Software Engineering for Artificial Intelligence

TECHNISCHE UNIVERSITÄT DARMSTADT

Debugging

Constantin Stipnieks & Florian Busch

Outline

- Introduction: Debugging in AI
- Debugging via State Differential Analysis
- Debugging via Decision Boundaries
- Model assertions
- Visualization Tools
- Summary
- References

Debugging in AI [1]

TECHNISCHE UNIVERSITÄT DARMSTADT

Machine learning (ML) models are hardly ever without mistakes.

Mistakes can be very dangerous/costly:

- Financial risks
- Legal risks
- Ethical problems (biases)

Debugging (non ML specific definition)

“to remove bugs (= mistakes) from a computer program” (Cambridge Dictionary, accessed on 25.06.2020)

Debugging in AI [1]
Failure models and model investigation

TECHNISCHE UNIVERSITÄT DARMSTADT

Failure models:

Many reasons, a model might not behave as intended, e.g. opaqueness, social discrimination, security vulnerabilities, privacy harms, model decay

Model investigation:

Sensitivity analysis: inspect model behavior on unseen (constructed) data

Residual analysis: inspect model errors (numeric)

Benchmark models: compare to well established benchmark models

ML security audits: inspect security of your model

Debugging in AI [1]
Improving your model (1/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Improving your model:

Data generation

- Create new data to avoid learning unwanted biases from the original dataset (representative data distribution)

Interpretable models

- Use interpretable models if possible, make models explain their predictions

Model editing

- In certain models, changes can be made by hand (e.g. decision trees)

Debugging in AI [1]
Improving your model (2/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Model assertions

- Business rules put on top of model predictions

Discrimination remediation

- Take steps to ensure the system is not discriminatory

Model monitoring

- Monitor the models behavior, it will most likely change over time

Anomaly detection

- Inspect behavior of the model on strange input data and for strange predictions (e.g. use constraints)

Model Bugs in Neural Networks

There are two types of model bugs:

- Structural Bugs
  - e.g. number of hidden layers and neurons, neuron connections
- Training Bugs
  - e.g. using biased training data, that does not follow the real world data distribution
  - results in over- or underfitting
  - can only be fixed by using more training samples that correct the bias

Model Bugs in Neural Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

There are two types of model bugs:

- Structural Bugs
  - e.g. number of hidden layers and neurons, neuron connections
- Training Bugs
  - e.g. using biased training data, that does not follow the real world data distribution
  - results in over- or underfitting
  - can only be fixed by using more training samples that correct the bias

Fixing Training Bugs

Main difficulties of fixing training bugs:

1. Reliably identify the problem in the existing training data
2. Find new samples that fix this problem

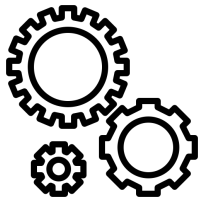
Previous approaches are rather agnostic to the first difficulty and just input any new samples in the hope that it fixes the problem

Before we delve into a solution for 1., let us consider were new training data can come from

Acquiring additional training data

In general there two main methods to get more training data

Artificially generating data



*1

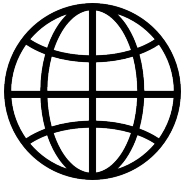
"Best" approach: **Generative Models**

Approximate the real world data distribution

\+ Able to efficiently generate as many new samples as needed

\- Getting a good generative model is hard

Extracting more data from the world



*2

\+ Likely to get good data

\- Can be very time consuming and expensive

*1 Icon made by surang from www.flaticon.com

*2 Icon made by Freepik from www.flaticon.com

Icons accessed 25.06.2020

Debugging via State Differential Analysis [3]
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

The following is an overview of the method described in [**"MODE: automated neural network model debugging via state differential analysis and input selection",** S Ma et al, 2018]

Goal:

- Identify the features responsible for a bug and fix that bug by training on targeted samples

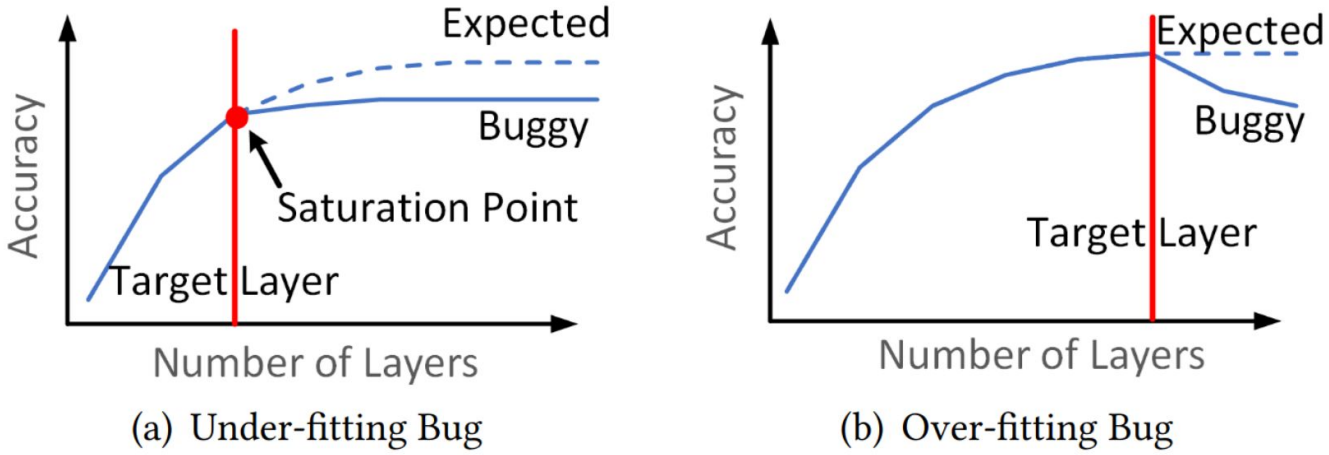
The method can be divided into two main steps:

1. Apply state differential analysis to identify the faulty features
2. Run an input selection algorithm to select samples with substantial influence on the faulty features

Debugging via State Differential Analysis [3]
Method: Layer Selection

TECHNISCHE UNIVERSITÄT DARMSTADT

If we have found an underfitting or overfitting bug, we will first determine the layer where the accuracy takes a turn for the worse.



The features in this layer seem the most promising to investigate, as it is the layer where the accuracy stops improving / decreases.

