Software Engineering for Artificial Intelligence

Debugging

TECHNISCHE UNIVERSITÄT DARMSTADT

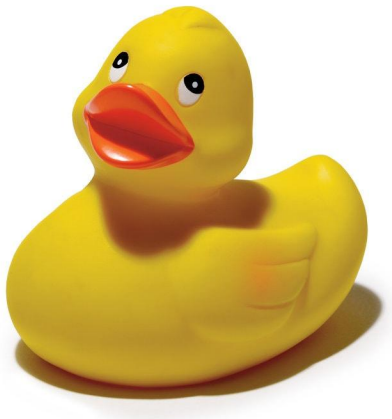


Fig. 1. Rubber duck. From *The New York Time Magazine*. (11.11.2011) Retrieved from https://www.nytimes.com/2011/11/13/magazine/the-rubber-duck-knows-no-frontiers.html

Imagine...

Fig. 2. This world is a BUG. From *Imgflip*. (2016) Retrieved from https://imgflip.com/i/1xvke8

Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- What are bugs in the AI context?
- What are their causes?
- How can we detect and fix bugs?
- What have we learned?

Causes

- The algorithm doesn't work
  - System outages [1]
  - Model outages
- The algorithm doesn't work well enough
  - Intelligence errors: over- / underfitting
  - Unbalanced or insufficient training sample
  - Bad models

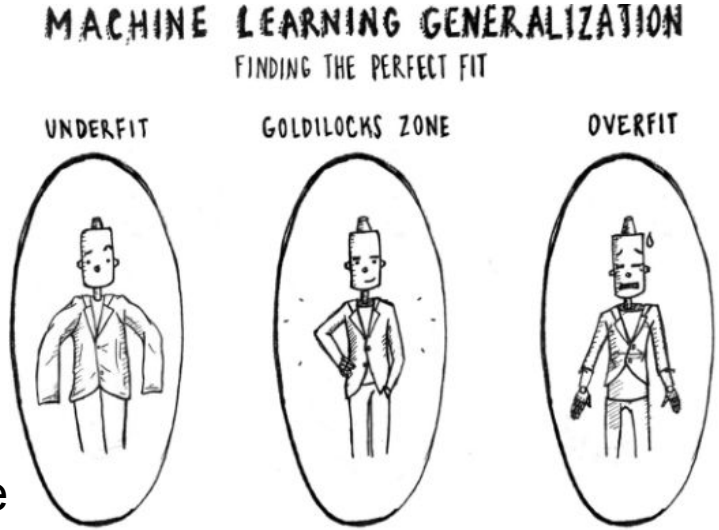


Fig. 3. Over- & Underfitting. From *Analytics Vidhya*. (07.02.2020) Retrieved from https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/

Why special?

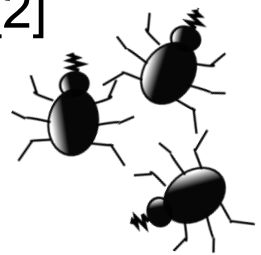
TECHNISCHE UNIVERSITÄT DARMSTADT

- Exponentially difficult debugging (2D to 4D)
  - Algorithm
  - Implementation
  - **Model**
  - **Data**

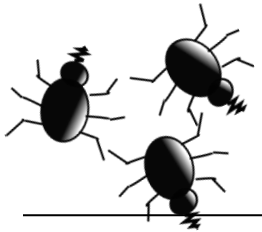
An Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- Training logistic regression using stochastic gradient descent [2]



  - Algorithm: gradient descent updates equations correctly
  - Implementation: feature and parameter computes correctly
  - Model: model capability, limitations (wrong classifier)
  - Data: Noisy labels, mistakes in preprocessing, insufficient/biased data



$n^2 \rightarrow n^4$ ways

Signals:

- ❑ Plot loss functions
- ❑ Actual output
- ❑ Summary of intermediate computations

