UNIVERSITÄT OSNABRÜCK

Eingebettete Softwaresysteme

Black-Box Models for Non-Functional Properties of AI Software Systems

Birte Friesel, Olaf Spinczyk

birte.friesel@uos.de

May 17, 2022

Osnabrück University / Embedded Software Systems

Variability of AI Tasks in Agriculture

Harvester control

Variability of AI Tasks in Agriculture

Harvester control

Obstacle detection

Variability of AI Tasks in Agriculture

Harvester control

Obstacle detection

Classification

Variability of AI Tasks in Agriculture

Harvester control

Obstacle detection

Classification

- Latency? Accuracy? Hardware cost? Model size? Memory usage?

Variability of AI Tasks in Agriculture

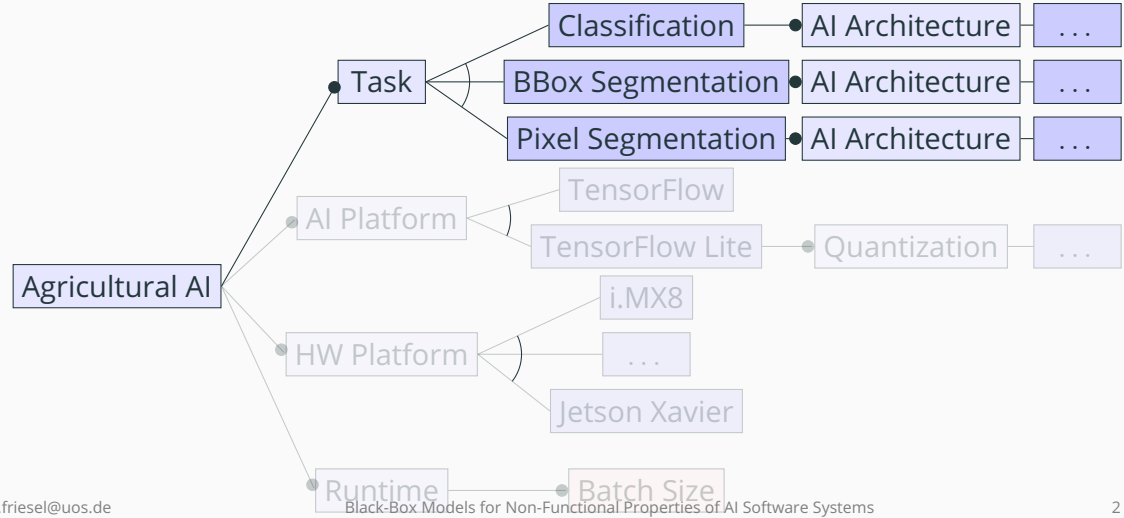
Harvester control

Obstacle detection

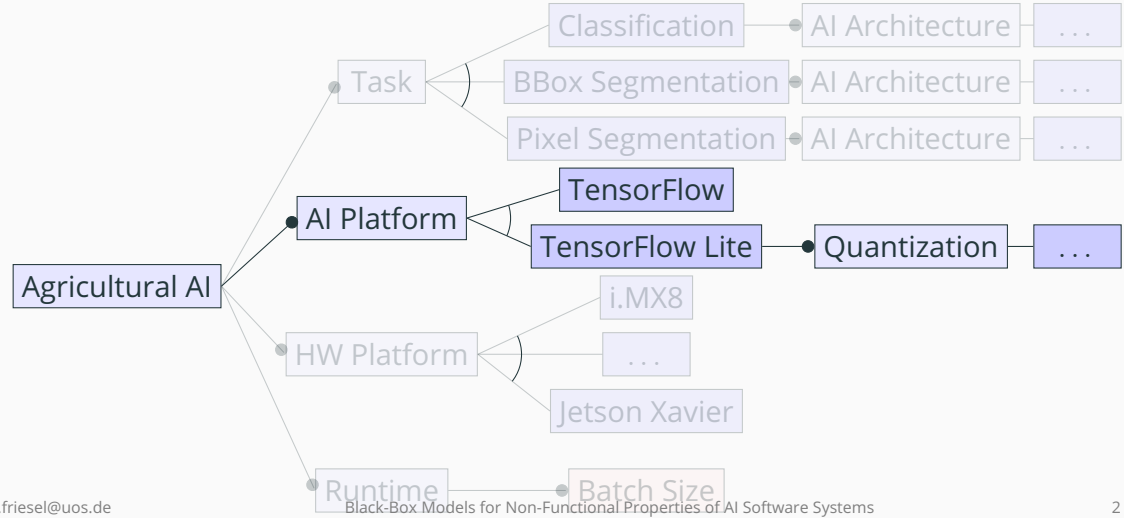
Classification

- Latency? Accuracy? Hardware cost? Model size? Memory usage?
- Software ↔ Hardware interactions
- “Use GPU-accelerated TensorFlow or CPU-bound TensorFlow Lite?”

