Sebastian **Buschjäger**

Wissenschaftlicher Mitarbeiter

Vereinsstraße 22, 44793 Bochum

0151 431 28988 | sebastian.buschjaeger@tu-dortmund.de | www.buschjaeger.it | sbuschjaeger |

Sebastian Buschjäger

Ausbildung

Overberg Grundschule *Fröndenberg*

Grundschule *1996 - 2000*

Gesamtschule Fröndenberg *Fröndenberg*

Abitur *2000 - 2010*

TU Dortmund *Dortmund*

Bachelor Informatik *2010 - 2013*

- Kerninformatik mit Nebenfach Elektrotechnik
- Bachelorarbeit mit dem Titel “Unsupervised Learning of Applied Robot Actuator Coordination”

TU Dortmund *Dortmund*

Master Informatik *2013 - 2016*

- Kerninformatik mit Nebenfach Elektrotechnik
- Masterarbeit mit dem Titel “Online Gauß-Prozesse zur Regression auf FPGAs”

Berufserfahrung

Kommunix GmbH *Unna*

Betriebspraktikum Softwareentwicklung *2006*

Lehrstuhl für Kommunikationsnetze, TU Dortmund *Dortmund*

Studentische Hilfskraft (SHK) *2010 - 2013*

- Implementierung eines Plotting-Tools in Matlab
- Implementierung einer Bibliothek zur Drohnenpositionierung in C/C++
- Umsetzung eines Verfahrens zur maximalen Abdeckung eines Suchgebietes mit Hilfe eines Drohnenschwarms in Matlab und C/C++

Lehrstuhl für Künstliche Intelligenz, TU Dortmund *Dortmund*

Wissenschaftliche Hilfskraft (WHF) *2013 - 2016*

- Literaturrecherche und Report zu Kommunikationsprotokollen für Ad-Hoc Netzwerken im Rahmen von Embedded Systems
- Implementierung eines Schedulingalgorithmus im `streams`-Framework
- Implementierung eines Webcrawlers für Newseinträge von `welt.de`

Lehrstuhl für Künstliche Intelligenz, TU Dortmund *Dortmund*

Wissenschaftlicher Mitarbeiter *2016 - dato*

- Wissenschaftlicher Mitarbeiter im SFB876, Teilprojekt A1

Fähigkeiten

Programmieren	C/C++, Python, Java, LaTex, Matlab
Frameworks und Entwicklung	Numpy, SciPy, Pandas, Docker, Git, GitHub und GitLab CI
Data Science	RapidMiner, Scikit-learn, PyTorch und Keras, Matplotlib, Plotly und Dash
Sprachen	Deutsch, Englisch

Sonstiges

2007 - 2010 **Erwerb universitärer Leistungen als Schüler**, Projekt SchülerUni der TU Dortmund

2008/09 **Jahrgangsbeste Facharbeit**, Gesamtschule Fröndenberg

2010 **Jahrgangsbestes Abitur Abschlussjahrgang 2010**, Gesamtschule Fröndenberg

2011 - 2012 **Stipendiat im Rahmen des Dortmunder-Modells**, TU Dortmund

2012 - 2013 **Stipendiat der Deutschen Telekom im Rahmen des Deutschlandstipendiums**, TU Dortmund

2016 **Masterabschluss mit Auszeichnung**, TU Dortmund

Ausgewählte Veröffentlichungen

Shrub Ensembles for Online Classification

S. Buschjäger, S. Hess, K. Morik

Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22), 2022

Margin-Maximization in Binarized Neural Networks for Optimizing Bit Error Tolerance

S. Buschjäger, J. Chen, K. Chen, M. Günzel, C. Hakert, K. Morik, R. Novkin, L. Pfahler, M. Yayla

Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021, 2021

Very Fast Streaming Submodular Function Maximization

S. Buschjäger, P. Honysz, L. Pfahler, K. Morik

Machine Learning and Knowledge Discovery in Databases. Research Track, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part III, 2021

Randomized Outlier Detection with Trees

S. Buschjäger, P.-J. Honysz, K. Morik

International Journal of Data Science and Analytics (2020). Springer International Publishing, 2020

On-Site Gamma-Hadron Separation with Deep Learning on FPGAs

S. Buschjäger, L. Pfahler, J. Buss, K. Morik, W. Rhode

Machine Learning and Knowledge Discovery in Databases: ADS Track, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part IV, 2020

Decision Tree and Random Forest Implementations for Fast Filtering of Sensor Data

S. Buschjäger, K. Morik

IEEE Trans. Circuits Syst. I Regul. Pap. 65-I.1 (2018) S. 209–222. 2018

Realization of Random Forest for Real-Time Evaluation through Tree Framing

S. Buschjäger, K.-H. Chen, J.-J. Chen, K. Morik

The IEEE International Conference on Data Mining series (ICDM), 2018

Ausgewählte Softwareprojekte

PyPruning (`https://github.com/sbuschjaeger/PyPruning`): PyPruning ist eine Softwarebibliothek zum pruning von Ensembles, d.h. dem Entfernen von einzelnen Modelle aus einem Ensemble. Pruning verbessert so die Vorhersagegüte von bereits trainierten Ensembles (z.B.: Random Forest) und reduziert gleichzeitig den Ressourcenverbrauch. PyPruning implementiert aktuell 16 verschiedenen Pruningverfahren aus 12 verschiedenen Papiere. Des Weiteren ist PyPruning modular aufgebaut, sodass sich existierenden Verfahren leicht erweitern lassen.

FastInference (`https://github.com/sbuschjaeger/fastinference`): FastInference ist ein Codegenerator und Modellkompiler für Machine Learning Modelle welcher modell- und zielarchitekturspezifischen Ausführungscode generiert. FastInference unterstützt sowohl moderne Deep Learning Modelle (z.B.: Convolutional Neural Networks), als auch klassische Verfahren (z.B.: Random Forests). FastInference kombiniert Modelloptimierung und Codegenerierung mithilfe einer Template-Engine. Ein gegebenes Modell wird zunächst optimiert (z.B. Quantisierung der Gewichte) und dann werden aus einer Template-Library die entsprechenden Codeteile geladen welche architekturspezifisch angepasst werden (z.B. durch Optimierungen des Speicherlayouts). Der resultierende Ausführungscode ist so für das Modell als auch auf die Zielarchitektur optimal abgestimmt. Aktuell unterstützt FastInference lineare Regression, Entscheidungsbäume, Multilayer Perceptrons, Convolutional Neural Networks, Binarized Neural Networks und Ensembles aus diesen Modellen. Die Zielsprache ist `C/C++` für Intel / ARM, wobei auch FPGAs (via High-Level Synthese) und weitere Metasprachen wie `haxe` und `iree` teilweise unterstützt werden.

Submodular Streaming Maximization (`https://github.com/sbuschjaeger/SubmodularStreamingMaximization`): Submodular Maximization implementiert in einer header-only `C++` Bibliothek mit Python-bindings. Dieses Projekt implementiert insgesamt 7 verfahren zur Maximierung von submodularen Funktionen. Die rechenaufwendigen Teile der Implementierung sind in `C++` ausgelagert, wobei es sowohl eine `C++` API, als auch eine Python API gibt. Neben den existierenden submodularen Funktionen lässt sich die Bibliothek leicht erweitern, da alle Python-Objekte und alle `C++` Objekte untereinander kompatible sind.