IOWA STATE UNIVERSITY
Department of Computer Science

DeepDiagnosis: Automatically Diagnosing Faults and Recommending Actionable Fixes in Deep Learning Programs




Mohammad Wardat, Breno Dantas Cruz, Wei Le, and Hridesh Rajan
wardat@iastate.edu, bdantasc@iastate.edu, weile@iastate.edu, and hridesh@iastate.edu
Dept. of Computer Science, Iowa State University

Deep Learning Is Being Used In Critical Systems

DNNs are used in autonomous vehicles to medical devices.

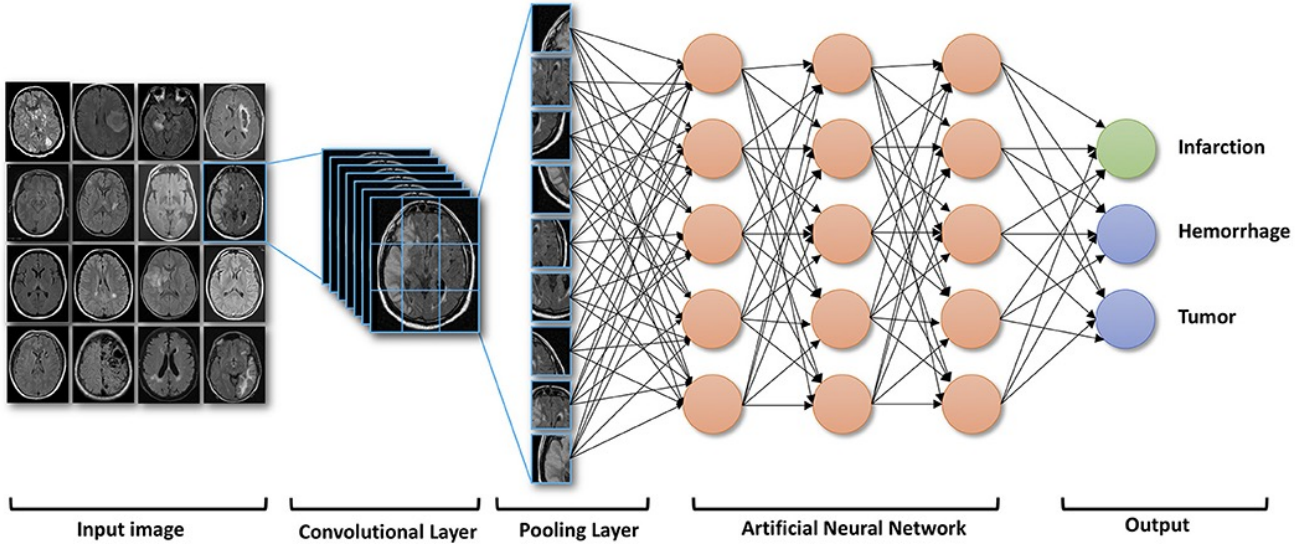
Automated Driving

Aerospace and Defense

Medical Research

Industrial Automation

Medical Electronics



Problem Statement

Existing software techniques are still not applicable to identify symptoms and their root causes.

Existing methodologies do not effectively identify bugs and provide actionable fixes to developers.

There is no existing technique that finds the relation between symptoms and root causes in the structure DNNs.

Prior Work

1. **DeepLocalize:** fault localization for deep neural networks

   Performs dynamic analysis during training to localize bugs by monitoring values.

2. **UMLAUT:** Debugging deep learning programs

   Checks the structure of deep learning programs and model behavior during training.

3. **AUTOTRAINER:** An automatic DNN training problem detection and repair

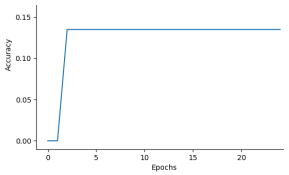
   Detects, repairs five symptoms, and provides a set of possible fixes.

4. **DEBAR:** Detecting numerical bugs in neural network architectures

   Detects numerical errors in the deep neural networks using static analysis.

[1] M. Wardat, W. Le, and H. Rajan, "Deeplocalize: fault localization for deep neural networks," in ICSE'21, 2021.
[2] E. Schoop, F. Huang, and B. Hartmann, "UMLAUT: Debugging deep learning programs using program structure and model behavior," in CHI'21, 2021.
[3] X. Zhang, J. Zhai, S. Ma, and C. Shen, "Autotrainer: An automatic dnn training problem detection and repair system," in ICSE'21, 2021.
[4] Y. Zhang, L. Ren, L. Chen, Y. Xiong, S.-C. Cheung, and T. Xie, "Detecting numerical bugs in neural network architectures," in FSE'21, 2021.

Motivation

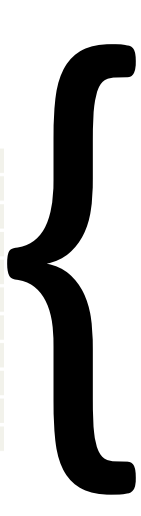


Q1) Why do I get such a bad result for such a simple dataset?

Q2) Is my dataset malformed?


```
model = Sequential()
model.add(Dense(128, 50))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(50, 50))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(50, 1))
model.add(Activation('softmax'))
model.compile(loss='binary_crossentropy', optimizer=RMSprop())
model.fit(X,Y,batch_size,epoch,validation_data=(X_test,Y_test))
```

https://stackoverflow.com/questions/31880720/



UMLAUT	No Output	STOP 104.65 Seconds
DeepLocalize	layer 7: Numerical Error	STOP 2.14 Seconds
AUTOTRAINER	Using ‘SeLU’ each layers Using ‘BatchNormalization’ … Unsolved.	STOP 495.83 Seconds

Contributions

We study the symptoms and their root causes to propose a mapping algorithm of symptoms to a fix.

We provide a benchmark with 444 models that practitioners can use to evaluate their techniques.

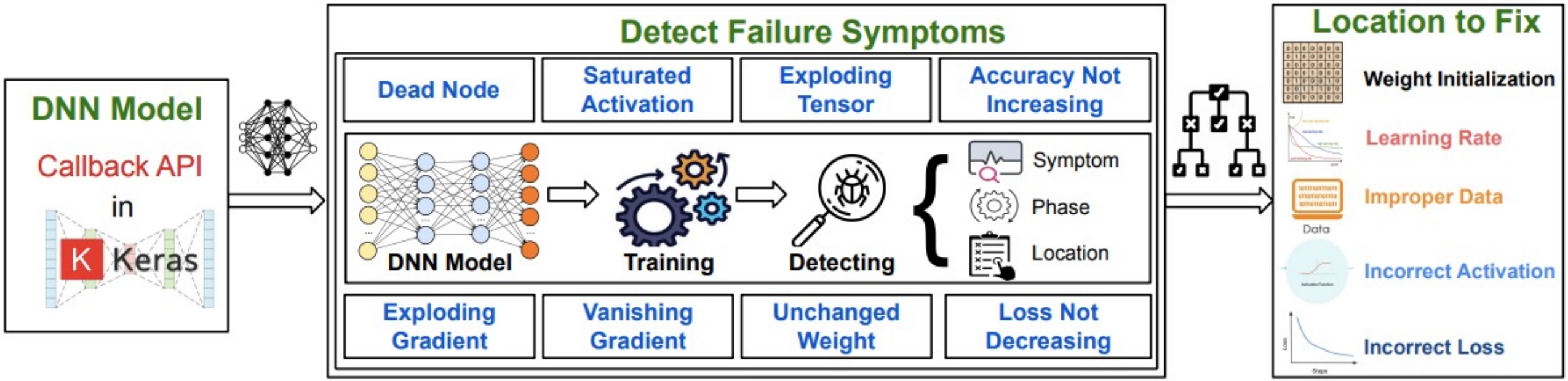
Our prototype is more comprehensive and efficient than:

UMLAUT

DeepLocalize

AUTOTRAINER

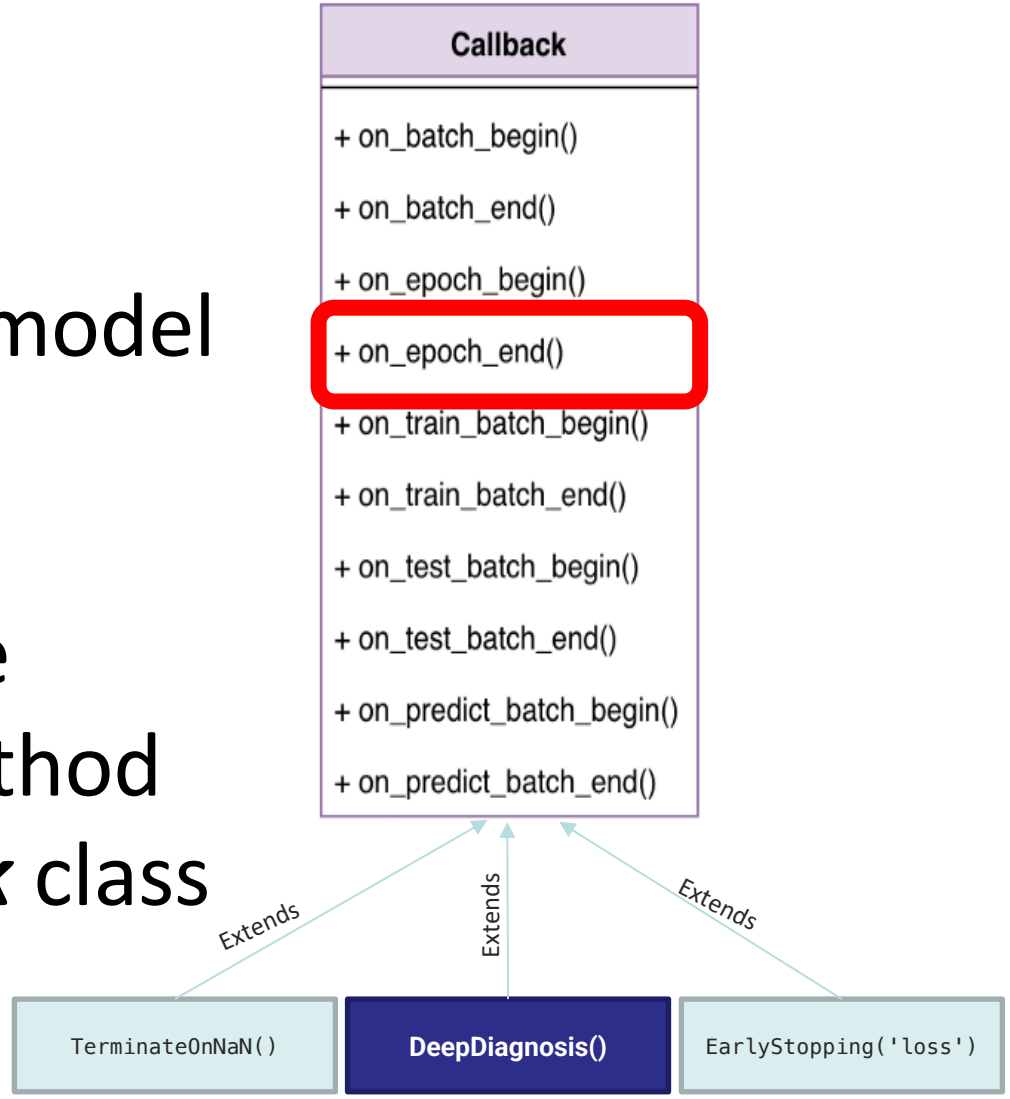
DeepDiagnosis Overview



Callback Method

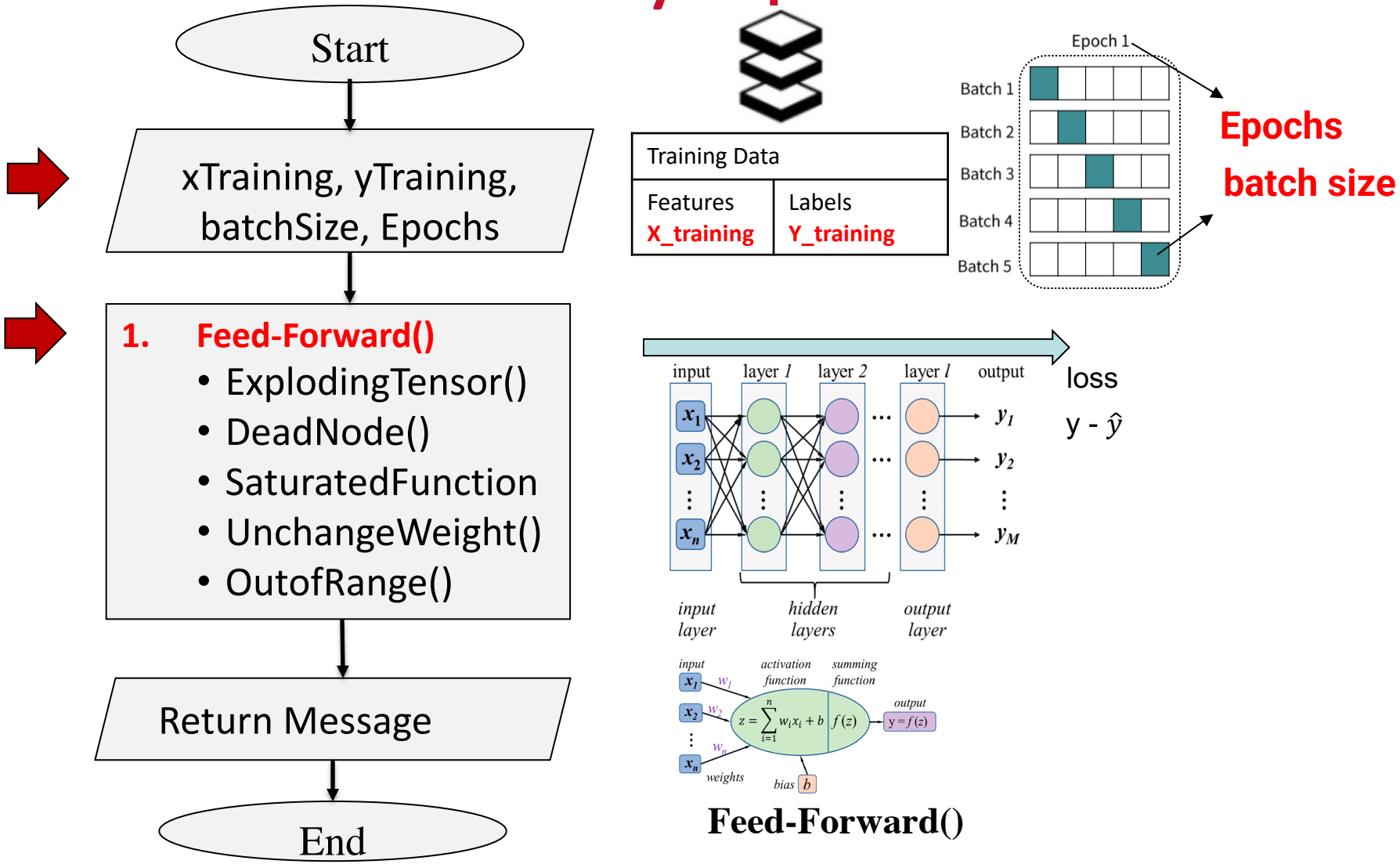
Our work focuses on monitoring model training.

To implement DeepDiagnosis, we override the ***on_epoch_end()*** method from the ***keras.callbacks.Callback*** class

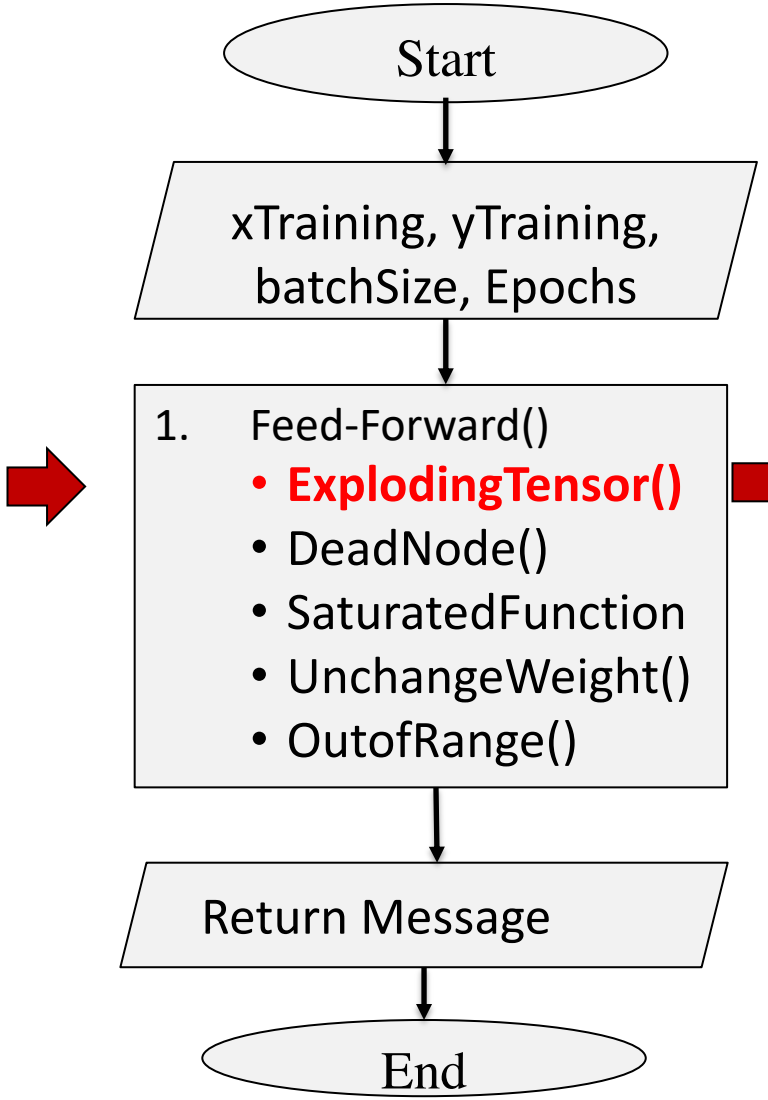


Class Diagram for Keras Callback

Algorithm: Failure Symptoms Detection



Algorithm: Failure Symptoms Detection

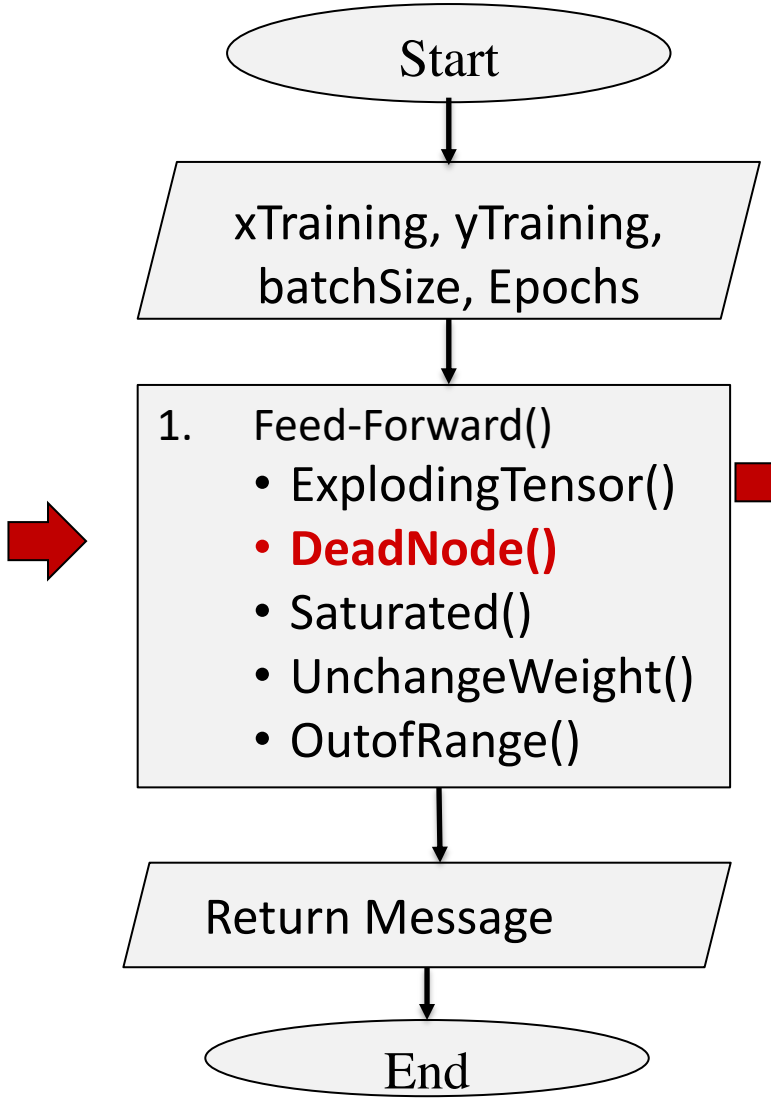


Detects any numerical error such as infinite, NaN (Not a Number), or zero.