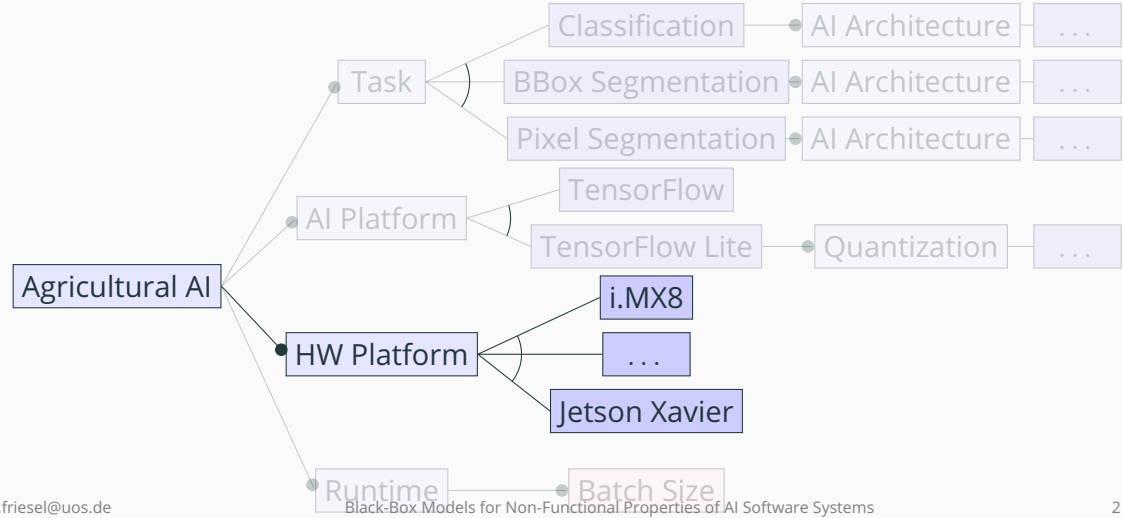
Variability of AI Software Systems



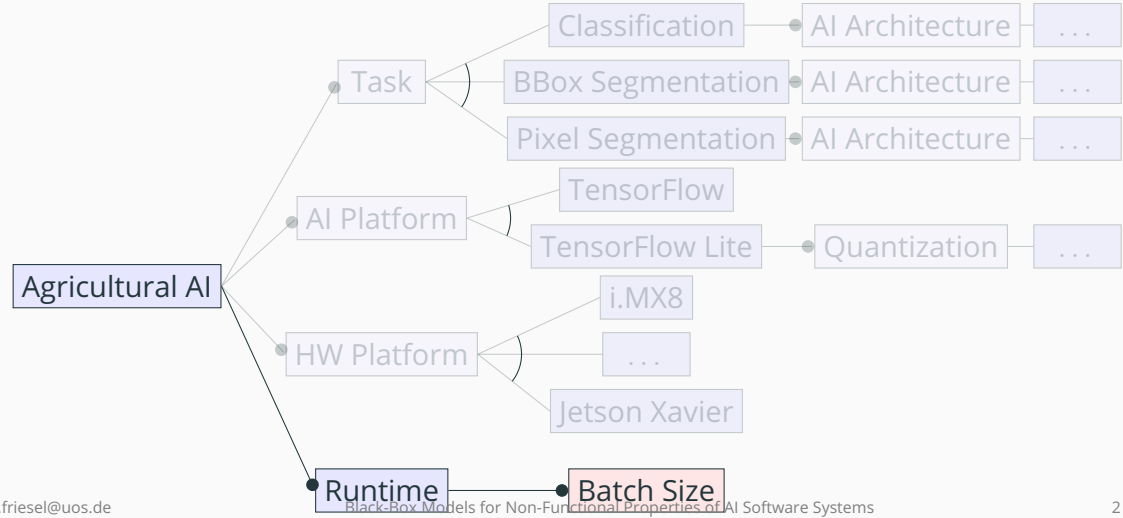
Variability of AI Software Systems



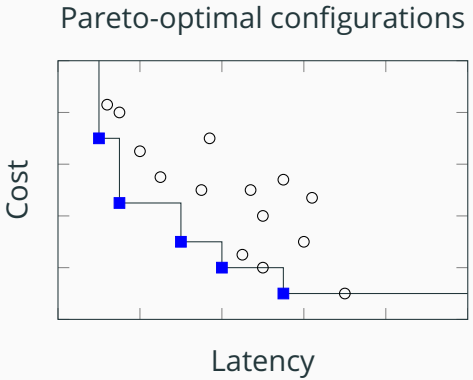
Variability of AI Software Systems



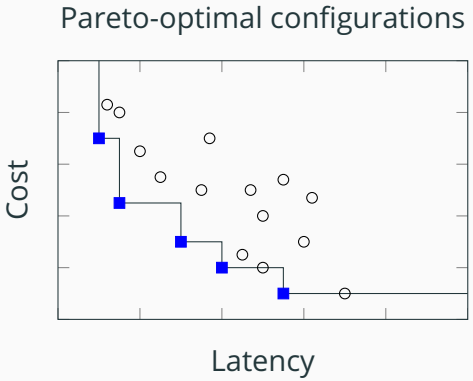
Variability of AI Software Systems



Cost and Performance Optimization



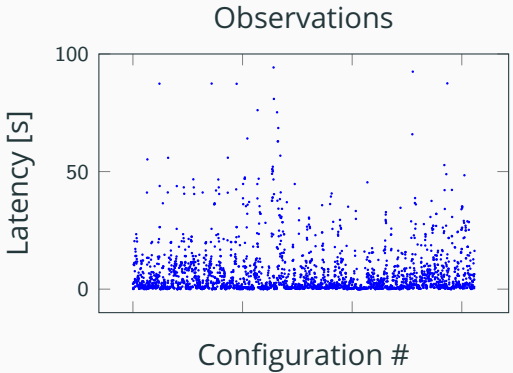
Cost and Performance Optimization



- Exhaustive search is impractical

→ Use black-box models for configuration [SSS10; Per+21]

Cost and Performance Optimization



- Exhaustive search is impractical

→ Use black-box models for configuration [SSS10; Per+21]

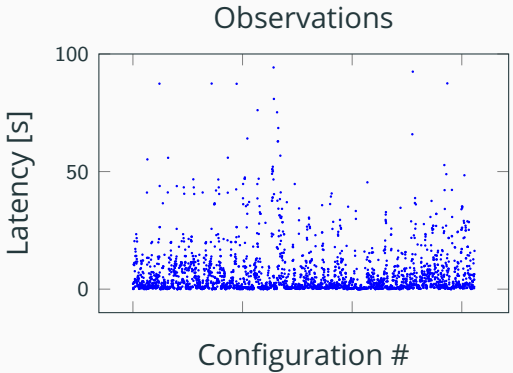
- Goal: automatic generation

Configurations $\vec{x}_1, \vec{x}_2, \ldots$
Observations $y_1, y_2, \ldots$

→ Machine Learning Algorithm →

NFP Model
$f : M \to \mathbb{R}$

Cost and Performance Optimization



- Exhaustive search is impractical

→ Use black-box models for configuration [SSS10; Per+21]

- Goal: automatic generation

→ Accurate? Interpretable? Noteworthy observations?

Configurations $\vec{x}_1, \vec{x}_2, \ldots$
Observations $y_1, y_2, \ldots$

→ Machine Learning Algorithm →

NFP Model
$f : M \to \mathbb{R}$

Related Work in AI Engineering

Focus on specific models or use cases

- Neural Architecture Search (NAS):
  Optimize accuracy of provided neural network [Tan+19; LDL21]
- White-box latency models for specific GPUs and TinyML [Li+21; Ban+21]

Related Work in AI Engineering

Focus on specific models or use cases

- Neural Architecture Search (NAS):
  Optimize accuracy of provided neural network [Tan+19; LDL21]
- White-box latency models for specific GPUs and TinyML [Li+21; Ban+21]

Here:

- Selection of promising neural networks for optimization
- Hardware selection and runtime configuration

