

# Detecting Numerical Bugs in Neural Network Architectures

Yuhao Zhang<sup>1</sup>, Luyao Ren<sup>1</sup>, Liqian Chen<sup>2</sup>, **Yingfei Xiong**<sup>1</sup>,  
Shing-Chi Cheung<sup>3</sup>, Tao Xie<sup>1</sup>

Peking University<sup>1</sup>

National University of Defense Technology<sup>2</sup>

The Hong Kong University of Science and Technology<sup>3</sup>

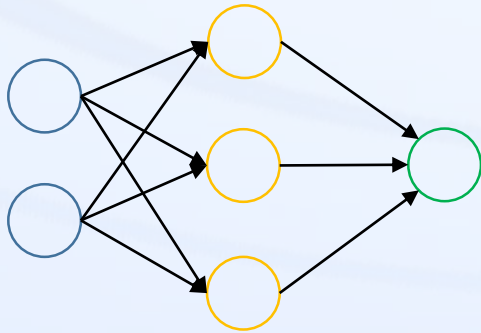


# ① Detecting <sup>②</sup> Numerical Bugs in Neural Network Architectures

Background & Motivation

# Neural Network Architecture

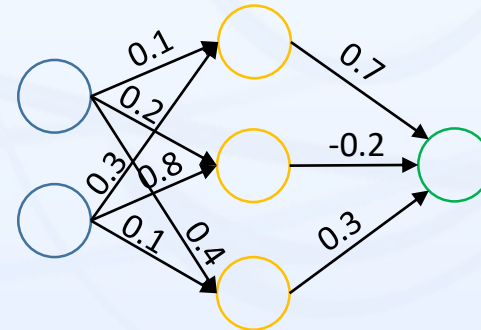
Neural Network Architecture



Training



Neural Network Model



Existing work on NN model:

- Testing
- Verification
- Bug Detection

...



```

import tensorflow as tf
import numpy as np

# Create 100 pairs x, y data points in NumPy. y = a * x + 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
# (the 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):
    
```

# Why Neural Network Architecture?

```

import tensorflow as tf
import numpy as np

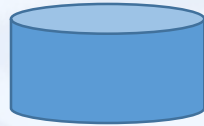
# Create 100 noisy x, y data points for noisy, y = x * 0.1 + 0.3
x_data = np.random.rand(100) * 0.9999999999999999
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that minimize y_data - W * x_data + b
# The idea that it should be 0.1 and 0.3, but TensorFlow will
# figure that out for us.
# TF Variabletf.Variable(tf.zeros([1], dtype=tf.float32))
W = tf.Variable(tf.zeros([1]))
b = tf.Variable(tf.zeros([1]))

# Pass in the noisy observed errors.
sess = tf.Session()
cost = tf.reduce_mean(tf.square(y_data -

```

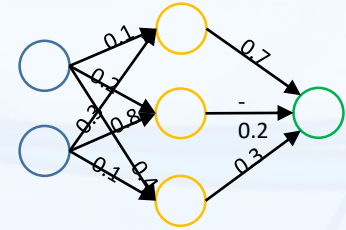
Code (Architecture)



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



An architecture vendor

A NN Architecture



Many Developers

Many NN Models



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.

```
1979 19476 numeric.py:91] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
0982 19476 numeric.py:99] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:105] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:93] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:99] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:105] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:93] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:99] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:105] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:93] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:99] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 numeric.py:105] Occurrence of 'nan' value in encoded numerical value: [nan, nan, nan]
1979 19476 nn.py:113] Model predictions and decoding completed
ger-7d037adc-3508-11ea-a583-d0c5d3758dfc:C:\Users\Vikas\PycharmProjects\mindsdb\mindsdb\libs\backends\lightwood.py:154 - We've reached training epoch nr 17017 with error nan

5329 19476 predictor.py:258] training iteration 17018, error nan
7653 19476 predictor.py:258] training iteration 17019, error nan
1002 19476 predictor.py:258] training iteration 17020, error nan
8367 19476 predictor.py:258] training iteration 17021, error nan
1743 19476 predictor.py:258] training iteration 17022, error nan
7114 19476 predictor.py:258] training iteration 17023, error nan
```