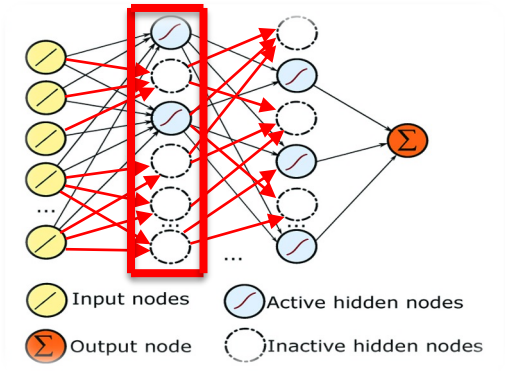
```
array([[  1.00000000e+00,   9.82503479e-05,  -2.52892733e-06,
          6.43756948e-07,  -2.64291408e-06,  -3.18651632e-06,
                     nan,              nan,              nan,
          0.00000000e+00,              nan,   5.08329823e-09,
          8.60062854e-10,   1.93595340e-09,   0.00000000e+00,
         -1.24836730e-09,   3.83645737e-09,  -3.90316446e-09,
                     nan,  -7.00323994e-07],
       [ -7.27595761e-11,   7.15255737e-07,  -2.68207733e-02,
         -6.68754578e-01,   1.68675050e-01,  -2.37232931e-02,
```

ExplodingTensor()

Algorithm: Failure Symptoms Detection

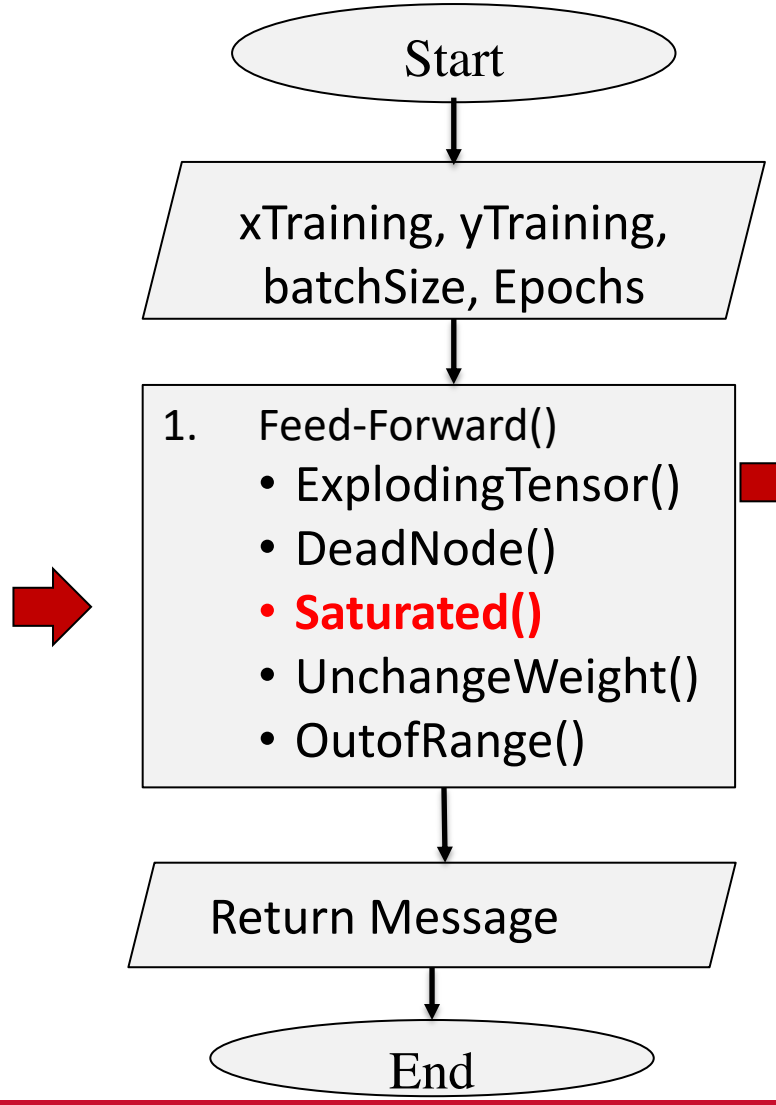


Takes the output of ReLU activation function as input, then computes how many inactive nodes dropped below Threshold.

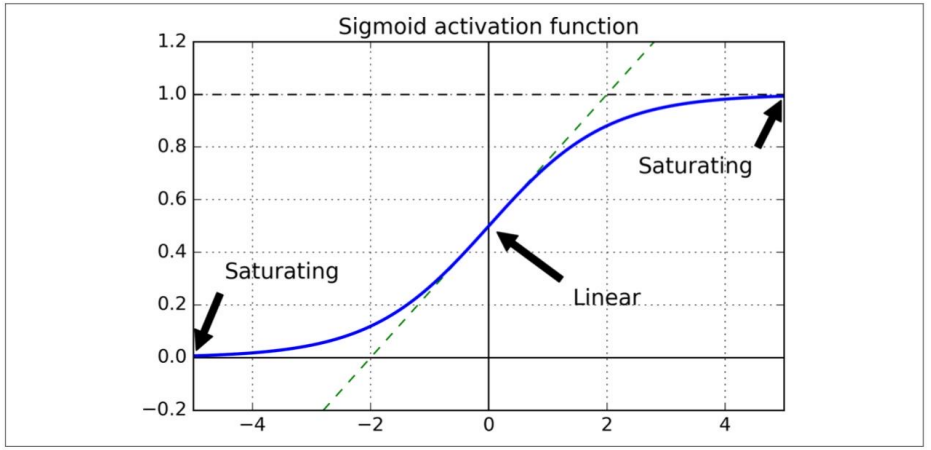


DeadNode()

Algorithm: Failure Symptoms Detection

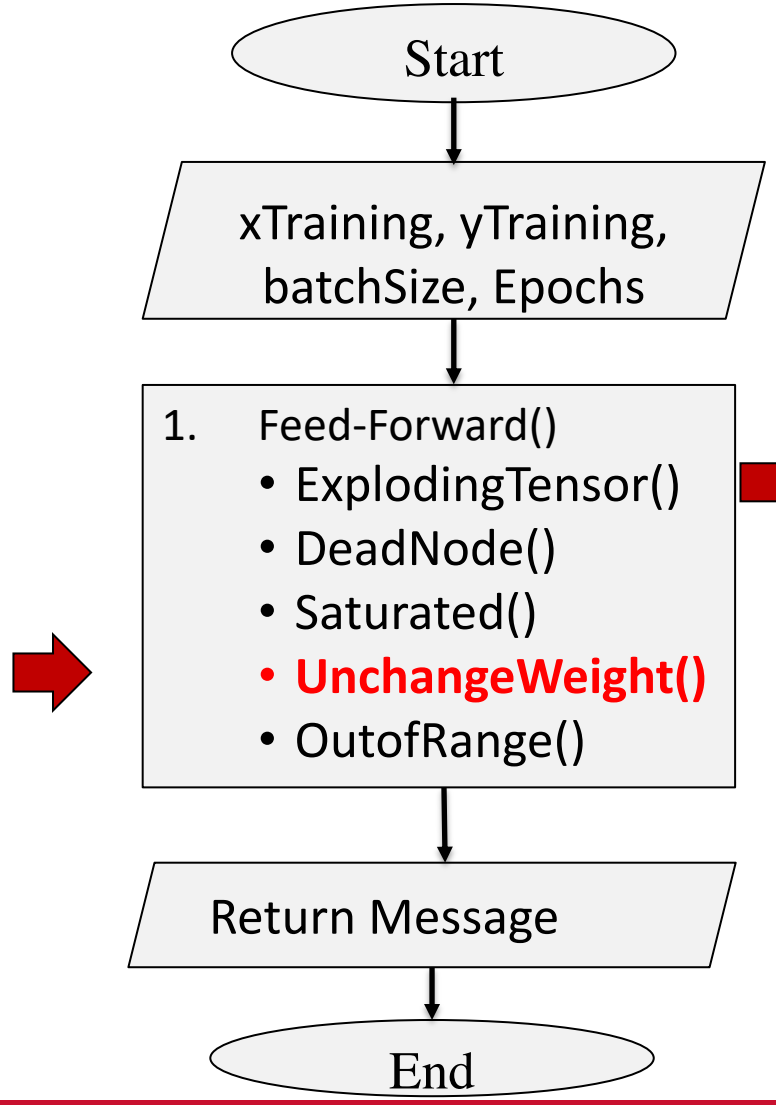


Detects if any logistic function are becoming saturated by checking when the input of function has reached either a max or min value.

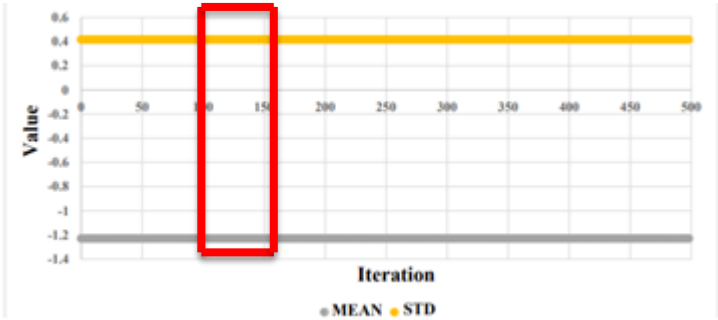


Saturated Activation Function()

Algorithm: Failure Symptoms Detection

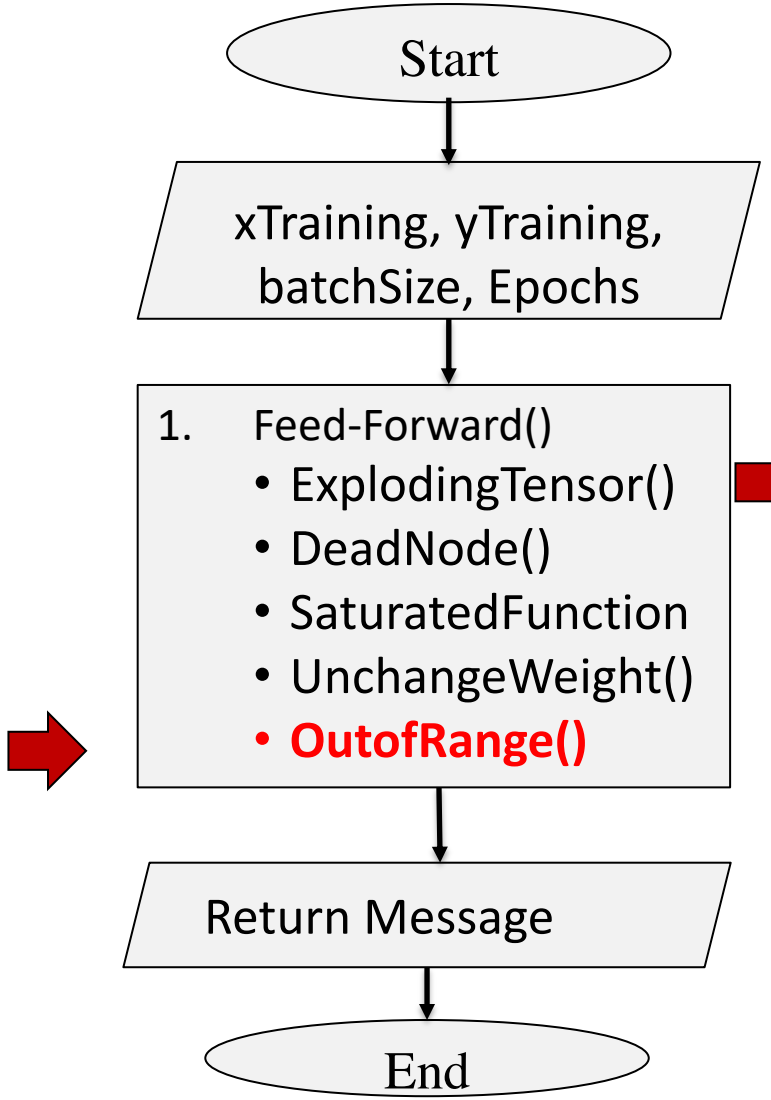


Detects if value is not changing by comparing the value for the current step with the mean value that stores in previous (N = 5) steps.