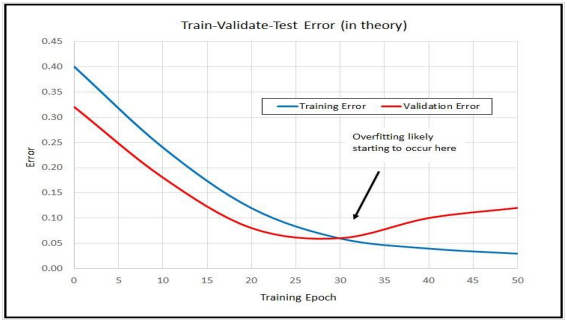


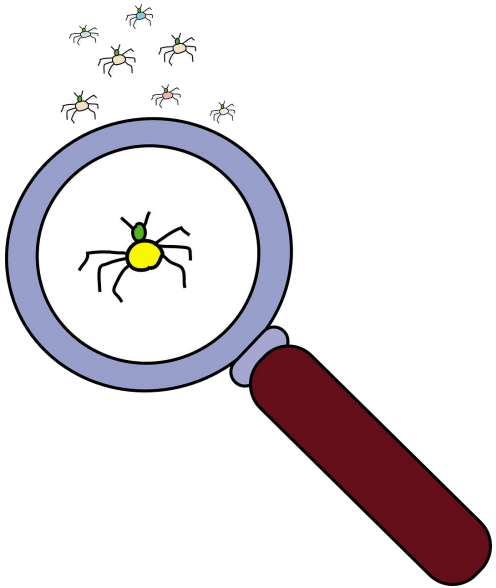
Fig. 4. Neural Network Train-Validate-Test. From *James D. McCaffrey*. (28.06.2018) Retrieved from https://jamesmccaffrey.wordpress.com/2018/06/28/neural-network-train-validate-test/

Why special?

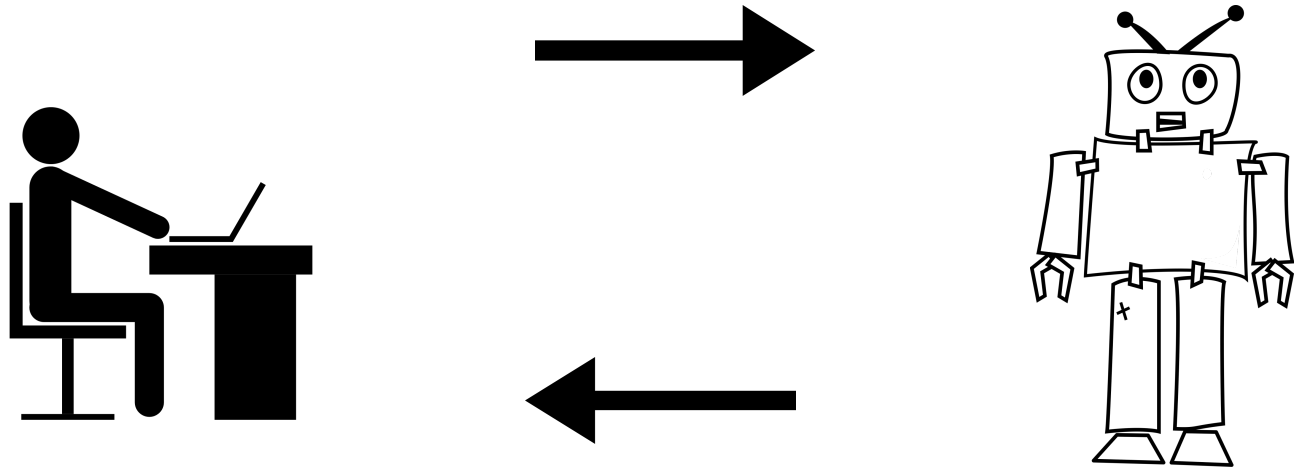
TECHNISCHE UNIVERSITÄT DARMSTADT

- Long debugging cycles

How to detect bugs?