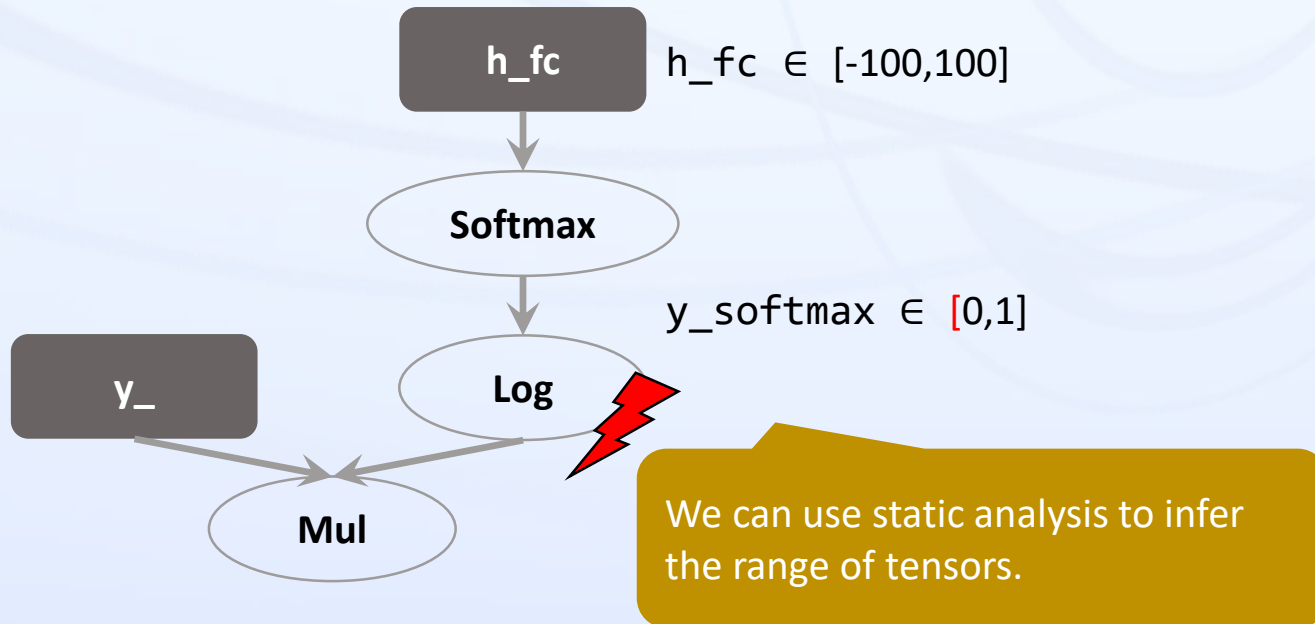
```
value: [nan, nan, nan]
value: [nan, nan, nan]
value: [nan, nan, nan]
value: [nan, nan, nan]
value: [nan, nan, nan]
```

NaN



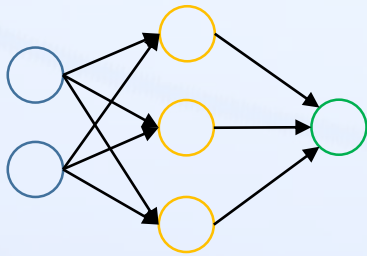
# An Example of Numerical Bugs

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

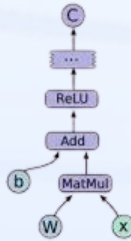


# Detecting Numerical Bugs in Neural Network Architectures

NN Architecture



Computation Graph



Static Analysis



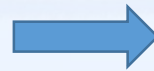
Check Unsafe Operations

Log  
Exp  
...