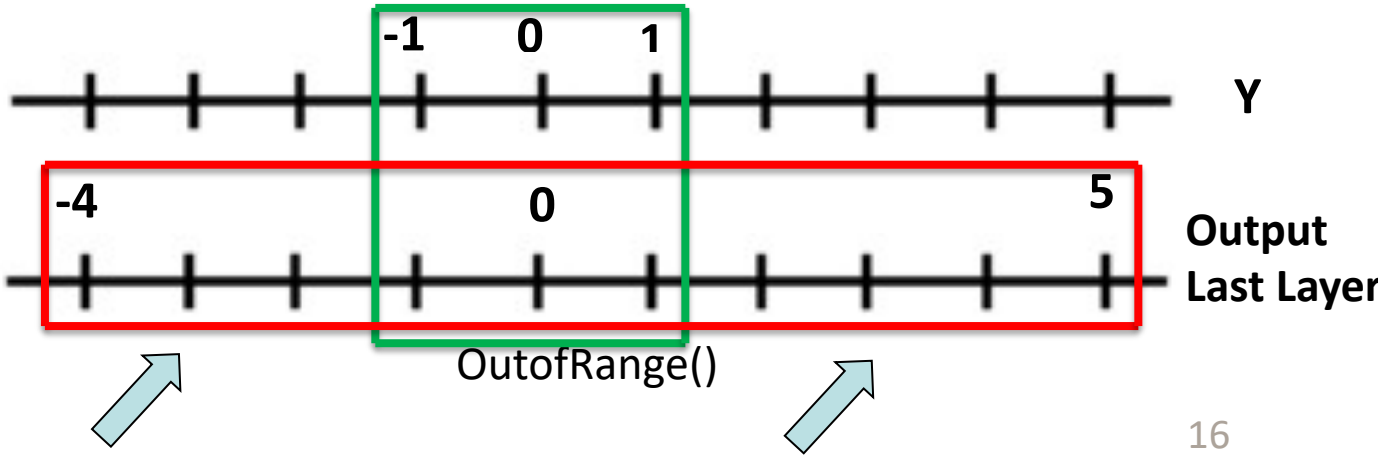


UnchangeWeight()

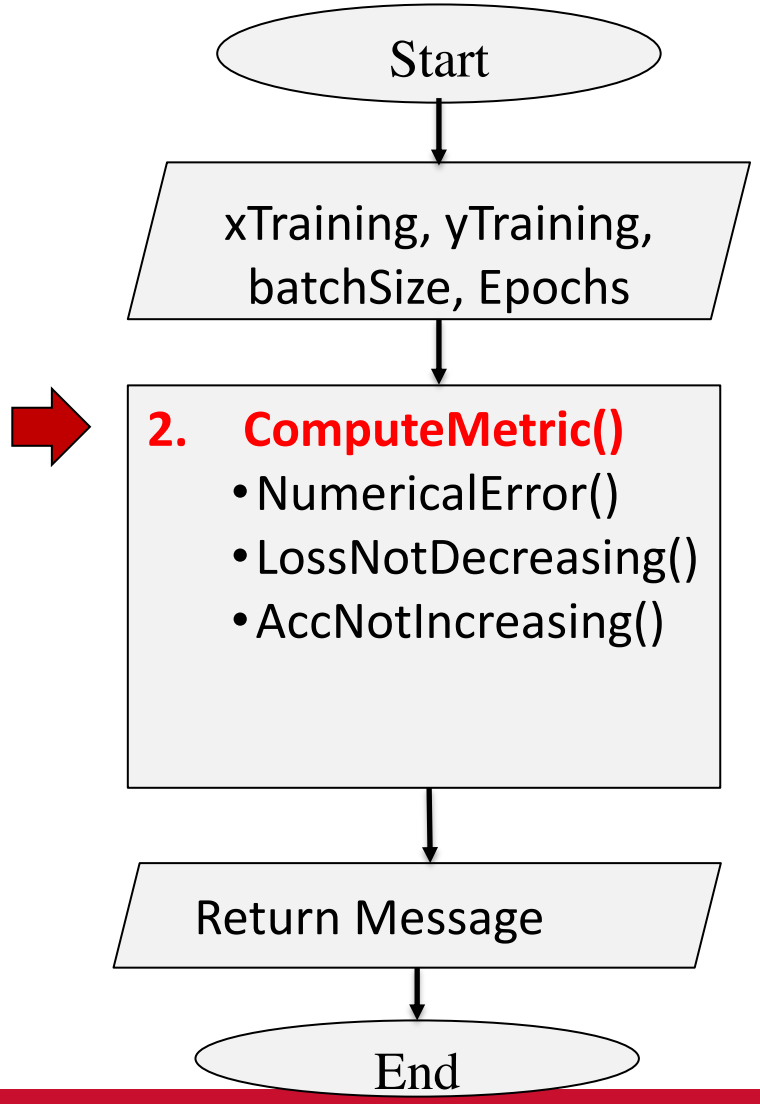
Algorithm: Failure Symptoms Detection

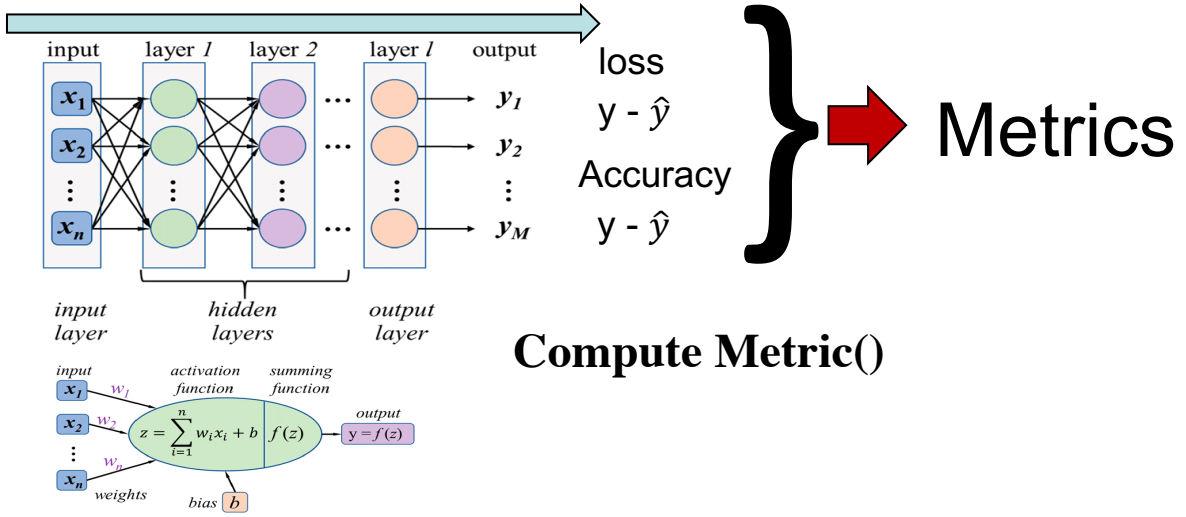


Detects if the activation function's output is becoming out of range for labeling training dataset Y.

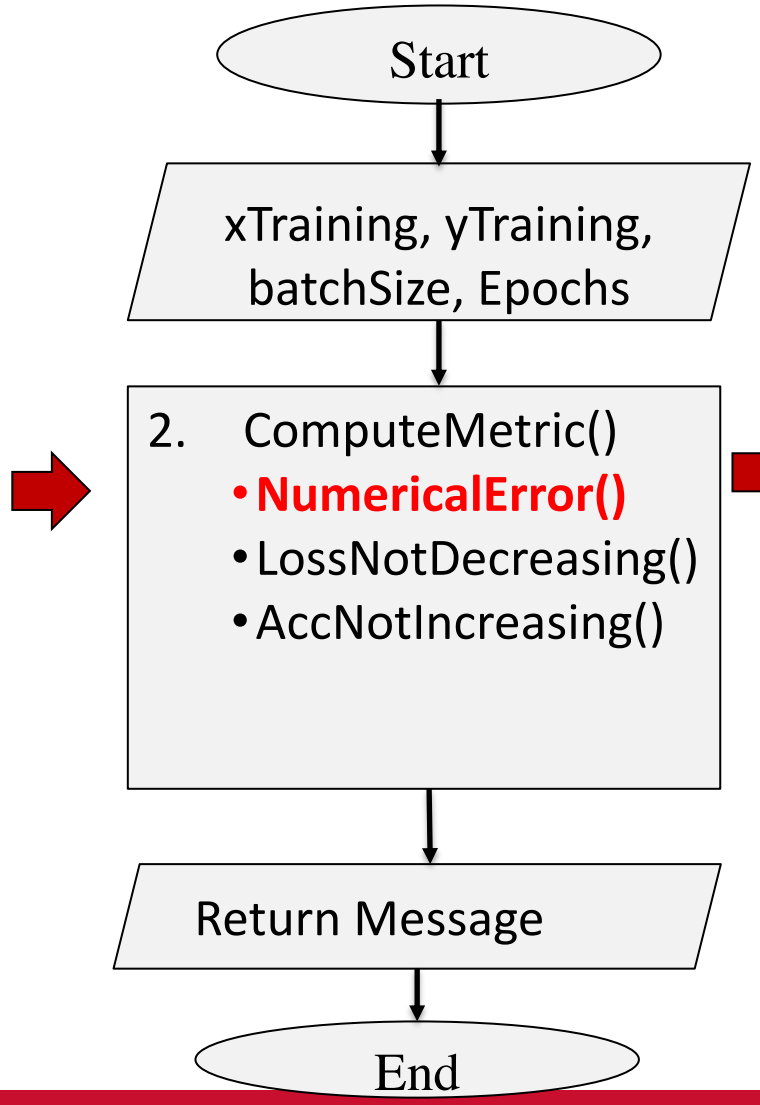


Algorithm: Failure Symptoms Detection





Algorithm: Failure Symptoms Detection

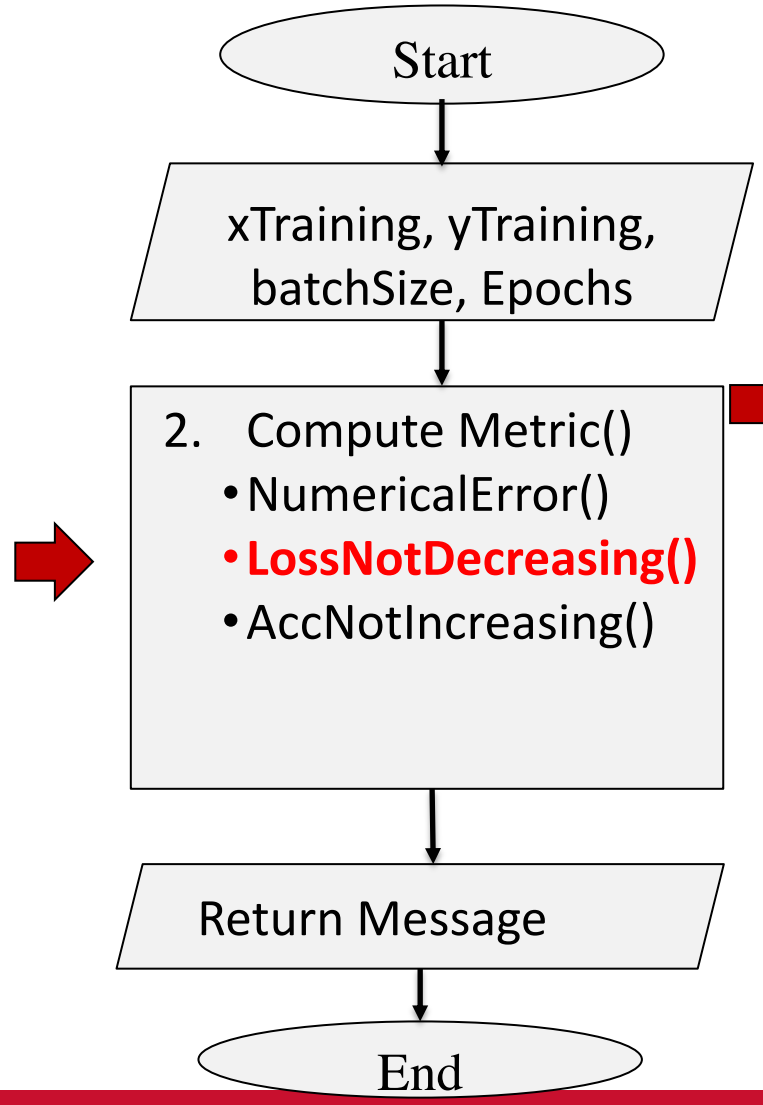


Detects any numerical error such as infinite, NaN (Not a Number), or zero.

