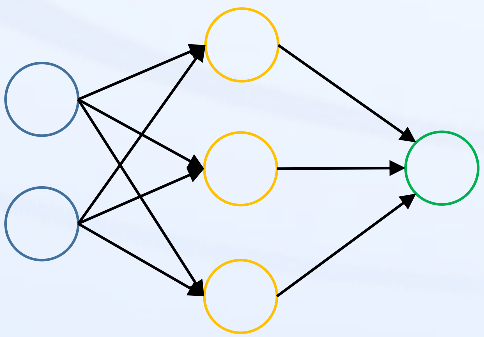
Peking University[1]
National University of Defense Technology[2]
The Hong Kong University of Science and Technology[3]

❶ **De**tecting ❷ Numerical **B**ugs in Neural Network **Ar**chitectures

Background & Motivation

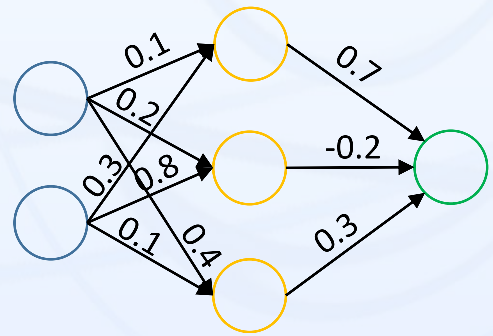
Neural Network Architecture

Neural Network Architecture



Training

Neural Network Model

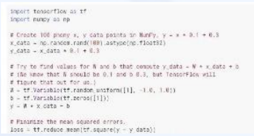


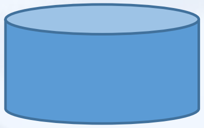

```
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
# (We know that W should be 0.1 and b 0.3, but TensorFlow will
# figure that out for us.)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.global_variables_initializer()

# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in range(201):
```

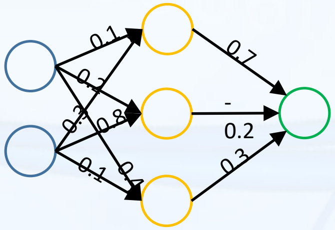
?

Existing work on NN model:
Testing
Verification
Bug Detection
…

Why Neural Network **Architecture**?







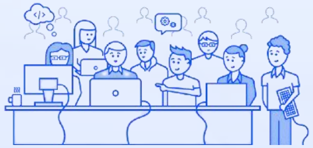
Code (Architecture) Data Training (Magic) NN Model