# Abstract Interpretation

Infinitely many possible inputs and parameters for an NN architecture

$x = 0.14, 0.55, \dots, 0.99$



$\sigma(x) = [0,1]$

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	
	<ul style="list-style-type: none"> <li>• Instantiate every element</li> <li>• Precise but not scalable</li> </ul>	<ul style="list-style-type: none"> <li>• Smash an array into one element</li> <li>• Scalable but not precise</li> </ul>	
<u>1. A = Matrix(2, 2);</u> <span style="color: red;">//within [-1,0]</span>	$\sigma(A) = \begin{pmatrix} [-1,0] & [-1,0] \\ [-1,0] & [-1,0] \end{pmatrix}$	<div style="background-color: #f0e68c; padding: 5px; border: 1px solid black;">           Tensor Partitioning combines the strengths, scalable and precise enough         </div>	
<u>2. A[1][1] = 1;</u>	$\sigma(A) = \begin{pmatrix} [-1,0] & [-1,0] \\ [-1,0] & [1,1] \end{pmatrix}$		$\sigma(A) = [-1,1]$
<u>3. A[0][0] += A[1][1];</u>	$\sigma(A) = \begin{pmatrix} [0,1] & [-1,0] \\ [-1,0] & [1,1] \end{pmatrix}$		$\sigma(A) = [-2,2]$

# 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]`

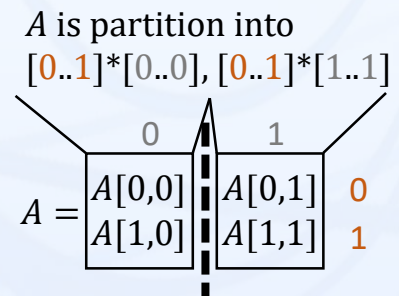
$$\sigma(A) = ([-1,0] \quad [-1,0])$$

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:  $\sum_i \omega_i x_i = \omega_0$

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.  $z = x + y;$

$$\sigma(z) = \cancel{[1,3]} + \cancel{[-2,0]} = \cancel{[-1,3]}$$

$$z = x + y = 2a$$

$$[0,1] + [0,1] = [0,2]$$

# 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

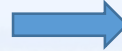
$$\sigma(A) = \begin{pmatrix} [0,1] & [-1,0] \\ [-1,0] & [1,1] \end{pmatrix}$$

## 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



Tensor Partitioning + Sole Interval Abstraction

Accuracy: 80.6%, 12.1s on average

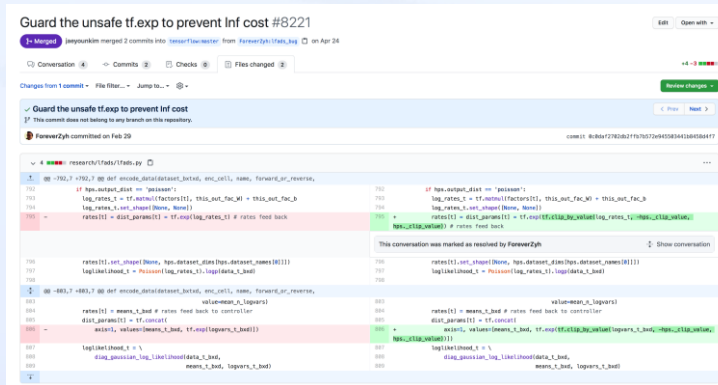
Smash an array into one element

$$\sigma(A) = [-2,2]$$

# Bugs in Real-World Architectures

Found 11 buggy statements in the code repository

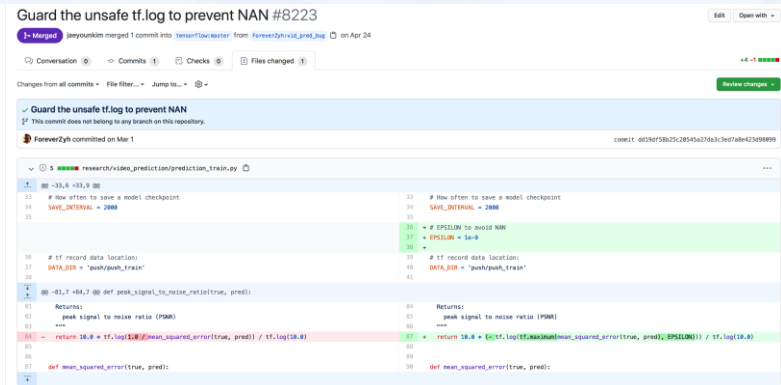
Submitted pull requests, and 3 buggy statements have been repaired by the developers