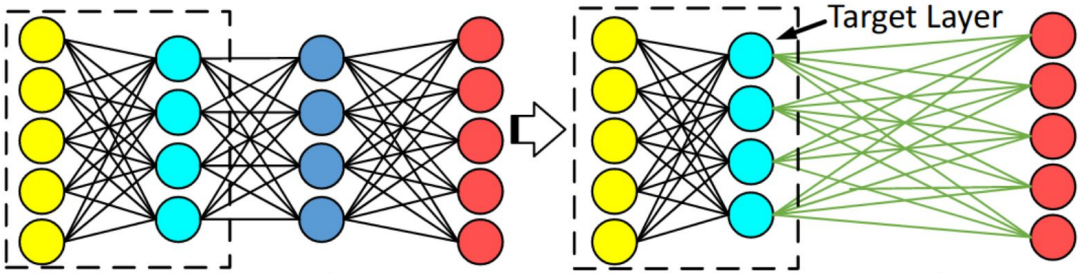
Debugging via State Differential Analysis [3]
Method: Layer Selection

TECHNISCHE UNIVERSITÄT DARMSTADT

Our algorithm for identifying the target layer of an **underfitting** bug for labelIconsists of the following steps:

For each hidden layer L from input to output **do**:

1. Extract the sub-model of all layers up to L
2. Freeze the weights in the sub-model
3. Append a fully connected output layer to L
   (with the same labels as in the original model)
4. Retrain this sub-model on the same training data
5. Compare the test result for labelIwith that of the previous sub-model.
   If they are very similar, the layer before L is the target layer.



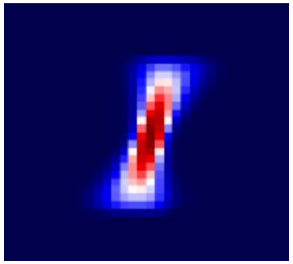
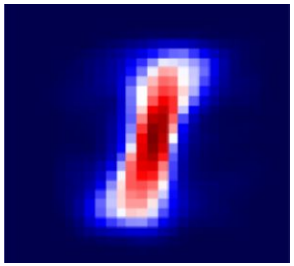
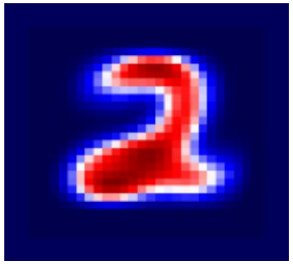
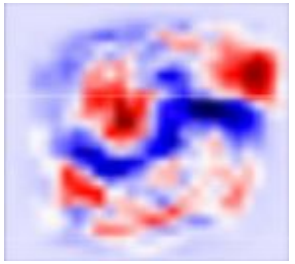
Debugging via State Differential Analysis [3]
Method: Feature Selection

TECHNISCHE UNIVERSITÄT DARMSTADT

Within the target layer we want to identify those features with the highest **importance** for correctly/incorrectly classifying labell.

For a specific input sample, the **feature values** in the target layer tell us the importance of those features for the correct/incorrect classification of that sample.

Given all samples that are correctly/incorrectly classified, we average their feature values and normalize them to [-1,1]. This yields us a **heat map:**

HC_1 =  HM_1 =  HC_2 =  $HC_?$ = 

Values in (0,1] are **red** and denote that the **presence** of the feature is important

Values in [-1,0) are **blue** and denote that the **absence** of the feature is important

Debugging via State Differential Analysis [3]
Method: Feature Selection

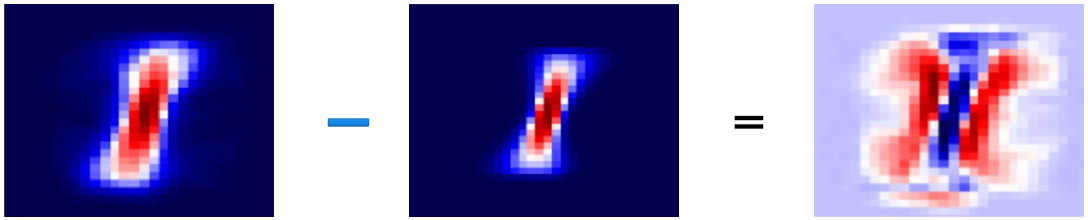
TECHNISCHE UNIVERSITÄT DARMSTADT

Now which features are important to fix an underfitting bug?

We want to **emphasize** the features that are **unique** tol.

To detect those features we calculate the **differential heat map** by subtracting HC_l with HC_k for k ≠l. For instance:

We also want to **suppress** those features that our model thinks are good indicators forlbut in reality aren't (Misclassification tol). To identify those features we subtract HM_l with HC_l :

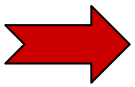


Debugging via State Differential Analysis [3]
Method: Choosing new samples

We can now select new data samples that match those heat maps. Doing so is easy:

1. Run the sample through the model until it reaches the target layer
2. Compare the feature values of the sample with those in the heat map, e.g. by taking the **dot product**
3. If the score is higher than a threshold, use the sample

However, we do not want to overfit on data that only matches the heat maps!



Mix in some randomly selected samples as well.

Debugging via Decision Boundaries [2]
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

The following is an overview of the basic ideas described in [**"Auditing and Debugging Deep Learning Models via Decision Boundaries: Individual-level and Group-level Analysis",** Yousefzadeh, R., & O'Leary, D. P., 2020]

Goal

- Gain knowledge about a deep learning model through its decision boundary

Method

- **Flip points** (next slide)

Outputs

- **Individual-level** auditing: explanation report about a **single** instance
- **Group-level** auditing: information about feature importance and impact (**multiple** instances)

Debugging via Decision Boundaries [2]
Flip points

TECHNISCHE UNIVERSITÄT DARMSTADT

Predictions for two classes, normalized to 1

$$z_1(x) + z_2(x) = 1$$

Positive class

$$z_1(x) < \frac{1}{2}$$

Negative class

$$z_1(x) > \frac{1}{2}$$

Decision boundary
→ points here: **flip points**

$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$

Debugging via Decision Boundaries [2]
Flip points

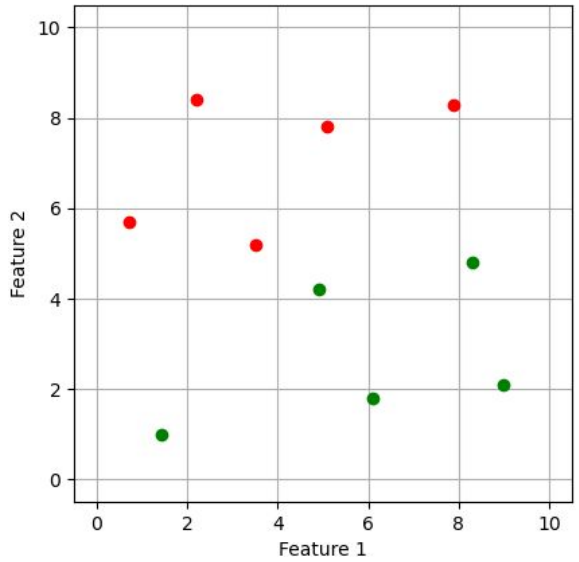
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