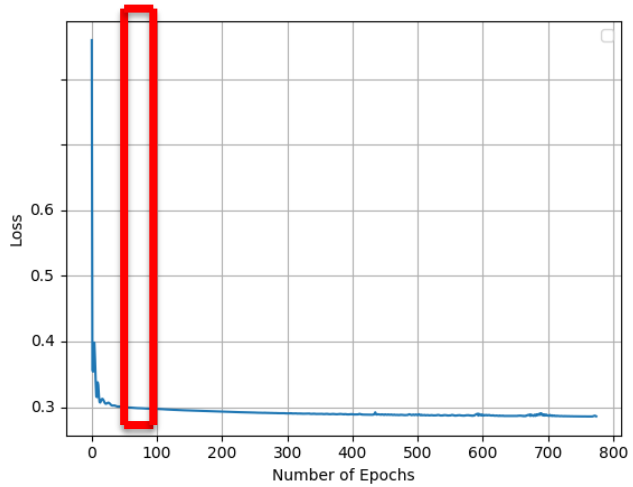
```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: nan - accuracy: 0.0987
Epoch 7/10
```

NumericalError()

Algorithm: Failure Symptoms Detection

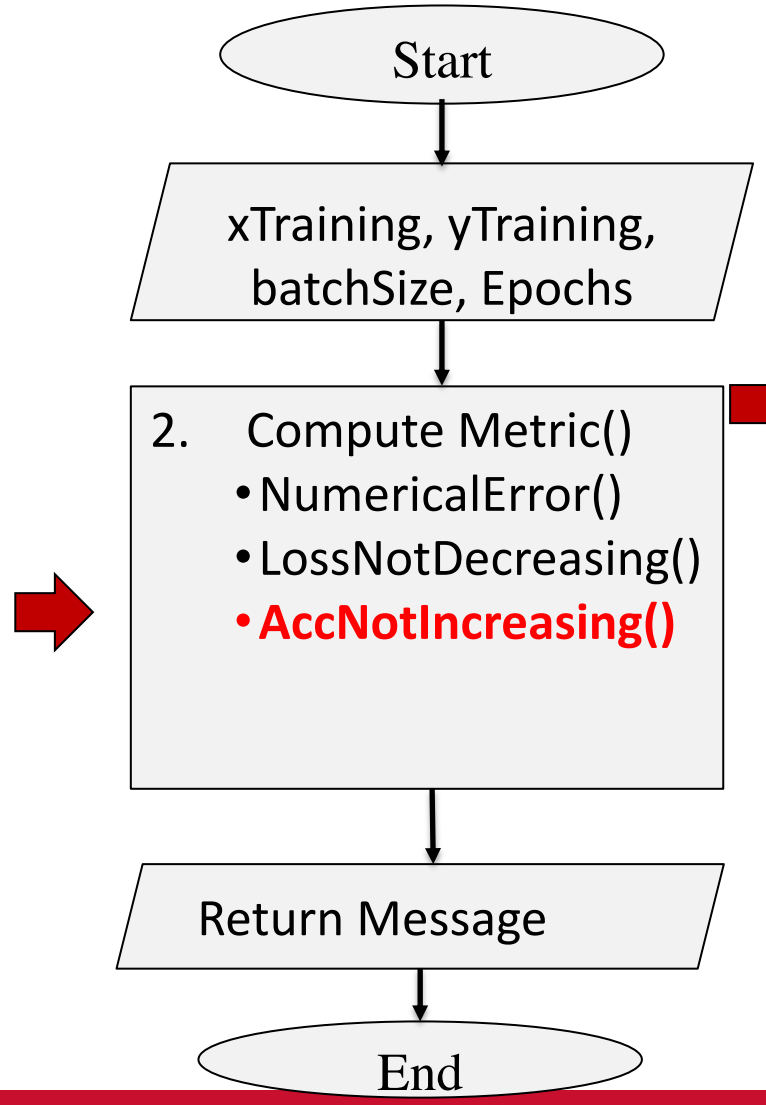


Compares the loss value for the current step with the mean value of losses that stores in previous (N = 5) steps

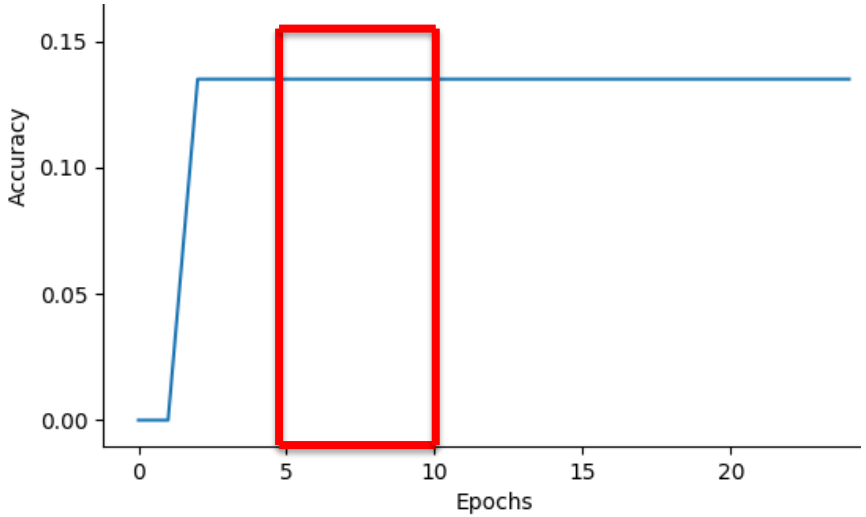


LossNotDecreasing()

Algorithm: Failure Symptoms Detection

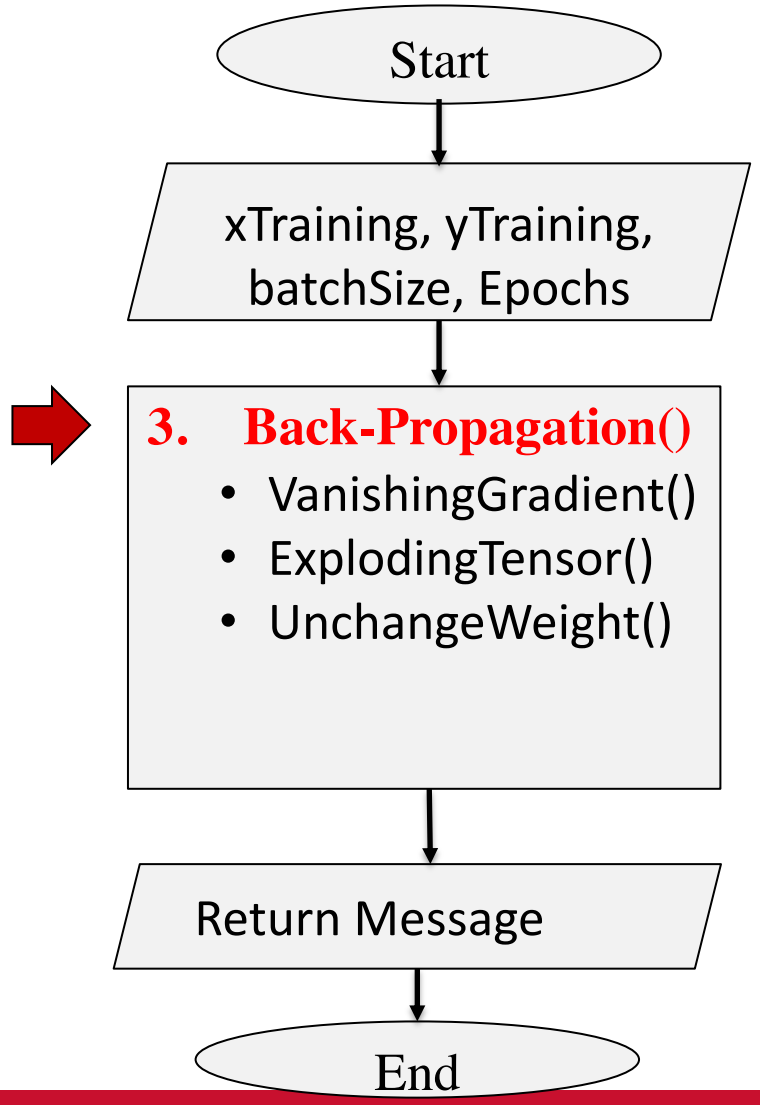


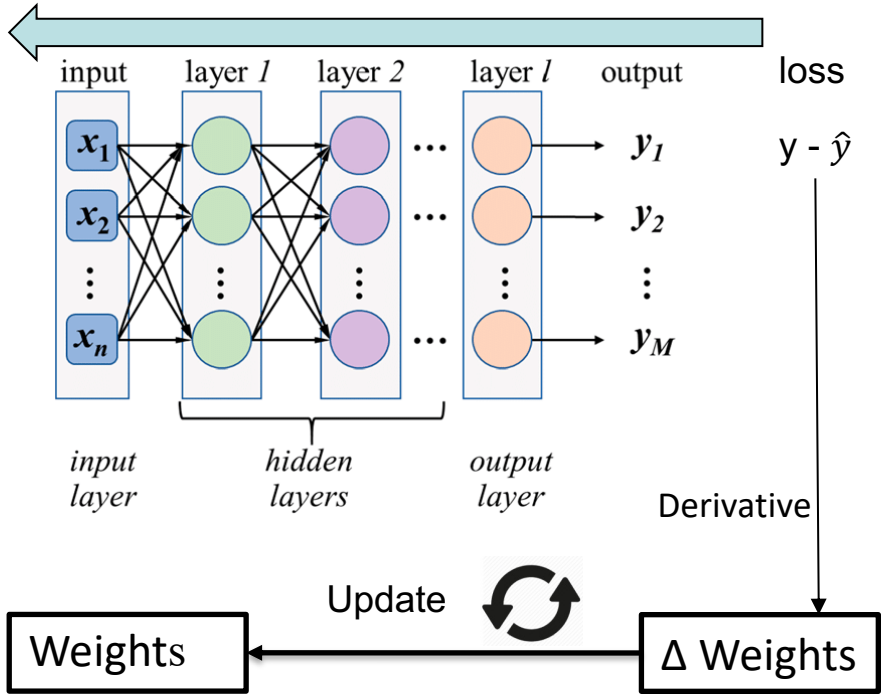
Compares the accuracy value for the current step with the mean value of accuracy that stores in previous (N = 5) steps.