Explainable Debugging



Tools

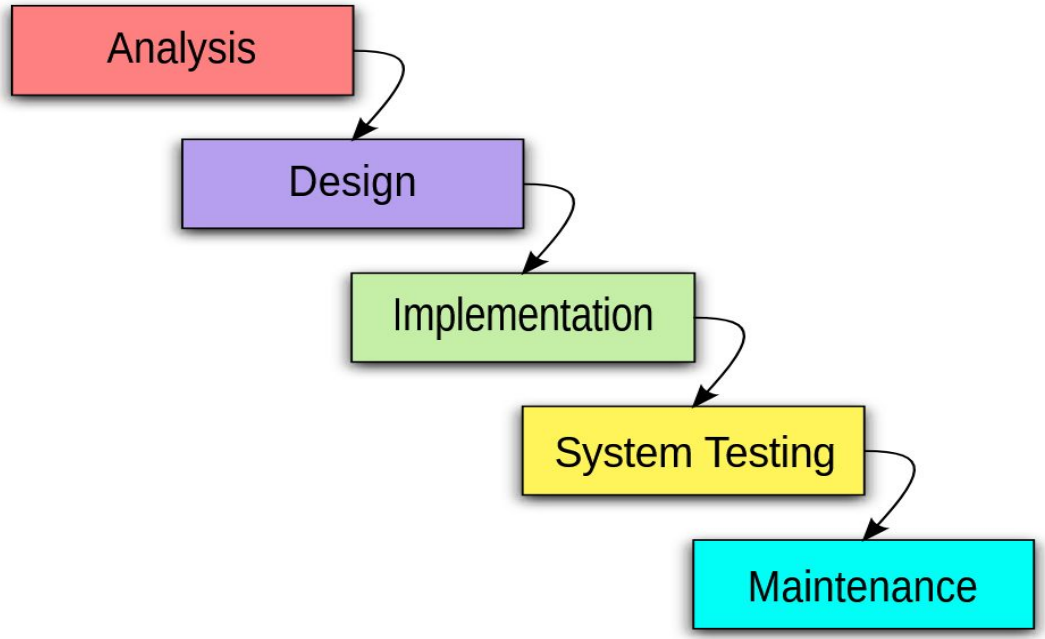


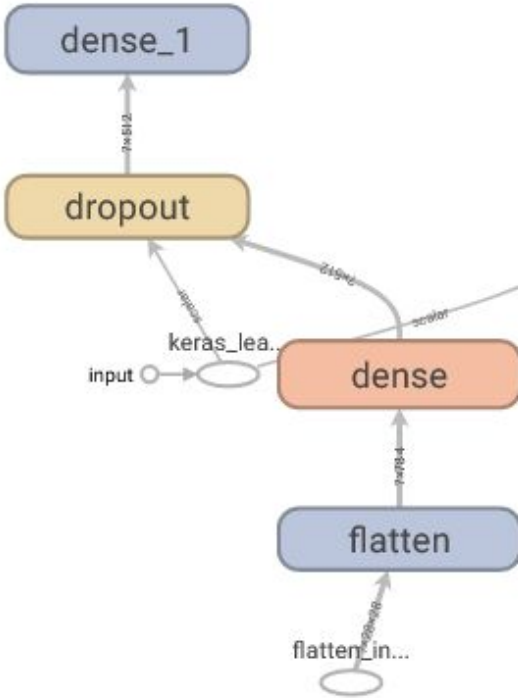
Fig. 5. Waterfall model. From *Wikipedia* Retrieved from https://en.wikipedia.org/wiki/Waterfall_model CC BY 3.0

- Focus on Debugging in Implementation and System Testing phase
- During Implementation bugs in code are targeted
- When System Testing bugs due to training are targeted

Bugs in Implementation

- Incorrect layer connections
- Faulty gradient propagation
- Bad choice of hyperparameters
  - Bad initializations
  - Activation functions
  - Loss function

Fixes for Implementation bugs



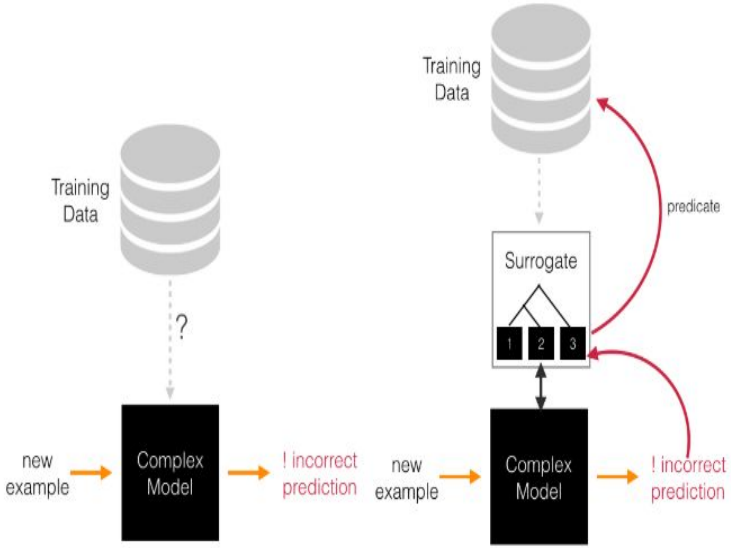
- ❏ Simple sanity checks
- ❏ Observe outputs and parameters of layers
  - ❏ Tensorboard / Neptune
- ❏ Hyper parameter optimization
  - ❏ Optuna / Hyperopt

Bugs in System Testing phase

Fig. 6 Adversarial samples(misclassified) in the bottom row are created from the legitimate samples in the top row. Taken from "Practical Black-Box Attacks against Machine Learning"

- ❑ Bad predictions on subset of data
- ❑ Prejudiced predictions
- ❑ Presence of adversarial inputs [3]