1. Bugs at model level are difficult to fix

Hours, Days, Weeks, Months, …

2. Bugs in architectures may cause failures in training

A NN Architecture →



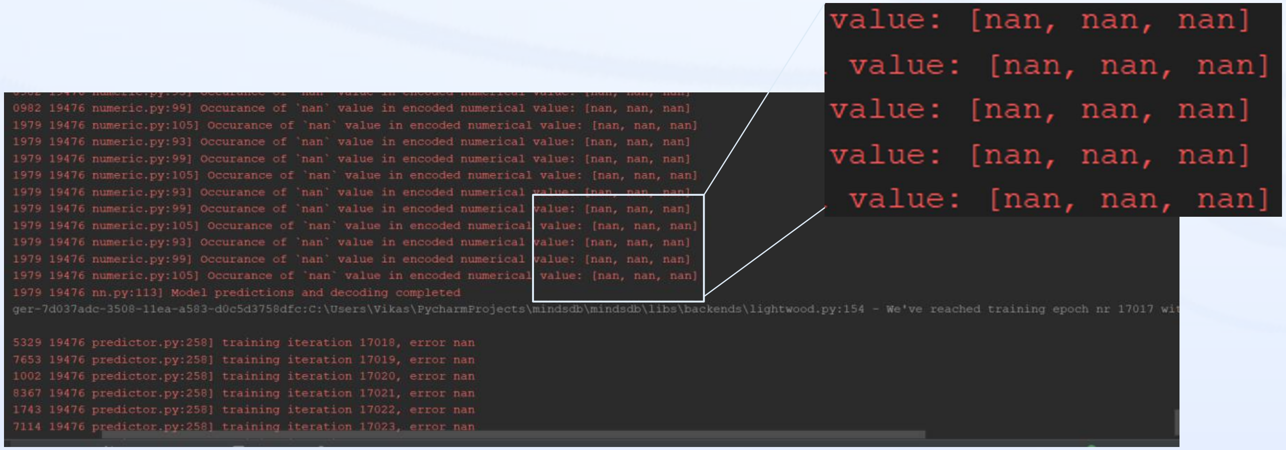
Many NN Models →

An architecture vendor Many Developers Software Systems

3. Quality assurance needs to be provided for architectures

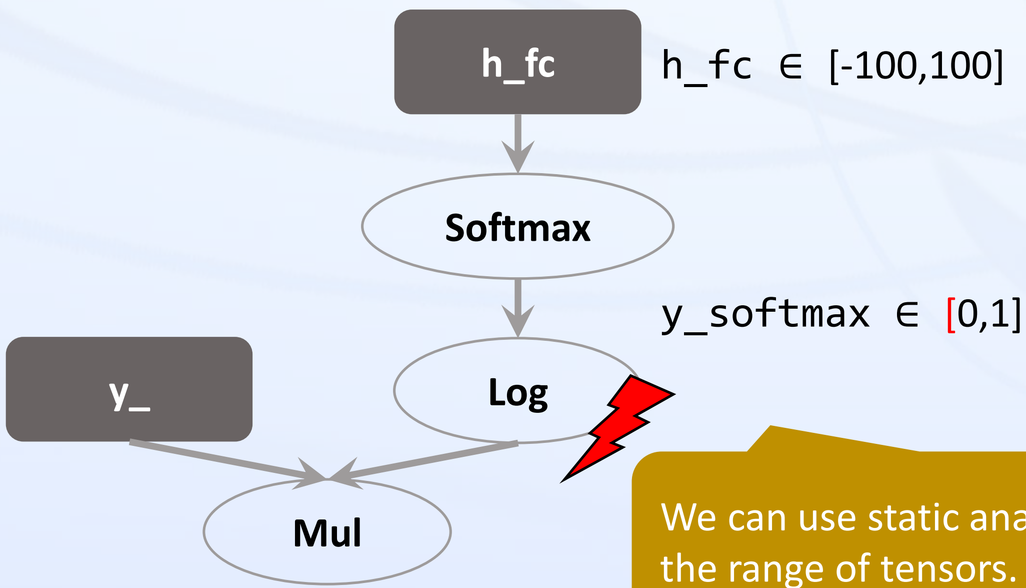
Numerical Bugs

Bugs leading to errors in numerical operations, such as “NaN”, “INF”, or crashes during training or inference.



An Example of Numerical Bugs

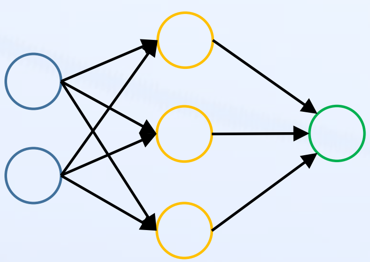
```
…
1. y_softmax = tf.nn.softmax(h_fc)
2. cross_entropy = y_ * tf.log(y_softmax)
…
```



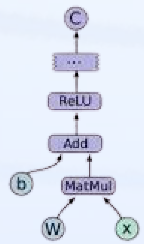
We can use static analysis to infer the range of tensors.

Detecting Numerical **B**ugs in Neural Network **Ar**chitectures

NN Architecture



Computation Graph



Static Analysis

Check Unsafe Operations

Log
Exp
...

