# Mutation-Based Accuracy Improvements in Neural Networks using Spectrum-Based Fault Localization

Lennart Mühlhahn

Institute for Software Systems

April 10, 2024

# Contents

Introduction

Motivations

Work

Results

# Introduction

- Neural Networks increasingly deployed in applications, often in safety critical domains:
  - Autonomous Vehicles
  - Medical Diagnostics
- How can we ensure reliable Networks?

# Neural Networks

- Neural networks, machine learning algorithms inspired by the brain.
- They consist of interconnected neurons organized in layers.
- Each neuron receives input, processes it, and generates an output signal.
- Neural networks can learn complex patterns in data and make predictions based on them.

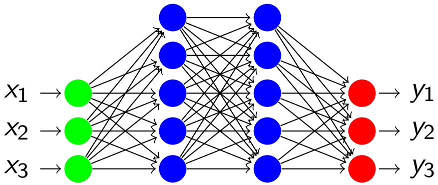



Figure: Neural Network

# Training a Neural Network

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right)$$

- **Forward Propagation**: Input data is passed through the network to generate predictions.
- **Backpropagation**: Error is calculated and propagated backward, to adjust the weights and biases.
- This process is repeated iteratively until the network's predictions are sufficiently accurate.

# Spectrum-Based Fault Localization

- Technique for identifying faults by analysing program execution
- Ochiai meassure:

$$\frac{N_{cf}}{\sqrt{(N_{cf} + N_{nf}) \cdot (N_{cf} + N_{cs})}}$$

- $N$ statement, $c$ covered, $n$ not covered, $s$ success, $f$ failure
- Covered meaning executed
- Challenge: False Positives

# DeepFault

- First Spectrum-Based Fault Localization for Neural Networks, using
- Aim: Identifying not rightly calibrated neurons
  - By constructing a hit spectrum matrix
  - Ranking neurons by suspiciousness value
- Train those with synthesized inputs

# Our Aim.

- Identifying suspicious neurons and mutating them to improve the network
- Mutating by changing weight or bias values or assigning completely new ones
- Improving trained models, for the accuracy and loss
- Retrain neurons of trained models

# Our Contribution

- Introduced savings for the hit spectrum and ranked neurons, to reduce redundant computations
- A function to take a model and modify it, according to the ranking
- Random Choosing of Neurons
- Modification functions, that modify a weight or bias
  - Assign a value, random or a set value
  - Scale by a value, random or a set value
  - Delete a neuron
- Updated libraries to current versions

# Our Contribution

- Break conditions, that are enacted when model is worsening in regards of the:
  - Loss
  - Accuracy
  - Loss or Accuracy
  - Loss and Accuracy
- An offset for the loss and accuracy break condition, to prevent premature termination
- A regression function for the offset to lead to faster breakage.

# Flow Chart

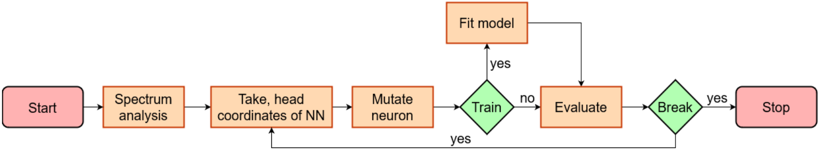



Figure: Flow Chart of Algorithm

# Evaluation

- 4 Architectures, 2 Convolutional and 2 Deep Neural Networks
- Dataset: Fashion-MNIST
- Trained: With full, half and quarter dataset
- Aggregated over 160000 Datapoints
- CL: Convoulutional Layer, stride (1,1), kernel size (3,3), pooling layer (2.2)

| Model Name | Mod. Param. | Architecture |
|---|---|---|
| DNN1 | 16 | $< 16 >$ |
| DNN2 | 64 | $< 4x16 >$ |
| CNN1 | 12 | $< CL, 4 >$ |
| CNN2 | 20 | $< CL, CL, 4 >$ |

# Parameters

- **Similarity coefficient:** tarantula, dstar with value 3, ochiai and random
- **Mutation functions:** `modify_weight_one_random_gauss`, `modify_weight_all_random_gauss`, `modify_bias`, `modify_bias_random_gauss`, `modify_all_weights`, `modify_all_weights_by_scalar`, `modify_all_weights_by_scalar_random_gauss`, `modify_weight_all_random_by_scalar_gauss`
- **Break conditions:** loss, accuracy, loss and accuracy, loss or accuracy
- **Loss offset:** 0.005, 0
- **Accuracy offset:** 0.01, 0
- **Loss and accuracy regression:** True for all runs
- **Values:** -1, -0.5, 0, 0.5, 1
- **Sigma for random:** 0.5, 1