AccuracyNotIncreasing()

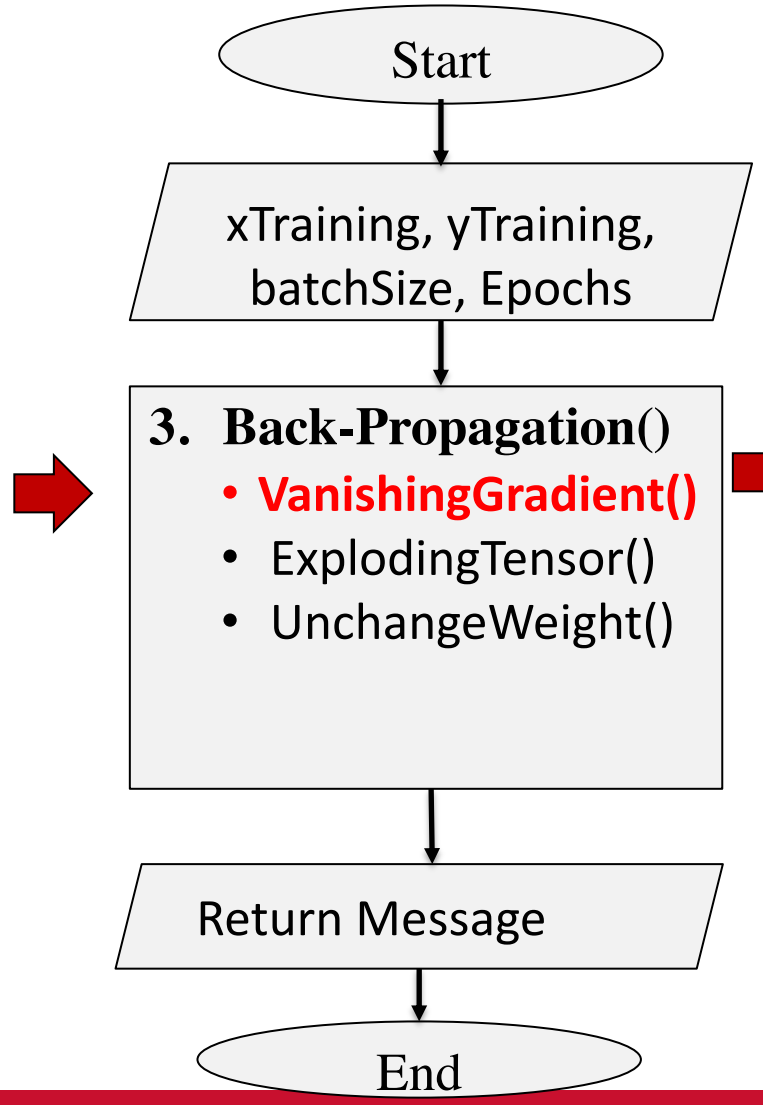
Algorithm: Failure Symptoms Detection



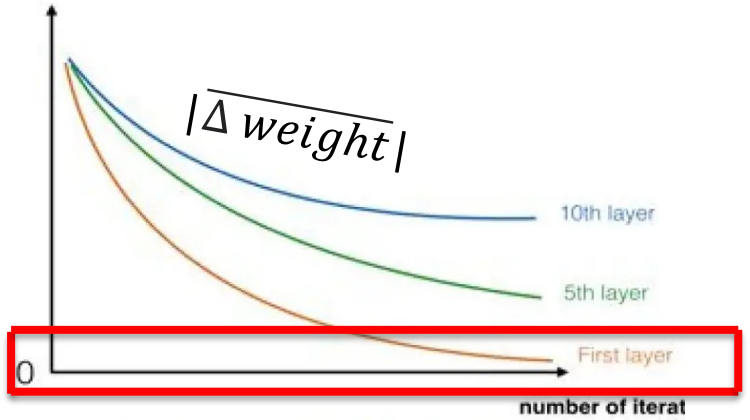


Back-Propagation()

Algorithm: Failure Symptoms Detection

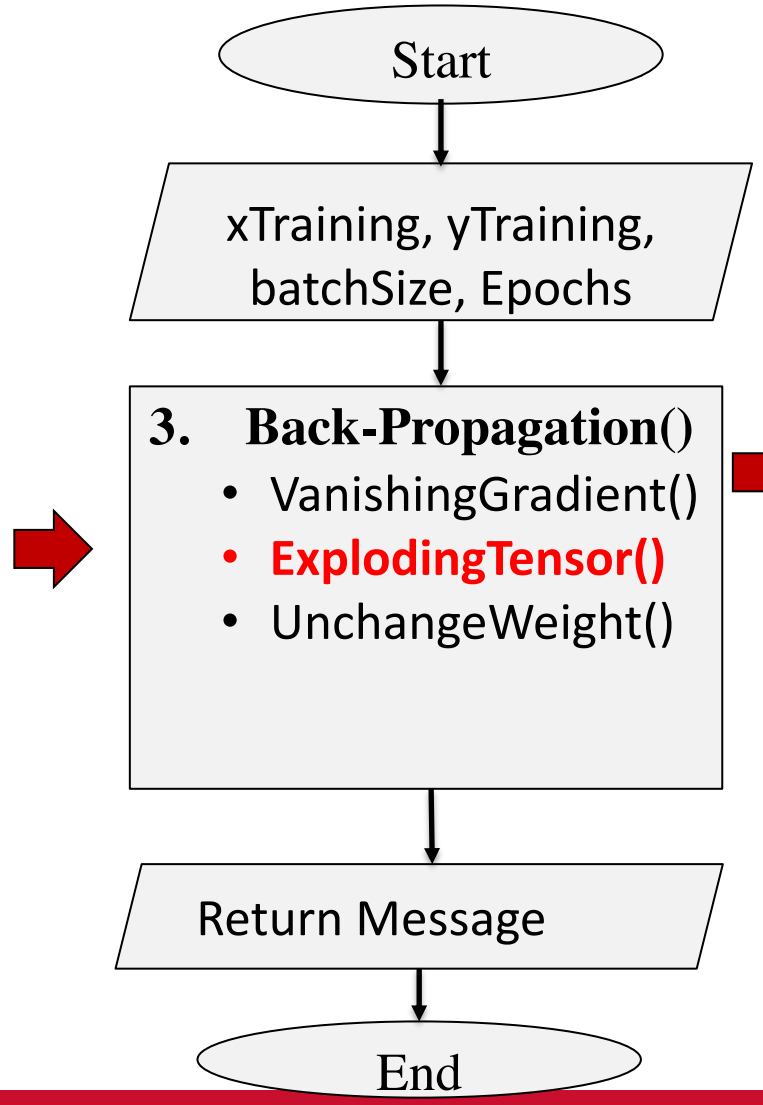


Detects the problem by checking the gradients when they become extremely small or drop to a zero.



VanishingGradient()

Algorithm: Failure Symptoms Detection

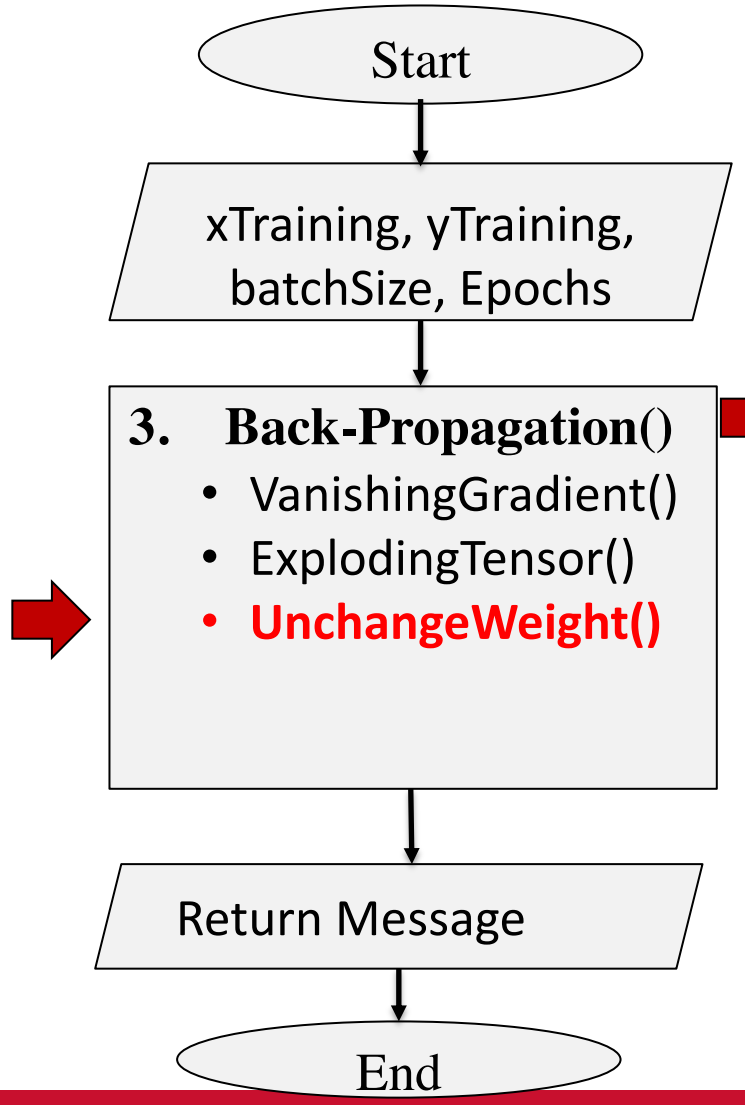


Detects any numerical error such as infinite, NaN (Not a Number), or zero.

```
array([[  1.00000000e+00,   9.82503479e-05,  -2.52892733e-06,
          6.43756948e-07,  -2.64291408e-06,  -3.18651632e-06,
                     nan,              nan,              nan,
          0.00000000e+00,              nan,   5.08329823e-09,
          8.60062854e-10,   1.93595340e-09,   0.00000000e+00,
         -1.24836730e-09,   3.83645737e-09,  -3.90316446e-09,
                     nan,  -7.00323994e-07],
       [ -7.27595761e-11,   7.15255737e-07,  -2.68207733e-02,
         -6.68754578e-01,   1.68675050e-01,  -2.37232931e-02,
```

ExplodingTensor()

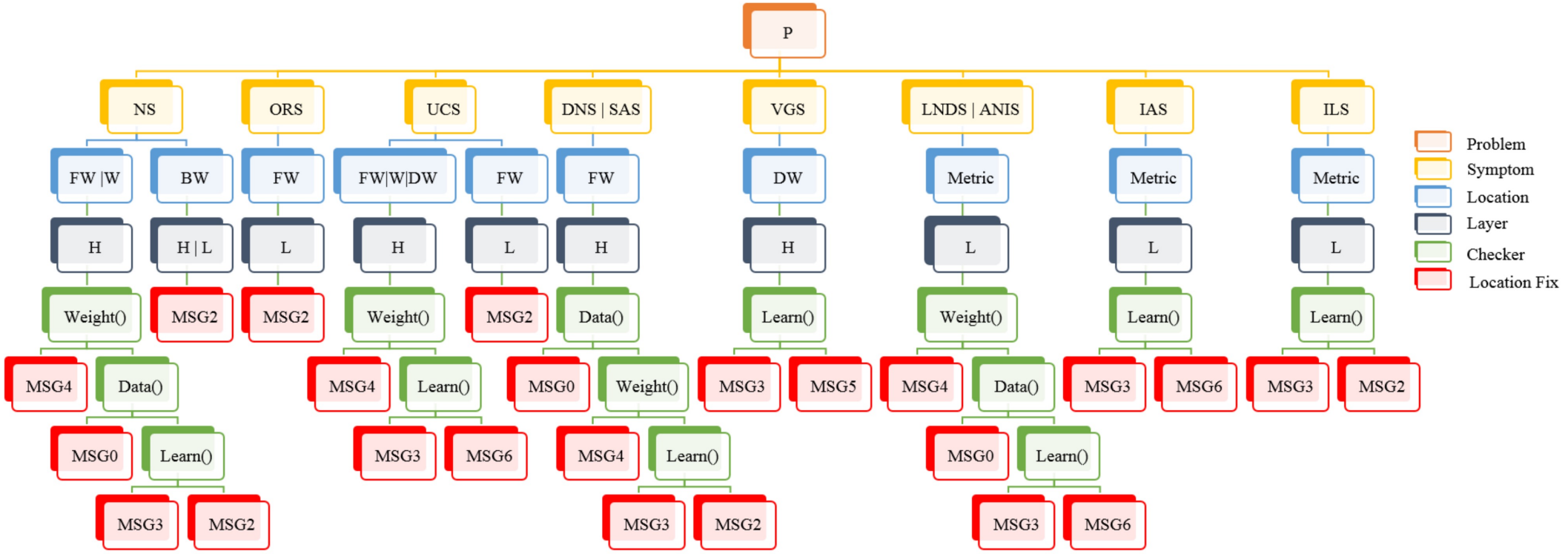
Algorithm: Failure Symptoms Detection



Detects if value is not changing by comparing the value for the current step with the mean value that stores in previous (N = 5) steps.