Abstract Interpretation

Infinitely many possible inputs and parameters for an NN architecture

$x = 0.14, 0.55, \ldots, 0.99$ → $\sigma(x) = [0,1]$

1. How about tensors?
2. How can we improve the precision of interval abstraction?

Abstraction for Neural Network Architectures

1. Tensor Partitioning
2. Interval Abstraction with Affine Equality Relation

Abstraction on Tensors

	Tensor Expansion	Tensor Smashing
	• Instantiate every element • Precise but not scalable	• Smash an array into one element • Scalable but not precise
`1. A = Matrix(2, 2);` `//within [-1,0]`	$\sigma(A) = \begin{pmatrix} [-1,0] & [-1,0] \\ [-1,0] & [-1 \ldots \end{pmatrix}$ [illegible]	[illegible]
`2. A[1][1] = 1;`	$\sigma(A) = \begin{pmatrix} [-1,0] & [-1 \ldots \\ [-1,0] & [1,1] \end{pmatrix}$ [illegible]	$\sigma(A) = [-1,1]$
`3. A[0][0] += A[1][1];`	$\sigma(A) = \begin{pmatrix} [0,1] & [-1,0] \\ [-1,0] & [1,1] \end{pmatrix}$	$\sigma(A) = [-2,2]$

Tensor Partitioning combines the strengths, scalable and precise enough

Motivation of Tensor Partitioning

- Many elements of a tensor are subject to the same operations.

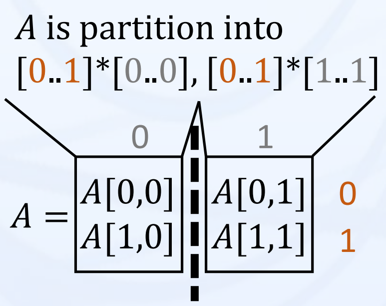
  Provide opportunity to reduce analysis effort
- Some operations like *concatenate* and *split* provide partition positions.

  Partition positions come free.

Abstraction on Tensors

Tensor Partitioning

- Partition the tensor into a set of disjoint parts
- Smash each part into one element




```
1. A = Matrix(2,2);
        //within [-1,0]
```

$\sigma(A) = ([-1{,}0] \quad [-1{,}0])$

```
2. A[1][1] = 1;
```

$\sigma(A) = ([-1{,}0] \quad [-1{,}1])$

```
3. A[0][0] += A[1][1];
```

$\sigma(A) = ([-2{,}1] \quad [-1{,}1])$

Motivation of Affine Equality Relation

Many elementwise affine operations in computation graphs

Affine equality relation is more precise than sole interval abstraction.

Sole Interval Abstraction

`1. a, b`	$\sigma(a) = [0,1], \sigma(b) = [1,2]$
`2. x = a + b;`	$\sigma(x) = [0,1] + [1,2] = [1,3]$
`3. y = a - b;`	$\sigma(y) = [0,1] - [1,2] = [-2,0]$
`4. z = x + y;`	$\sigma(z) = [1,3] + [-2,0] = [-1,3]$

Interval abstraction abstracts away the relation between x, y, and z

Interval Abstraction with Affine Equality Relation

Affine Equality Relation: $\sum_i \omega_{\mathrm{i}} x_i = \omega_0$

`1. a, b`	$\sigma(a) = [0,1], \sigma(b) = [1,2]$	
`2. x = a + b;`	$\sigma(x) = [0,1] + [1,2] = [1,3]$	`x = a + b`
`3. y = a - b;`	$\sigma(y) = [0,1] - [1,2] = [-2,0]$	`y = a - b`
`4. z = x + y;`	$\sigma(z) =$ ~~$[1,3] + [-2,0] = [-1,3]$~~ $[0,1] + [0,1] = [0,2]$	`z = x + y = 2a`

Evaluation

A Collection of Neural Network Architectures

- 9 buggy architectures from previous study [1, 2]
- 48 real-world architectures from tensorflow/models [3], containing different NN architectures (including CNN, RNN, GAN, HMM) in various research domains

[1] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An empirical study on TensorFlow program bugs. ISSTA 2018.
[2] Augustus Odena, Catherine Olsson, David Andersen, and Ian J. Goodfellow. 2019. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. ICML 2019.
[3] https://github.com/tensorflow/models/tree/master/research

Main Results

Framework
(Tensor Abstraction + Numerical Abstraction)

Instantiate every element in an array
$\sigma(A) = \begin{pmatrix} [0,1] & [-1,0] \\ [-1,0] & [1,1] \end{pmatrix}$