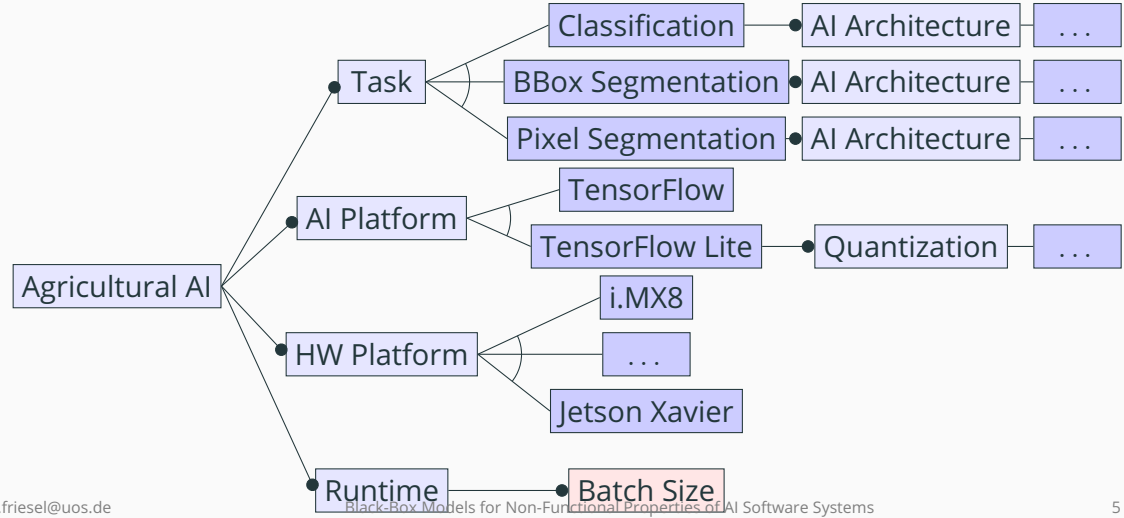
Contents

Methodology

Accuracy and Interpretability

Observations

AI Product Line / Feature Model



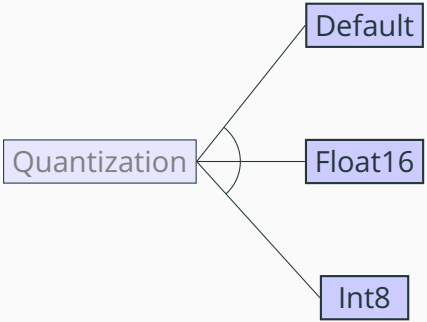
Feature Extraction

Product line configuraton ⇒ Feature vector $\vec{x}$

Scalar:
Batch Size $\in \mathbb{N}_{>0}$

Feature Extraction

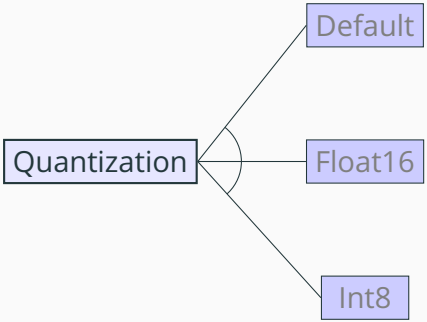
Product line configuraton ⇒ Feature vector $\vec{x}$



- Boolean:
  Default, Float16, Int8 $\in \{0, 1\}$
  $\vec{x} \in \{\{1, 0, 0\}, \{0, 1, 0\}, \{0, 0, 1\}, \{0, 0, 0\}\}$

Feature Extraction

Product line configuraton $\Rightarrow$ Feature vector $\vec{x}$



- Boolean:
  Default, Float16, Int8 $\in \{0, 1\}$
  $\vec{x} \in \{\{1, 0, 0\}, \{0, 1, 0\}, \{0, 0, 1\}, \{0, 0, 0\}\}$
- Categorial:
  Quant $\in \{\text{Default}, \text{Float16}, \text{Int8}, \perp\}$

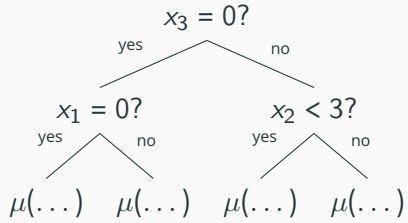
Modeling Methods

- Focus on decision trees:
  - Automatic generation of tree structure
  - Easy to understand