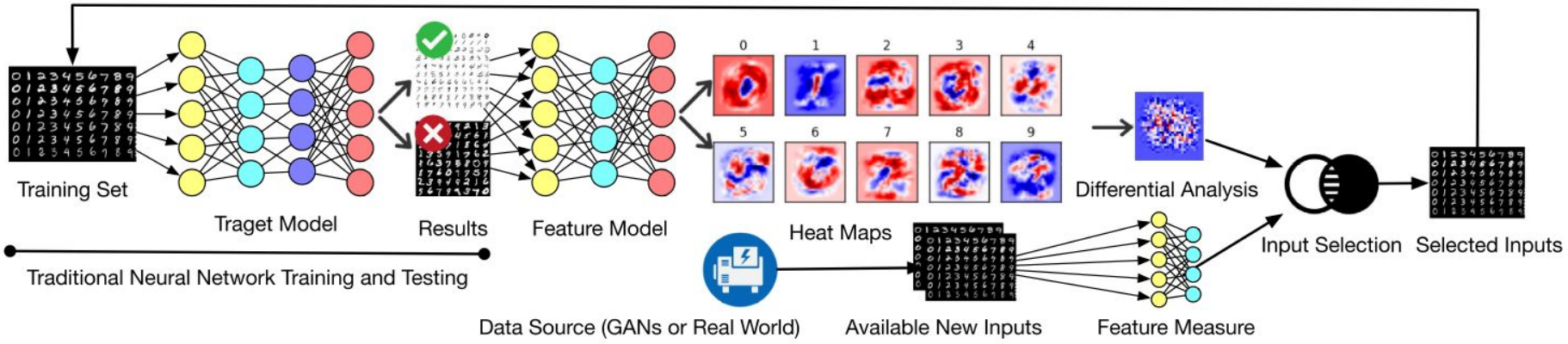
Tools in System Testing phase

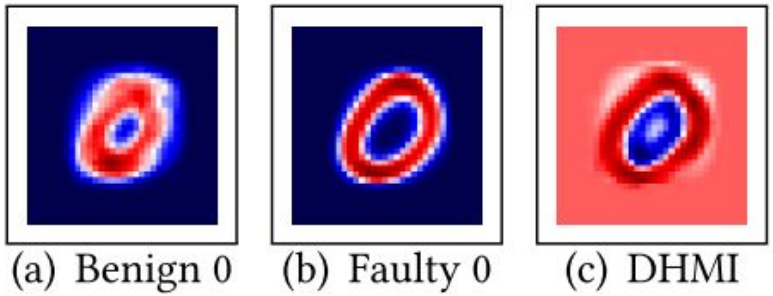


- PALM
  - Surrogate model is decision tree over submodels
- Use of fairness metrics for training [4]
- Retraining on adversarial examples

Fig. 7 Explainable surrogate model to measure influence of training data, taken from Sanjay Krishnan, PALM: Machine Learning Explanations For Iterative Debugging

MODE





- Core idea: Faulty features
- Search for best submodel
- Generating differential Heatmap for the classes
- Retraining with a mix of data that fulfills the criteria of the heatmap and random training data

Fig.8: Workflow of MODE,Taken from MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bugs in AI differ from typical SE bugs
  - Exponentially harder to find the cause
  - Long debugging cycle
- Tools in different phases
  - Implementation Bugs
  - System Testing Bugs
- Bugs are inevitable

知己知彼，百戰不殆。

——《*孫子兵法*》

If you know both yourself and your enemy, you can win numerous battle without jeopardy.

- The Art of War

Wenn man sowohl sich selbst als auch seinen Feind kennt, kann man zahlreiche Schlachten ohne Gefahr gewinnen.

- Sun Tzu, Die Kunst des Krieges

Recommendation

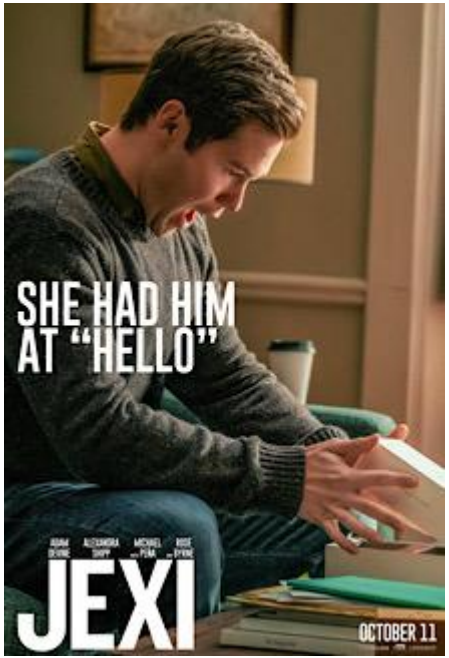


Fig. 9. Jexi. From *Wikipedia*. (2019) Retrieved from https://en.wikipedia.org/wiki/Jexi

When a self-aware smartphone with a female-voiced virtual assistant that becomes emotionally attached to its socially awkward owner....

Trailer

Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

Reference

[1] Geoff Hulten, “Building Intelligent Systems: A Guide to Machine Learning Engineering”. In: 2018.

[2] S. Zaya Enam, “Why is machine learning hard?”, In: 10/11/2016

[3] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, “Practical Black-Box Attacks against Machine Learning”

[4] Mengnan Du, Fan Yang, Fairness in Deep Learning:A Computational Perspective

Acknowledgements & License

- Material Design Icons, by Google under Apache-2.0
- Other images are either by the authors of these slides, attributed where they are used, or licensed under Pixabay, Unsplash or Pexels