DEBAR
(Tensor Partitioning + Affine Equality Relation):
Accuracy: 93.0%, all in 3 minutes, 12.1s on average
100% accuracy on 9 buggy architectures

Tensor Expansion + Affine Equality Relation:
33/57 > 30mins; on rest 24, DEBAR doesn't lost accuracy

Tensor Smashing + Affine Equality Relation:
Accuracy: 87.1%, 12.2s on average

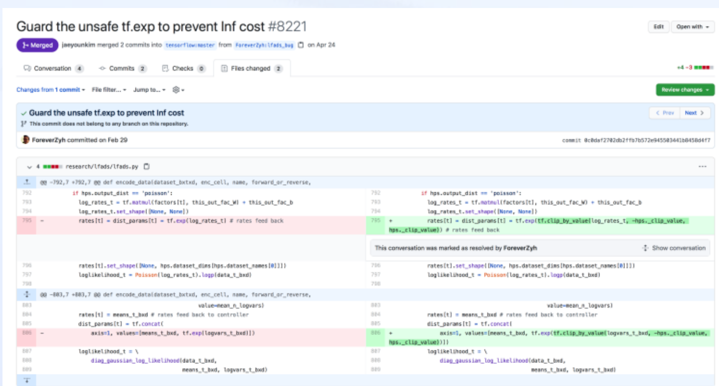
Smash an array into one element
$\sigma(A) = [-2,2]$

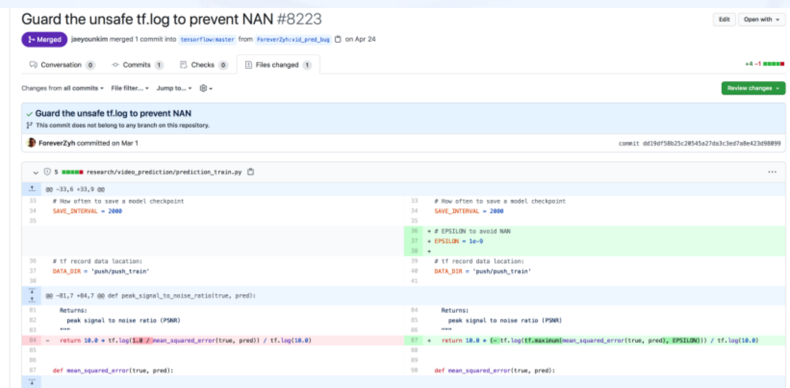
Tensor Partitioning + Sole Interval Abstraction
Accuracy: 80.6%, 12.1s on average

Bugs in Real-World Architectures

Found 11 buggy statements in the code repository

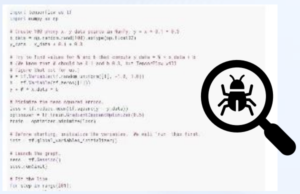
Submitted pull requests, and 3 buggy statements have been repaired by the developers

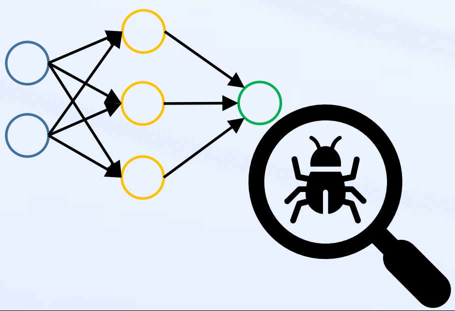




Summary

How to map the buggy operations to the buggy code statements ?



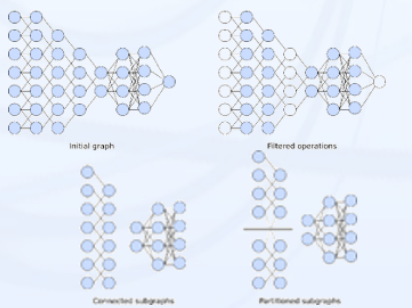


Detect Numerical Bugs in Neural Network Architectures

Design Abstraction Techniques for Analyzing Neural Architectures

1. Interval Abstraction with Affine Equality Relation
2. Tensor Partitioning

How to analyze the dynamic computation graphs?



Collect 57 Computation Graphs for Future Research