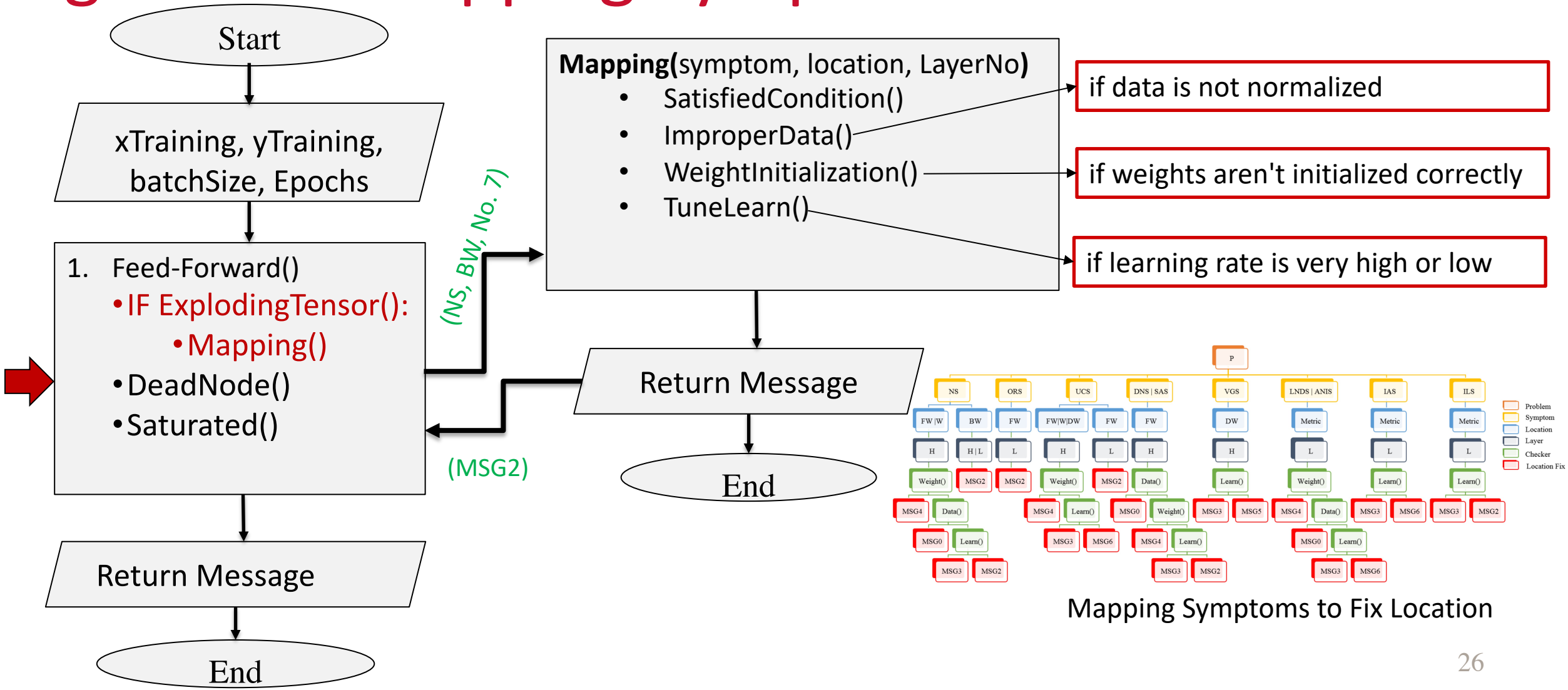
UnchangeWeight()

Decision Tree



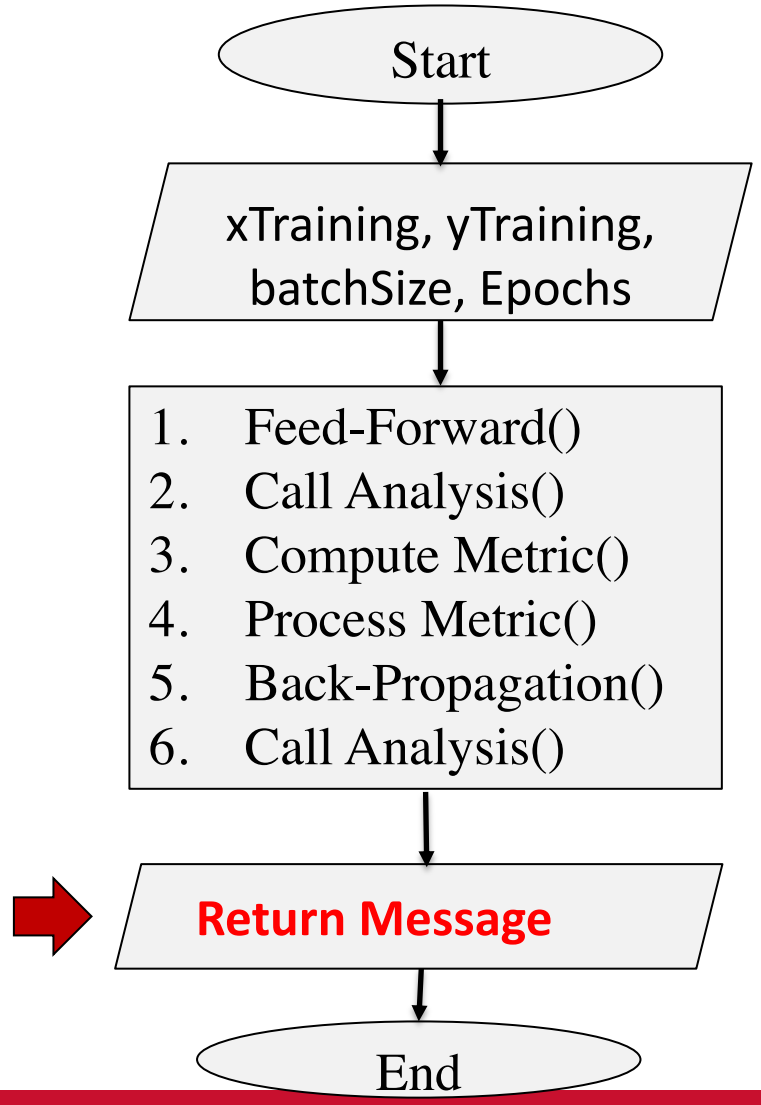
Mapping Symptoms to Fix Location

Algorithm: Mapping Symptoms to Locate Fix



Mapping Symptoms to Fix Location

Algorithm: Failure Symptoms Detection



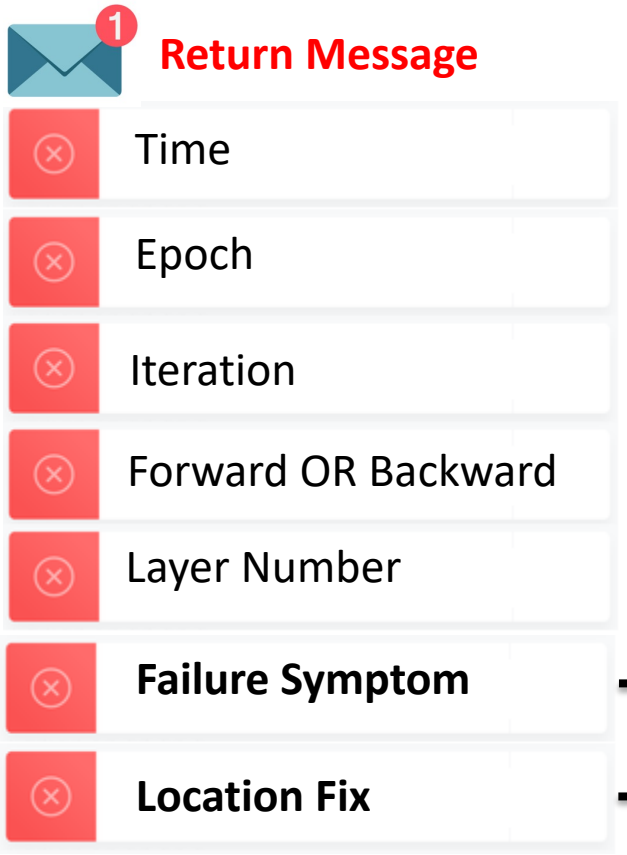


TABLE III: Abbreviation of Failure Symptoms

No	Symptoms	Abbreviation
1	Numerical Errors	NS
2	Unchanged weight	UCS
3	Saturated Activation	SAS
4	Dead Node	DNS
5	Out of Range	ORS
6	Loss Not Decreasing	LNDS
7	Accuracy Not Increasing	ANIS
8	Vanishing Gradient	VGS
9	Invalid Loss	ILS
10	Invalid Accuracy	IAS
11	Correct Model	CM

TABLE IV: Abbreviation of Actionable Changes

No	Message Guideline	Abbreviation
1	Improper Data	MSG0
2	Change the loss function	MSG1
3	Change the activation function	MSG2
4	Change the learning rate	MSG3
5	Change the initialization of weight	MSG4
6	Change the layer number	MSG5

Research Questions

Validation

RQ1: Can our technique **localize faults and report the symptoms** in deep learning programs effectively?

Comparison

RQ2: How **effectively** and **how fast** can our technique localize the faults and report the symptoms compared to existing methodologies?