Debugging via Decision Boundaries [2]
Flip points

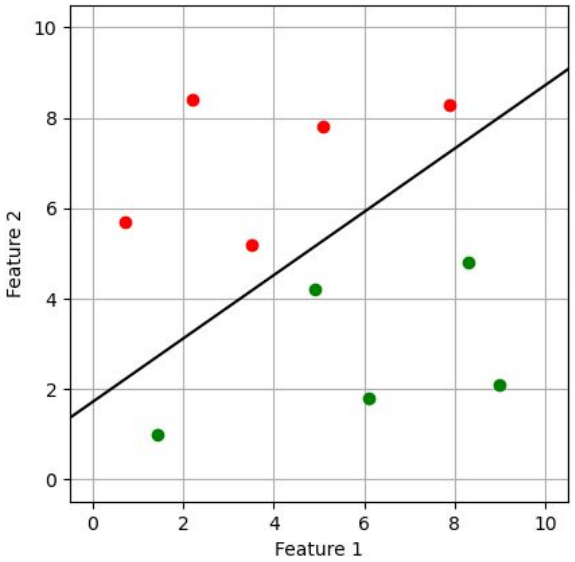
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



Debugging via Decision Boundaries [2]
Flip points

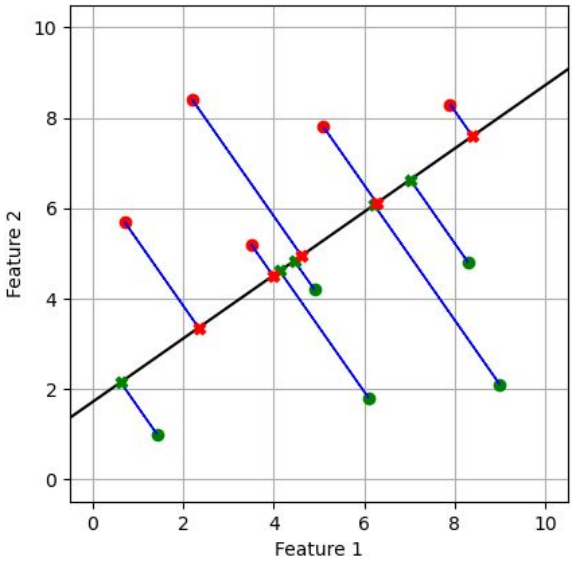
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



Debugging via Decision Boundaries [2]
Flip points

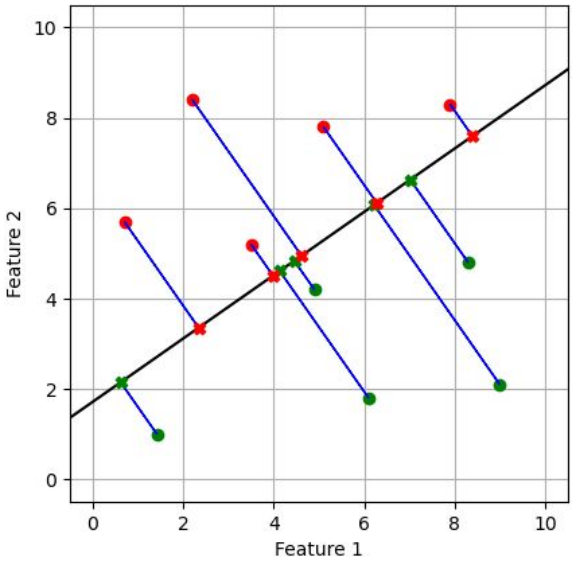
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



flip points

Debugging via Decision Boundaries [2]
Flip points

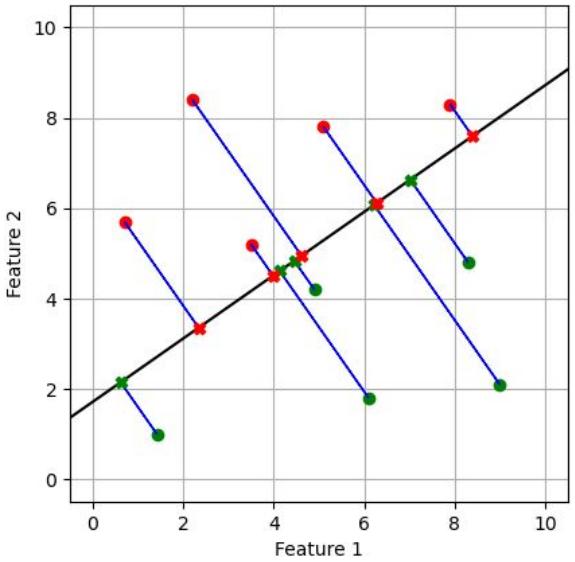
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

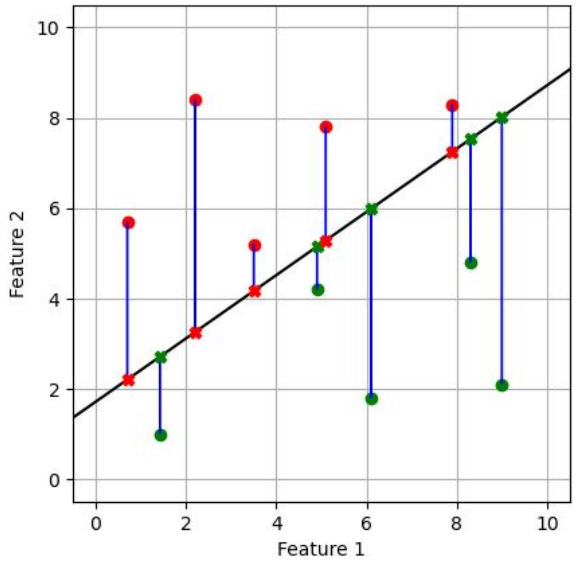
$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