Modeling Methods

- Focus on decision trees:
  - Automatic generation of tree structure
  - Easy to understand

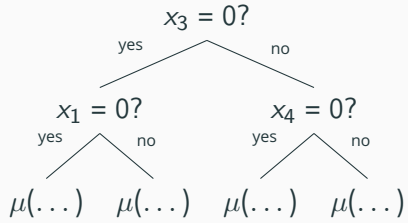
- CART (classification and regression trees) [Bre+84]



Modeling Methods

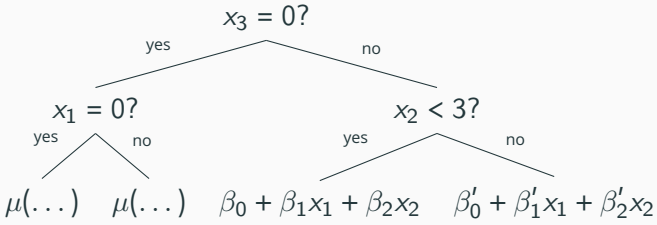
- Focus on decision trees:
  - Automatic generation of tree structure
  - Easy to understand

- CART (classification and regression trees) [Bre+84]
  - DECART (data-efficient CART: no scalars) [Guo+18]



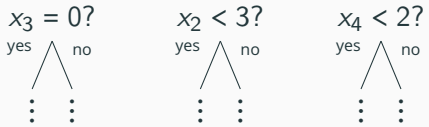
Modeling Methods

- Focus on decision trees:
  - Automatic generation of tree structure
  - Easy to understand
- CART (classification and regression trees) [Bre+84]
  - DECART (data-efficient CART: no scalars) [Guo+18]
- LMT (linear model trees) [Qui+92]



Modeling Methods

- Focus on decision trees:
  - Automatic generation of tree structure
  - Easy to understand
- CART (classification and regression trees) [Bre+84]
  - DECART (data-efficient CART: no scalars) [Guo+18]
- LMT (linear model trees) [Qui+92]
- XGB (extreme gradient boosting) [CG16]



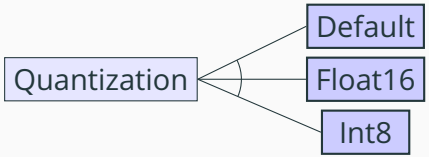
Tree Structure: Binary vs Categorial

Binary

def, f16, i8 $\in \{0, 1\}$

Categorial

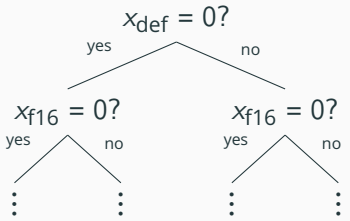
Quant $\in \{\text{Default}, \text{Float16}, \text{Int8}, \perp\}$



Tree Structure: Binary vs Categorial

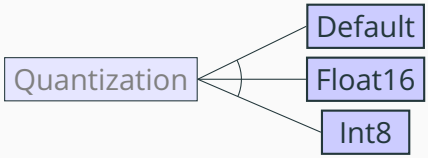
Binary Trees (conventional)

def, f16, i8 ∈ {0, 1}



Categorial

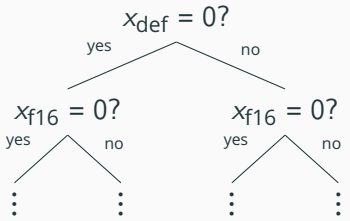
Quant ∈ {Default, Float16, Int8, ⊥}



Tree Structure: Binary vs Categorial

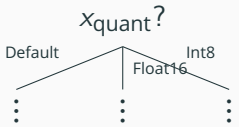
Binary Trees (conventional)

def, f16, i8 ∈ {0, 1}

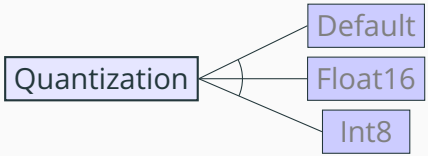


Categorial / non-binary trees [Kep96]