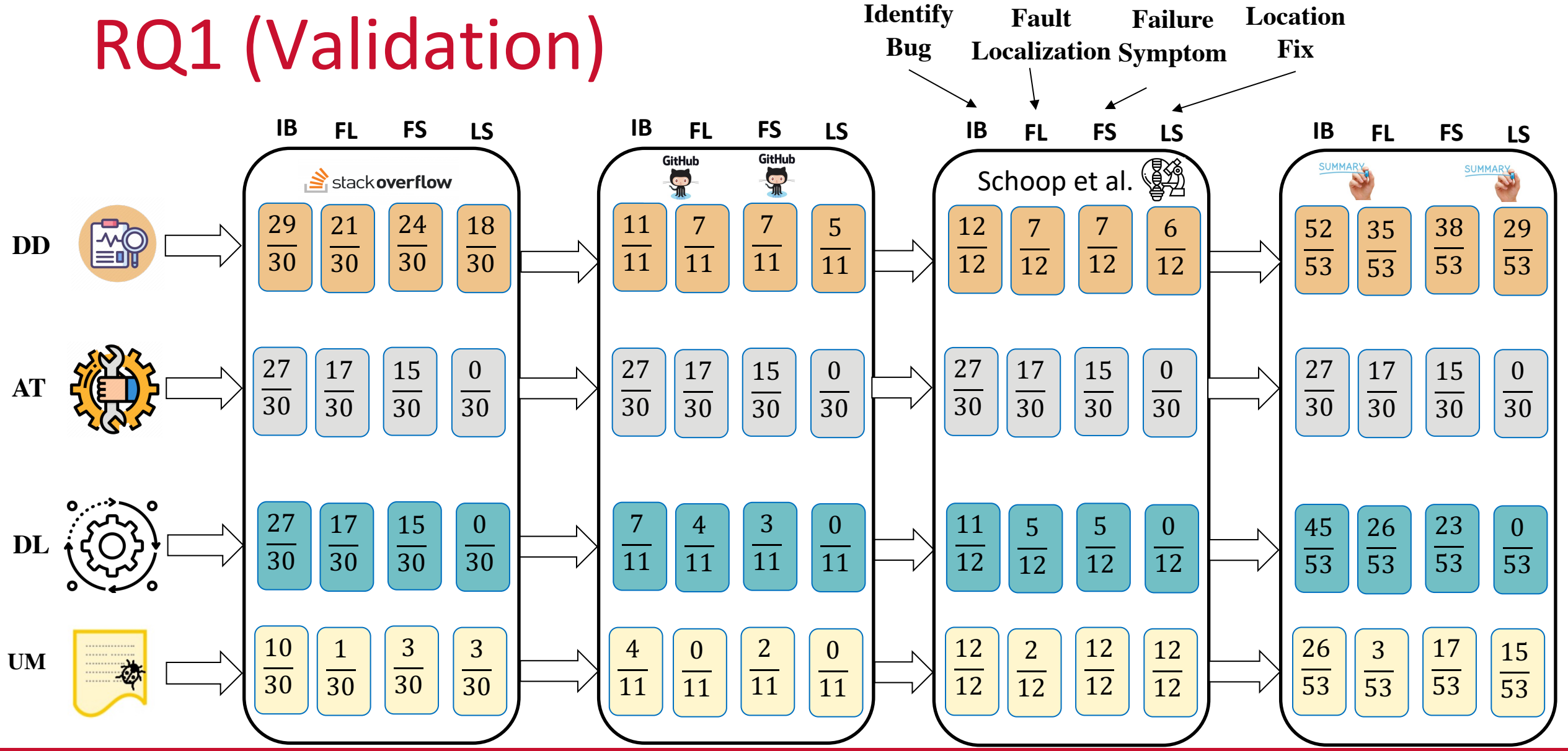
Limitation

RQ3: In which cases does our technique **fail to localize** the faults and report the symptoms?

Ablation

RQ4: To what extent does each component of our approach **contribute to the overall performance**?

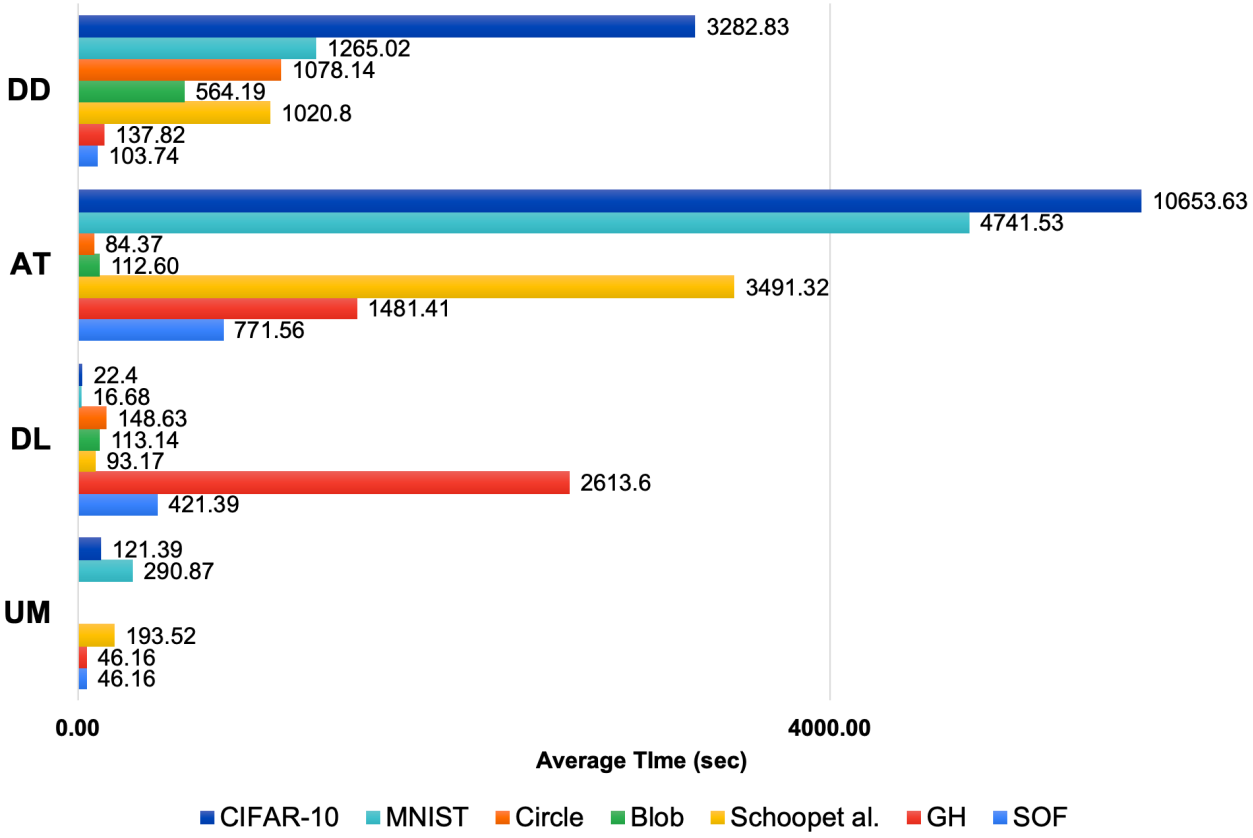
RQ1 (Validation)



RQ2 (Comparison)

We measured the analysis time for:

- **UMLAUT (UM), DeepLocalize (DL), AUTOTRAINER (AT),** and **DeepDiagnosis (DD)**
- Using Stack Overflow (SOF), GitHub (GH), UM, and AT benchmarks.



DeepDiagnosis is the most efficient for Stack Overflow and *Schoop et al.*'s model.

- **Slower** than UMLUAT on the GitHub models.
- **Faster** than AT on all benchmarks except for the Blob and Circle datasets.

Our runtime overhead mainly is the online dynamic analysis performed on the internal parameters.

RQ3 (Limitation)

- Our prototype does not support all Keras APIs such as ***fit_generator()*** instead of ***fit()*** function.

- Our technique does not support Recurrent Neural Networks (RNNs) models.

- Our approach does not support some hyperparameters such as the batch size, epoch, and dropout rate.

RQ4 (Ablation)

1. The result shows which **procedure** report the **symptom** in buggy model:

Dataset	Exploding	Saturated	DeadNode	OutofRange	UnchangeWeight	InvalidAcc	AccNotIncreasing	VanishingGradient
Stack Overflow, GitHub, Schoop et al.	23	7	5	5	2	2	1	1
AUTOTRAINER	50	5	8	0	23	0	21	31
Total	73	12	13	5	25	2	22	32

2. Our study found that **data preparation** is a frequently occurred issue.
3. **SGS** benchmark does not have a very deep model that contains **many layers**.
4. **VanishingGradient()** is invoked very **frequently** in **AUTOTRAINER** models, because this dataset has **many layers** using **sigmoid** and **tanh** as activation functions.

Conclusions and Future Work

We put forward an approach dynamically detecting bugs in DNNs that uses a map between symptoms to root causes.

We compared DeepDiagnosis against SoTA using several benchmarks.
The results show that DeepDiagnosis is both more comprehensive and efficient than prior work.

Future work includes adding support to automatically:

Detecting additional bug types in other model architectures

Repairing models without developer intervention

DeepDiagnosis: Automatically Diagnosing Faults and Recommending Actionable Fixes in Deep Learning Programs

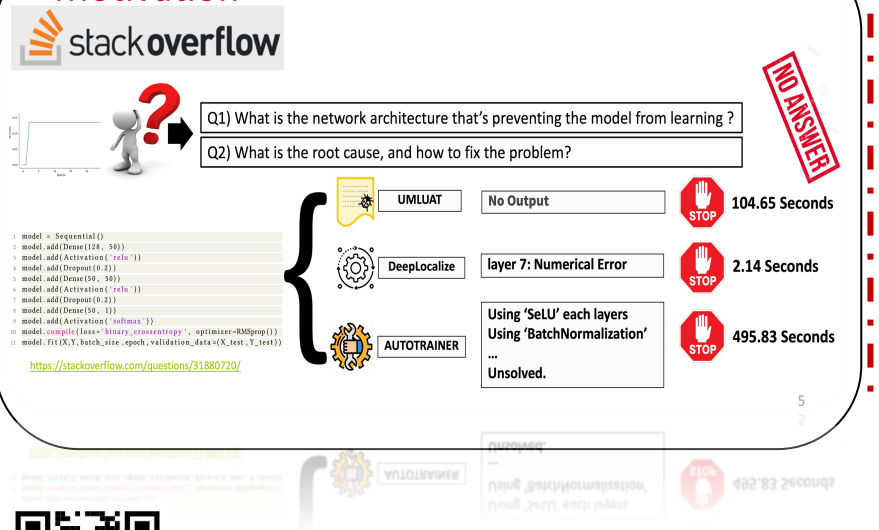
Mohammad Wardat
wardat@iastate.edu
Dept. of Computer Science, Iowa State University
226 Atanasoff Hall, Ames, IA, USA

Breno Dantas Cruz
bdantasc@iastate.edu
Dept. of Computer Science, Iowa State University
226 Atanasoff Hall, Ames, IA, USA

Wei Le
weile@iastate.edu
Dept. of Computer Science, Iowa State University
226 Atanasoff Hall, Ames, IA, USA

Hridesh Rajan
hridesh@iastate.edu
Dept. of Computer Science, Iowa State University
226 Atanasoff Hall, Ames, IA, USA

Motivation



Contributions

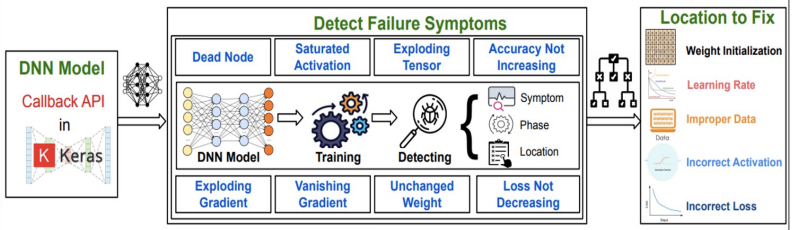
We study the symptoms and their root causes to propose a mapping algorithm of symptoms to fix.

We provide a benchmark with 444 models that practitioners can use to evaluate their techniques.

Our prototype is more comprehensive and efficient than:

- UMLUAT
- DeepLocalize
- AUTOTRAINER

DeepDiagnosis Overview



Preprint: arxiv.org/abs/2112.04036

Artifacts: github.com/DeepDiagnosis/ICSE2022