flip points



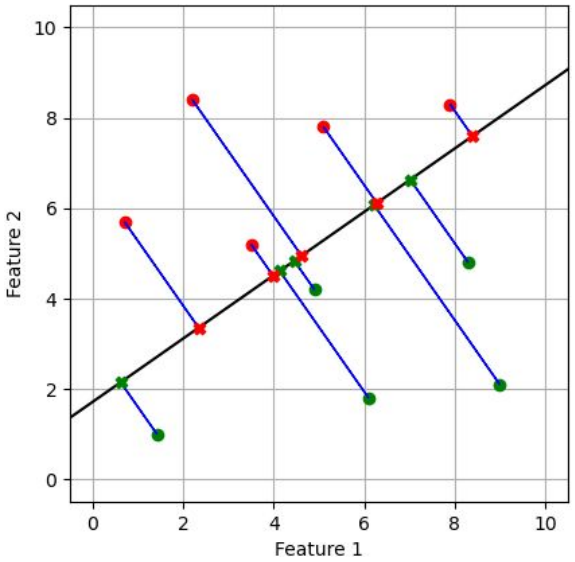
Constrained flip points

Debugging via Decision Boundaries [2]
Flip points

TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

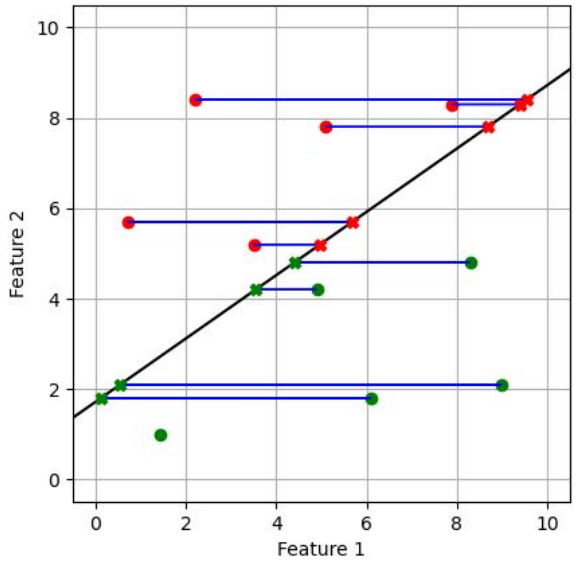
$$z_1(x) = z_2(x) = \frac{1}{2}$$



flip points

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



Constrained flip points

Debugging via Decision Boundaries [2]
Flip points

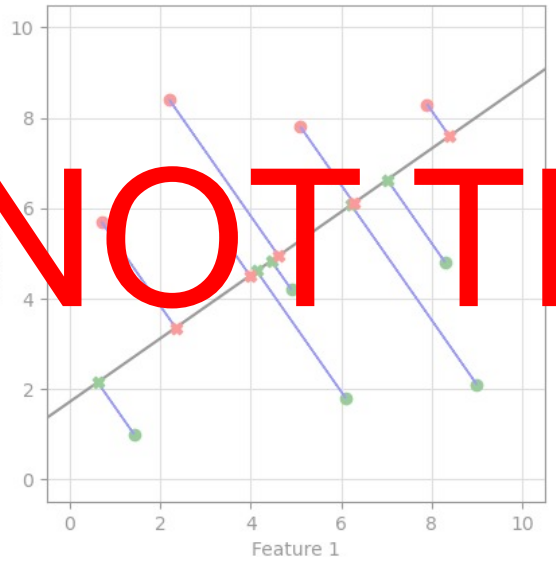
TECHNISCHE UNIVERSITÄT DARMSTADT

Decision boundary
→ points here: **flip points**

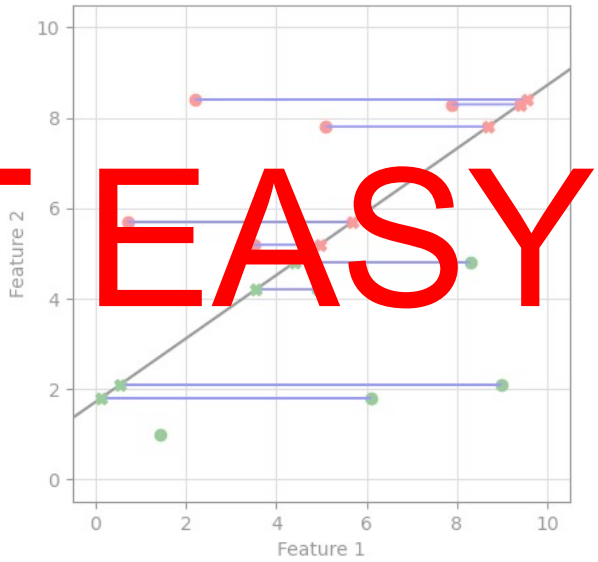
$$z_1(x) = z_2(x) = \frac{1}{2}$$

Closest flip point $\hat{x}$ for x:

$$\min_{\hat{x}} ||\hat{x} - x||, z_1(\hat{x}) = \frac{1}{2}$$



flip points



Constrained flip points

NOT THAT EASY

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

(Based on an example given in the paper)

Imagine a dataset which is about if a person should be given bail or not.

There are 4 features:

- Gender (“male” or “female”)
- Age (*number*)
- Employment status (“employed” or “unemployed”)
- Number of prior arrest (*number*)

The output is either “release with bail” or “not release with bail”

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

(Based on an example given in the paper)

Imagine a dataset which is about if a person should be given bail or not.

There are 4 features:

- **Gender** (“male” or “female”)
- Age (*number*)
- **Employment status** (“employed” or “unemployed”)
- Number of prior arrest (*number*)

The output is either “release with bail” or “not release with bail”

Features values either are **discrete**

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

(Based on an example given in the paper)

Imagine a dataset which is about if a person should be given bail or not.

There are 4 features:

- Gender (“male” or “female”)
- **Age** (*number*)
- Employment status (“employed” or “unemployed”)
- **Number of prior arrest** (*number*)

The output is either “release with bail” or “not release with bail”

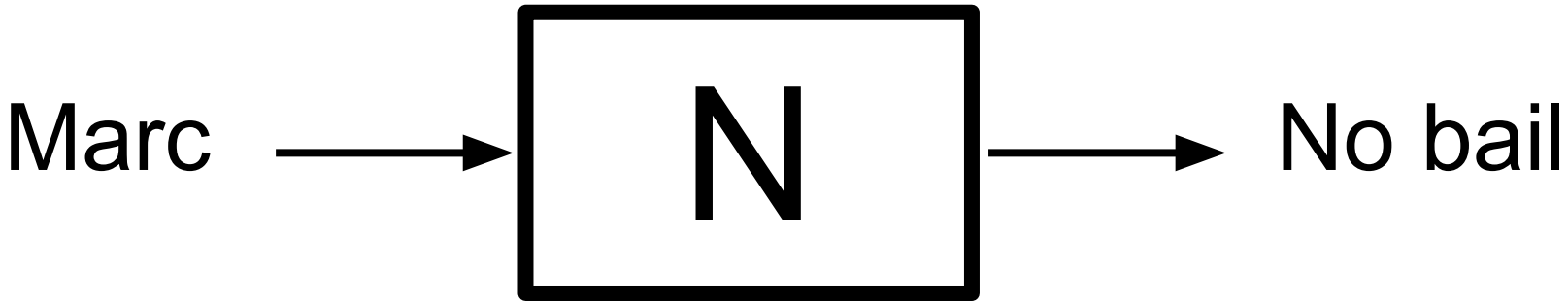
Features values either are discrete or **ordered**