Guard the unsafe tf.exp to prevent Inf cost #8221

research/infra/infra.py

```
@@ -792,7+792,7 @@ def encode_dataset_batch, enc_ctx, name, forward_tf_reverser,
792     if hps.debug_list == 'debug':
793         log_rate = tf.math.reduce_max(log_rate)
794         log_rate = tf.math.reduce_max(log_rate)
795         rates[0] = dist_params[0] + tf.math.exp(rates[0]) # rates feed back
796         rates[0] = dist_params[0] + tf.math.exp(rates[0]) # rates feed back
797
798 rates[0] = dist_params[0] + tf.math.exp(rates[0]) # rates feed back
799 log_likelihood = tf.math.reduce_sum(rates)
800
801 @@ -883,7+883,7 @@ def decode_dataset_batch, dec_ctx, name, forward_tf_reverser,
883     rates[0] = mean_s_bnd # rates feed back to controller
884     dist_params[0] = tf.constant(
885         rates[0], value=[mean_s_bnd, tf.math.exp(rates[0])], name='dist_params')
886     rates[0] = mean_s_bnd + tf.math.exp(rates[0]) # rates feed back
887     log_likelihood = tf.math.reduce_sum(rates)
888     dist_params[0] = tf.constant(
889         rates[0], value=[mean_s_bnd, tf.math.exp(rates[0])], name='dist_params')
890     rates[0] = mean_s_bnd + tf.math.exp(rates[0]) # rates feed back
891     log_likelihood = tf.math.reduce_sum(rates)
892     dist_params[0] = tf.constant(
893         rates[0], value=[mean_s_bnd, tf.math.exp(rates[0])], name='dist_params')
```



Guard the unsafe tf.log to prevent NAN #8223

research/infra/infra.py

```
@@ -32,6+32,6 @@
32     # How often to save a model checkpoint
33     SAVE_INTERVAL = 2000
34     # EPSSLON to avoid NAN
35     EPSSLON = 1e-9
36     # tf record data location:
37     DATA_DIR = 'path/path_train'
38
39 @@ -81,7+81,7 @@ def peak_signal_to_noise_ratio(pred):
81     Returns:
82     peak signal to noise ratio (PSNR)
83
84     return 10 * tf.math.log10(tf.math.reduce_max(tf.abs(pred - tf.math.reduce_mean(pred))) / tf.math.reduce_max(tf.abs(pred)))
85
86 def mean_squared_error(true, pred):
87     Returns:
88     mean squared error (MSE)
89
90     return tf.math.reduce_mean(tf.square(true - pred))
```

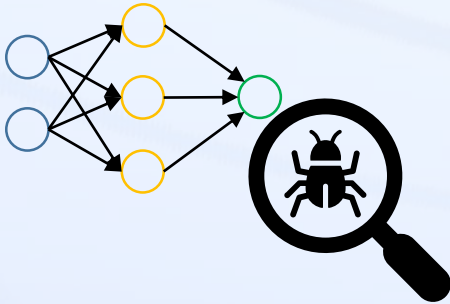
# Summary

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

```

1: class Conv2d(nn.Module):
2:     def __init__(self, in_channels, out_channels, kernel_size, stride, padding):
3:         super(Conv2d, self).__init__()
4:         self.in_channels = in_channels
5:         self.out_channels = out_channels
6:         self.kernel_size = kernel_size
7:         self.stride = stride
8:         self.padding = padding
9:         self.conv = nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)
10:    def forward(self, x):
11:        return self.conv(x)

```

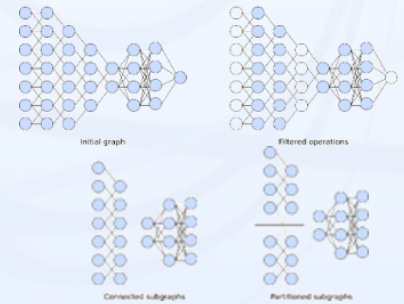


Detect Numerical Bugs in Neural Network Architectures

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