# Results

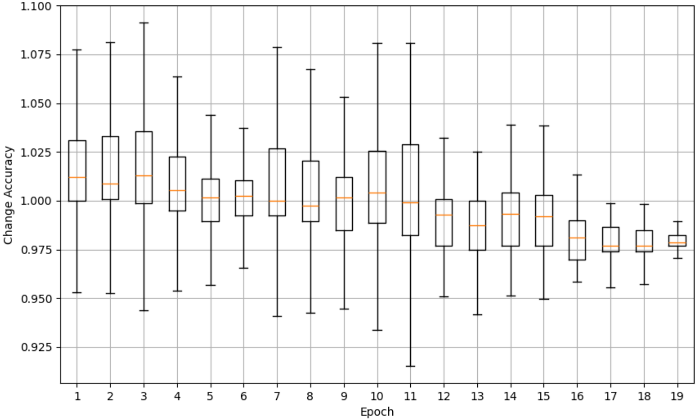



Figure: Accuracy of the models, with training

# Results

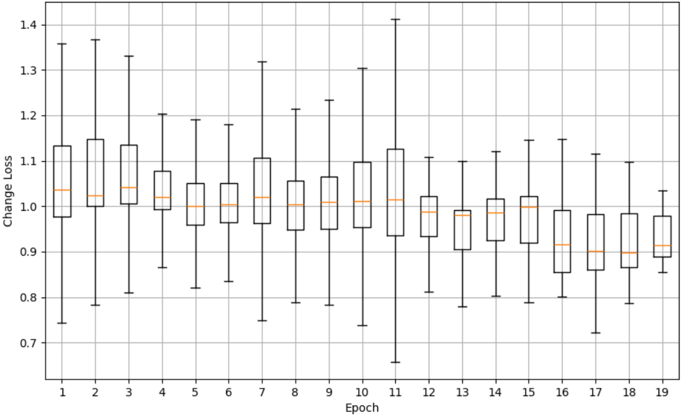



Figure: Loss of the models, with training

# Results

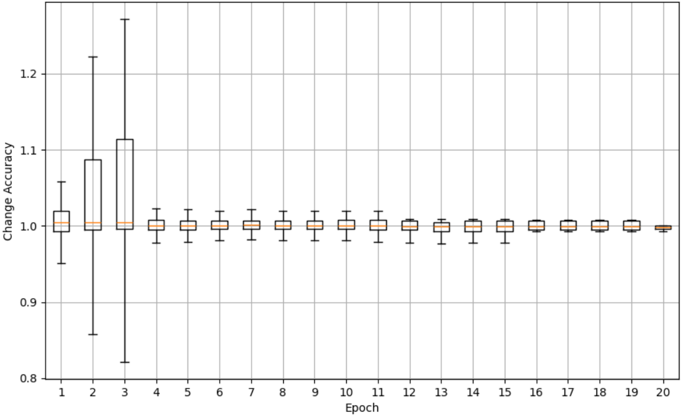



Figure: Accuracy of the models, without training

# Results

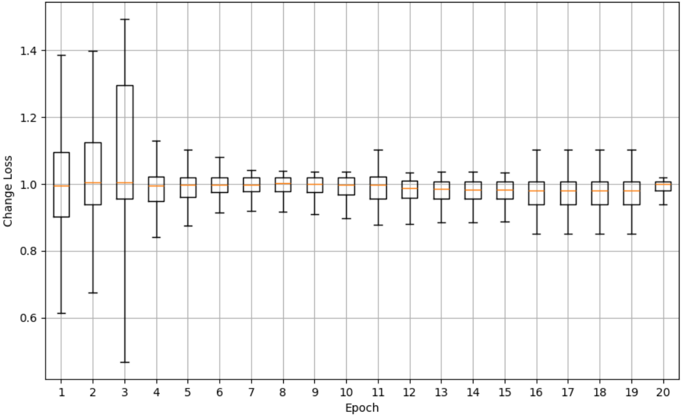
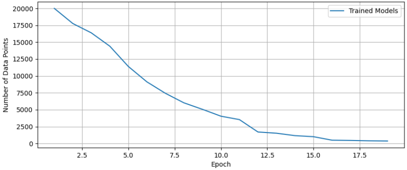



Figure: Loss of the models, without training

# Results



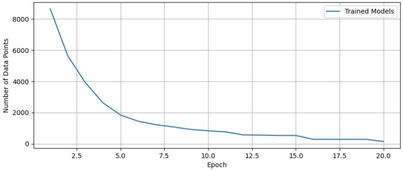




Figure: Trained and Not Trained data points.

In the further evaluation epochs where reduced to:

- Trained: 20 epochs
- Not trained: 10 epochs

# Some more Results

- Works best for models with less epochs, for 1 pre-trained epochs then 6 pre-trained epochs.
- Better results for the full and half dataset, quarter a worsening outlook.
- Tarantula is the most promising measure
  - For the trained runs, the three other aren't much different
  - For the untrained runs, we see ochiai, d-star then random.
- Performs better for CNN models
  - Could be due to the larger number of values.

# Some more Results

- The Offset seems to be essential for the performance, and yields far better results
- Breaking on Accuracy more important, even for the loss
  - solely on accuracy or heavily on accuracy
- Weight functions better performing, than bias functions
  - No difference between random and a set value
  - Caution, only Gauss variables were examined
- The value used in the mutation has no significant impact

# Future Work

- Testing the approach on a larger model
- Testing on different datasets
- Elicit a ratio of modifiable neurons
  - Using the process on multiple different models and datasets.
  - How many neurons of a model can we mutate before damaging it?
- Using a dynamic spectrum matrix.
  - To account for changes in the model during execution of the model.

# Future Work

- Genetic programming derived suspiciousness measures
- Aggregate layer suspiciousness and modify the whole layer or add another layer
  - before or after the most suspicious layer
- Use localised neurons and try to fix it by identifying problem exploiting phenomena like:
  - dying ReLu.
  - vanishing and exploding gradients