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, unemployed, 2 prior arrests)

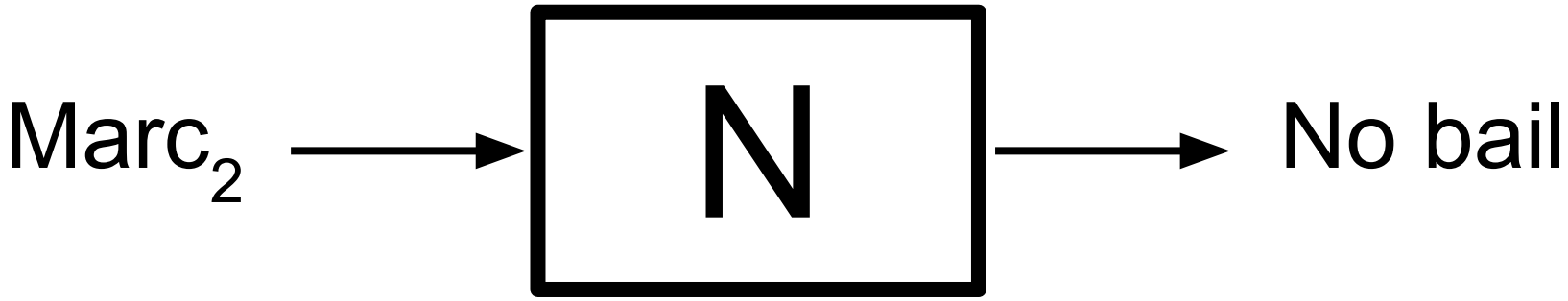


Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (**male**, 40 years old, unemployed, 2 prior arrests)



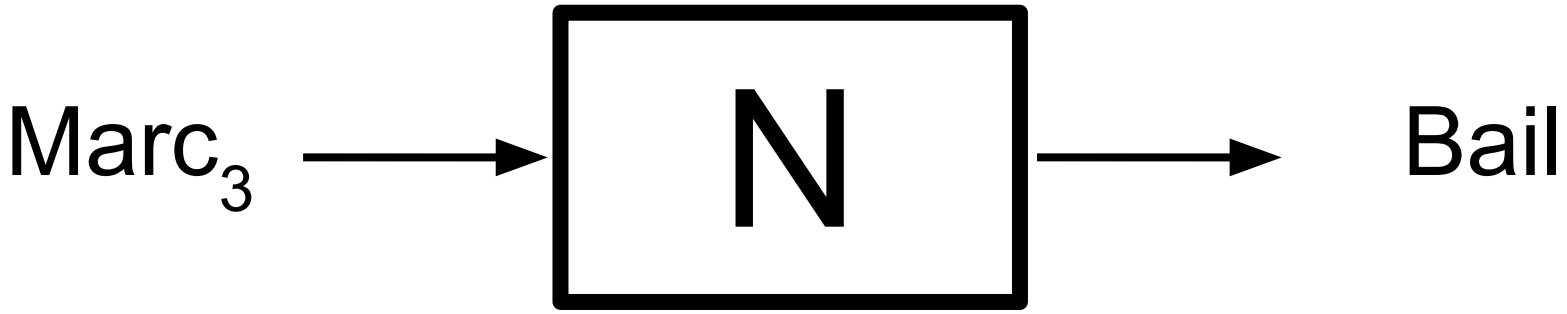
Marc$_2$ (**female**, 40 years old, unemployed, 2 prior arrests)

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, **unemployed**, 2 prior arrests)



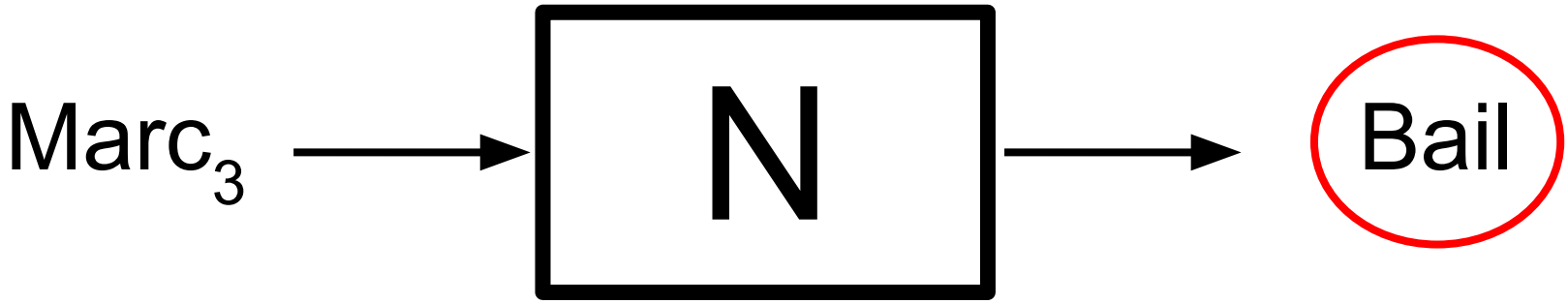
Marc$_3$ (female, 40 years old, **employed**, 2 prior arrests)

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, **unemployed**, 2 prior arrests)



Marc$_3$ (female, 40 years old, **employed**, 2 prior arrests)

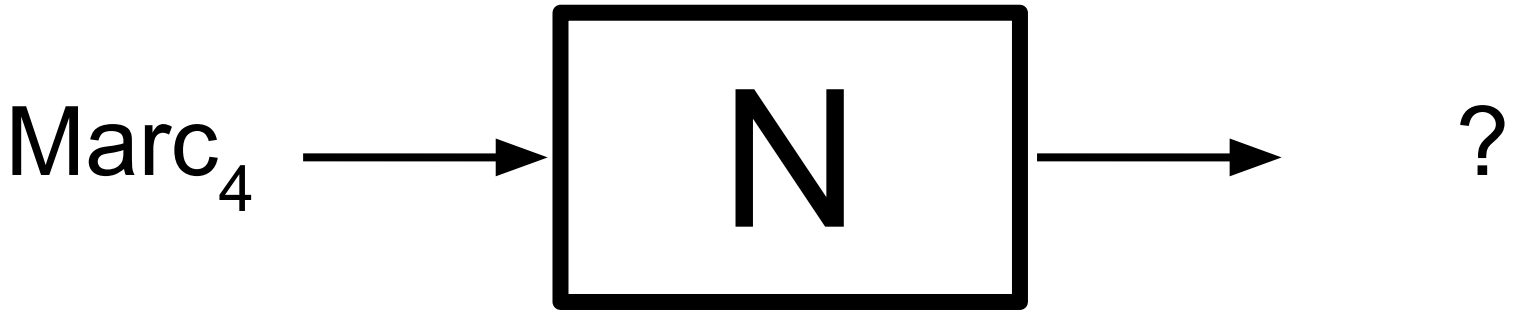
Note: *employment status* = ***employed*** changes prediction

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, **40 years old**, unemployed, 2 prior arrests)



Marc$_4$ (male, **x years old**, unemployed, 2 prior arrests)

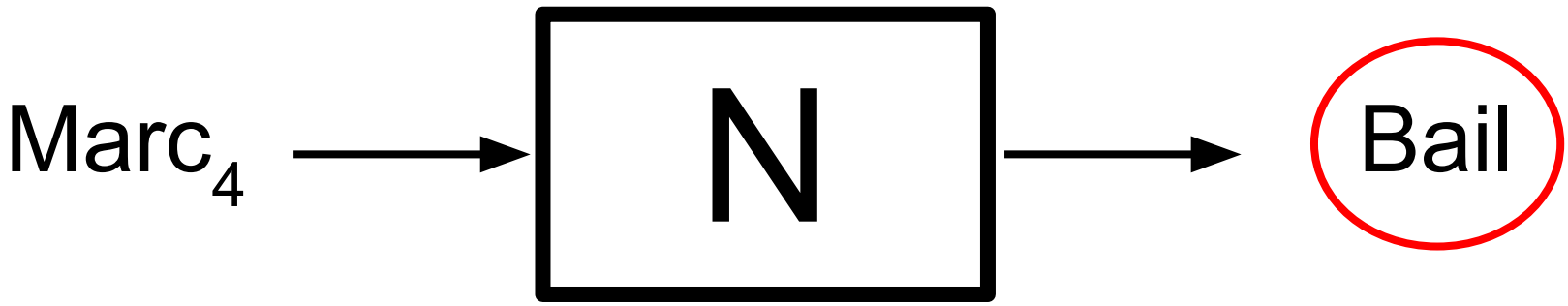
Constrained optimization problem: Find the value of **x** closest to 40, so that ***N*(x) = Bail** and **x** ≥ 0 (and **x** is an integer)

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, **40 years old**, unemployed, 2 prior arrests)



Marc$_4$ (male, **54 years old**, unemployed, 2 prior arrests)

→ **x = 54**

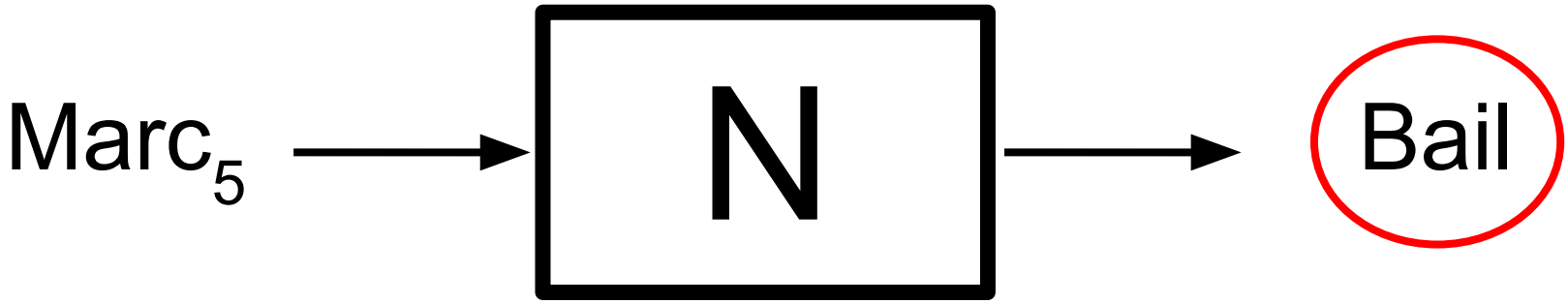
Note: *age* = ***54*** changes prediction

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, unemployed, **2 prior arrests**)



Marc$_5$ (male, 40 years old, unemployed, **0 prior arrests**)

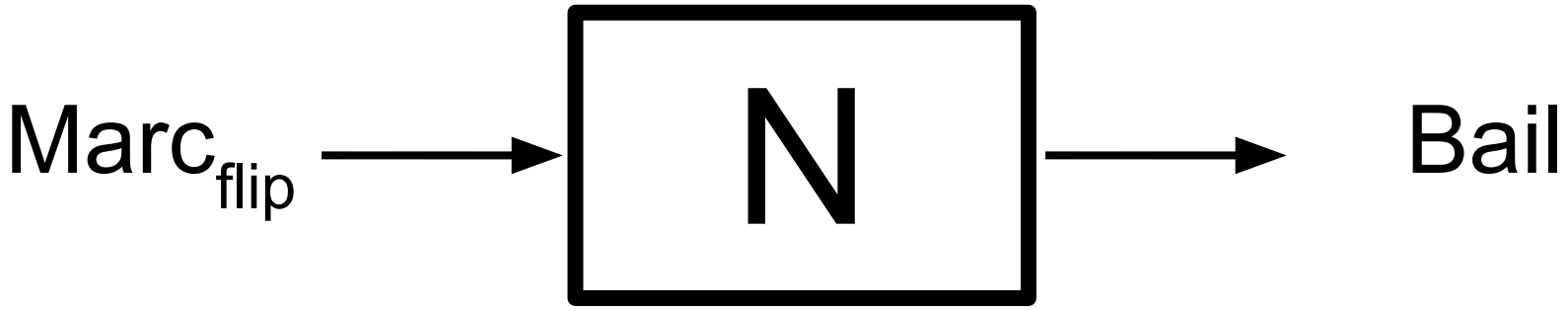
Note: *prior arrests* = ***0*** changes prediction

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, unemployed, 2 prior arrests)



Unconstrained flip point:

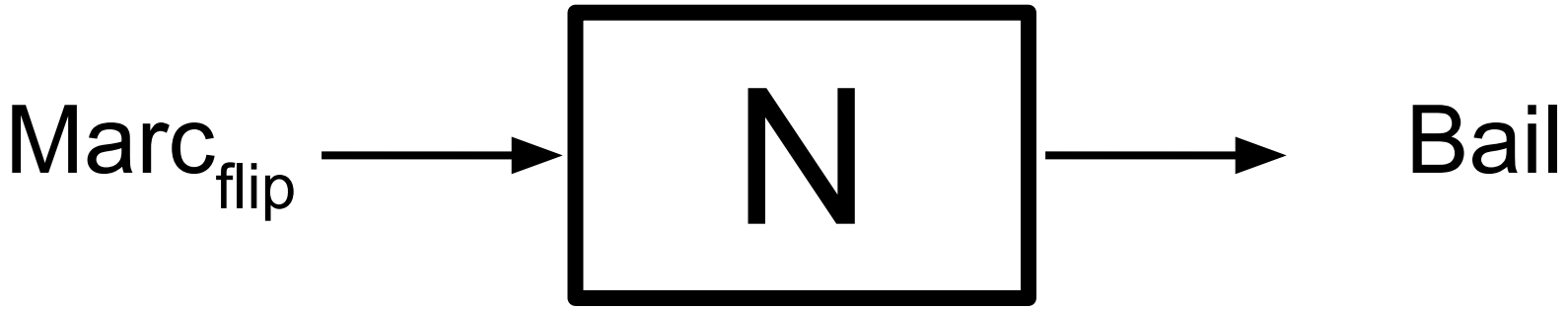
Marc$_{flip}$ (x_1, x_2 years old, x_3, x_4 prior arrests)

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Deep learning model ***N*** is trained on a large set of data points

New data point: **Marc** (male, 40 years old, unemployed, 2 prior arrests)



Unconstrained flip point:

Marc$_{flip}$ (male, 45 years old, unemployed, 1 prior arrests)

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Report

Based on the following facts:

Marc: male, 40 years old, unemployed, 2 prior arrests

The model recommendation for Marc is: No Bail

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Report

Based on the following facts:

Marc: male, 40 years old, unemployed, 2 prior arrests

The model recommendation for Marc is: No Bail

The recommendation would remain No Bail if Marc

- was female (gender = female)
- ***or*** had 1 prior arrest

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Report

Based on the following facts:

Marc: male, 40 years old, unemployed, 2 prior arrests

The model recommendation for Marc is: No Bail

The recommendation would change to Bail if Marc

- was employed
- ***or*** had 0 prior arrest
- ***or*** was 54 years old

Debugging via Decision Boundaries [2]
Individual-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Report

Based on the following facts:

Marc: male, 40 years old, unemployed, 2 prior arrests

The model recommendation for Marc is: No Bail

The smallest change in features that would change the prediction to Bail is if Marc

- had 1 prior arrest
- ***and*** was 45 years old

Debugging via Decision Boundaries [2]
Group-level auditing

TECHNISCHE UNIVERSITÄT DARMSTADT

Possible way to do group-level auditing:

- Compute flip points for all data point:
  - data point matrix ***D*** → flip point matrix ***B***
- Calculate the difference of ***D*** and ***B***, i. e. the direction from data to flip points → flip direction matrix ***F***
  - ***F*** = ***B*** - ***D***
- Now you can analyse ***F***
  - identify most/least influential features
  - study feature dependency
- Model debugging: alter the decision boundary
  - add and teach constrained flip points or flip points with a special flip label to impact the decision boundary

Debugging via Decision Boundaries [2]
Group-level auditing - Example

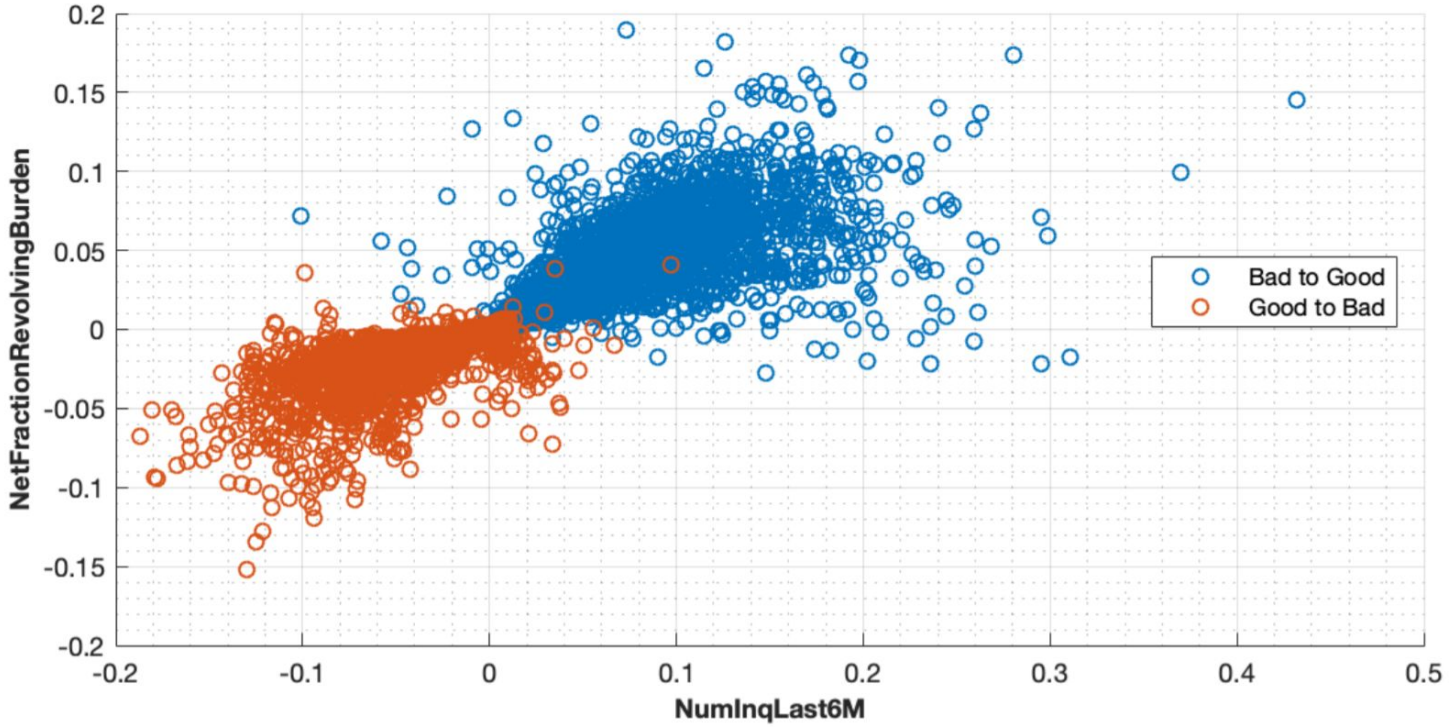


Figure 3: Directions between the inputs and their closest flip point for two influential features. Points are distinctly clustered based on the direction of the flip.

Model Assertions for Debugging Machine Learning [4]

Idea: Model assertions to find errors → can use those to improve your model

(a) Frame 1, base SSD (b) Frame 2, base SSD (c) Frame 3, base SSD

(d) Frame 1, base SSD (e) Frame 2, fixed frame (f) Frame 3, base SSD

Figure 1: **Top row:** example of flickering in three consecutive frames of a video. The object detection method failed to identify the car in the second frame. **Bottom row:** example of correcting the output of a model. The car bounding box in the second frame can be automatically filled in using the frames around it. Best viewed in color.

Visualization Tools

Training machine learning models is **expensive** in time and energy

If something goes wrong in the training, we want to stop asap

Keeping track of all previous experiments helps for debugging

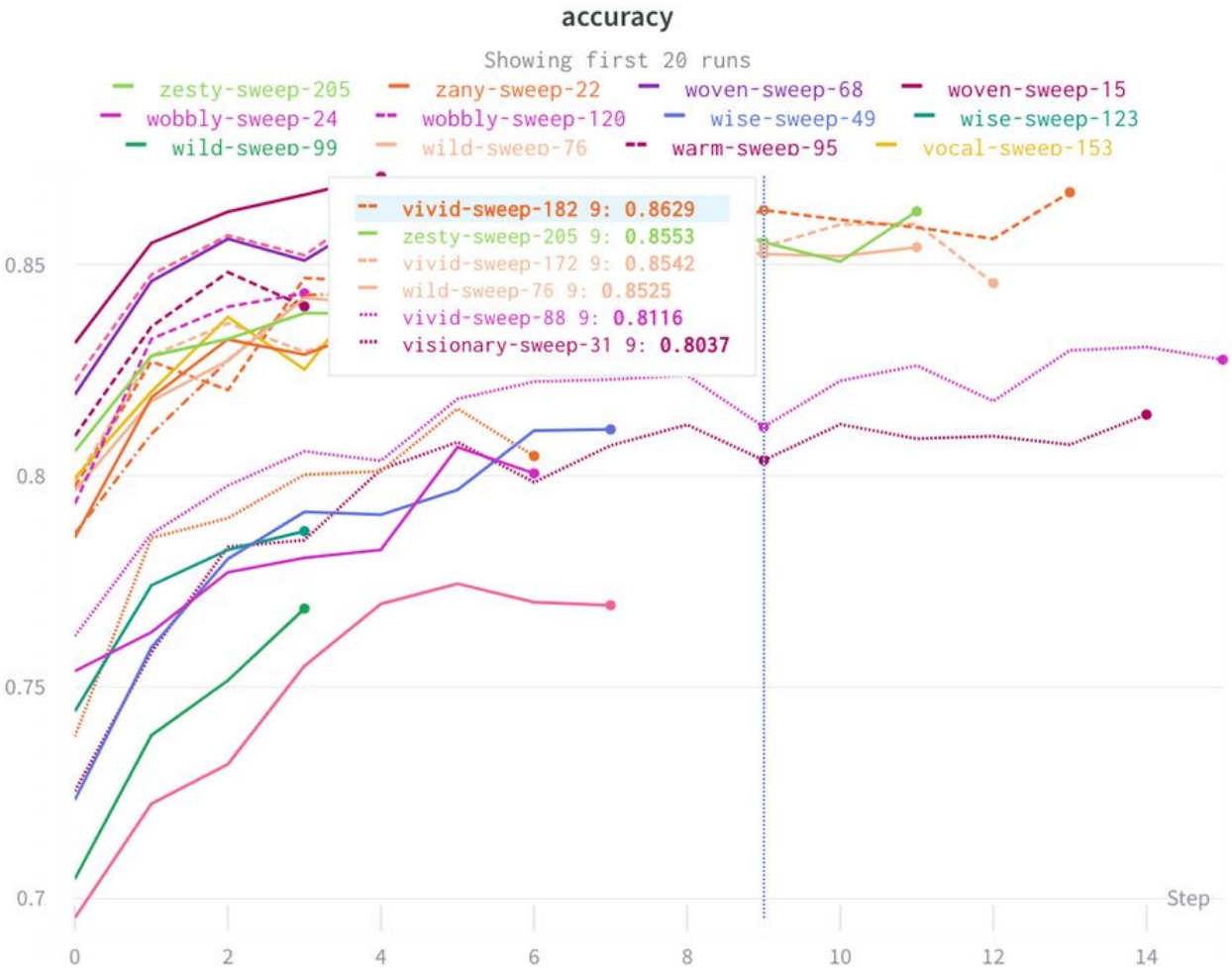
People often try to remember their results or write them down => cumbersome!

A lot of potentially useful data can get lost like that

It is good practice to use assisting tools such as:

- wandb: https://www.wandb.com/
- comet: https://www.comet.ml/site/
- mlflow: https://mlflow.org/

Visualization Tools



https://www.wandb.com/experiment-tracking
accessed on 25.06.2020

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Debugging machine learning is difficult
  - New risks, new types of errors, new dangers
  - No simple “Go-To-Cookbook”
- Established software debugging tools can be useful if adapted correctly but probably do not suffice
- Problem and model specific debugging methods should be used, much literature exists
- Visualization tools available which can help with debugging in an easy and helpful way
  - But might not be enough!

Literature

- [1] Hall, Patrick and Burt, Andrew. **Why you should care about debugging machine learning models**. O'Reilly. 12/12/2019.
  https://www.oreilly.com/radar/why-you-should-care-about-debugging-machine-learning-models/
- [2] Yousefzadeh, R., & O'Leary, D. P. (2020). **Auditing and Debugging Deep Learning Models via Decision Boundaries: Individual-level and Group-level Analysis**. *arXiv preprint arXiv:2001.00682*.
  https://arxiv.org/abs/2001.00682
- [3] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, Ananth Grama. **MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection**. *ESEC/FSE 2018*. November 2018
  https://dl.acm.org/doi/pdf/10.1145/3236024.3236082
- [4] Daniel Kang, Deepti Raghavan, Peter Bailis, Matei Zaharia. **Model Assertions for Debugging Machine Learning**. *MLSys 2020*
  https://arxiv.org/pdf/2003.01668.pdf

Literature

- Yousefzadeh, R., & O'Leary, D. P. (2019). **Interpreting neural networks using flip points**. *arXiv preprint arXiv:1903.08789*.
  https://arxiv.org/abs/1903.08789
- Chris Nicholson. **A Beginner's Guide to Generative Adversarial Networks (GANs).** Pathmind. 2019.
  https://pathmind.com/wiki/generative-adversarial-network-gan
- W&B. **Six Ways to Debug a Machine Learning Model**. MC.AI. 21/10/2019.
  https://medium.com/six-ways-to-debug-a-machine-learning-model/six-ways-to-debug-a-machine-learning-model-57c0829e85f4
- Chandrasekhar, Govind. **Debugging Neural Networks: A Checklist**. Semantics3 Blog. 8/10/2016.
  https://www.semantics3.com/blog/debugging-neural-networks-a-checklist-ca52e11151ec/

Acknowledgements & License

- Images are either by the authors of these slides or attributed where they are used
- These slides are made available by the authors (Florian Busch, Constantin Stipnieks) under CC BY 4.0