Quant ∈ {Default, Float16, Int8, ⊥}



Close to feature model



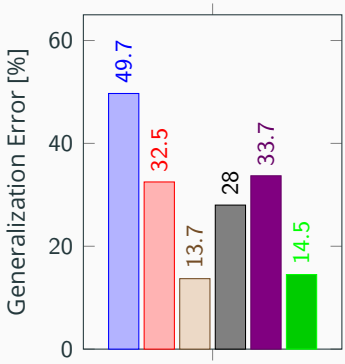
Contents

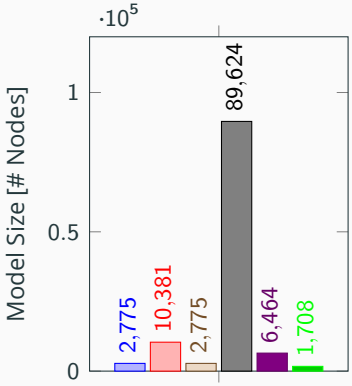
Methodology

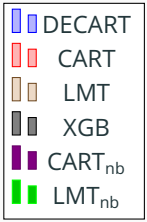
Accuracy and Interpretability

Observations

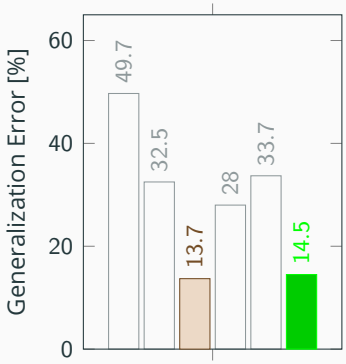
Runtime Memory Usage

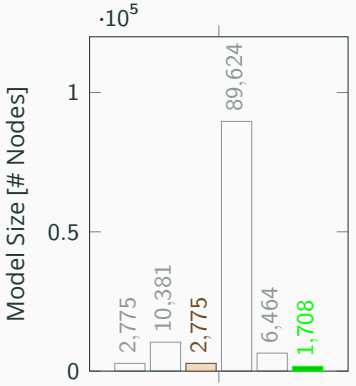


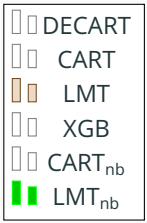




Runtime Memory Usage







⇒ Linear Model Trees perform best; overall acceptable model accuracy

Contents

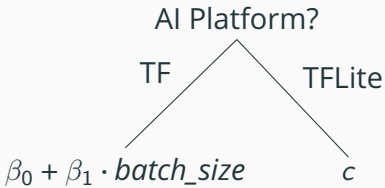
Methodology

Accuracy and Interpretability

Observations

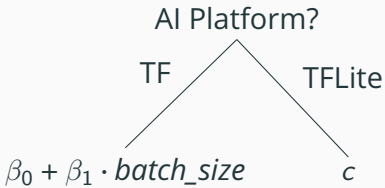
Interpreting the NFP Model

Typical sub-structure in Linear Model Tree for Throughput:



Interpreting the NFP Model

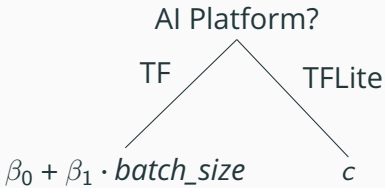
Typical sub-structure in Linear Model Tree for Throughput:



⇒ batching useless on TensorFlow Lite?

Interpreting the NFP Model

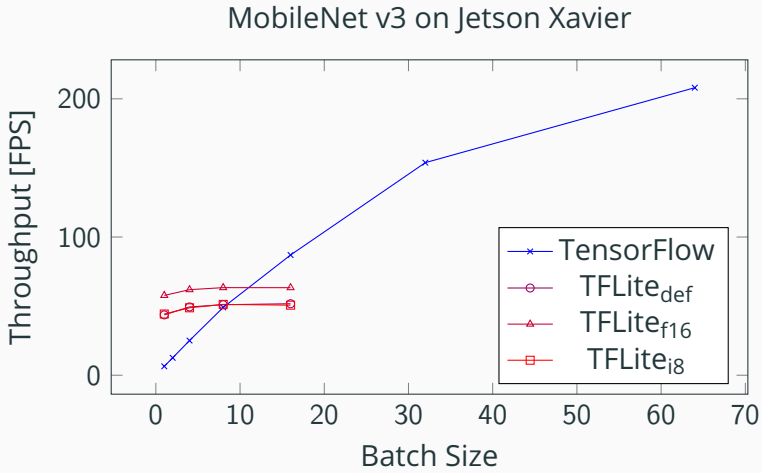
Typical sub-structure in Linear Model Tree for Throughput:



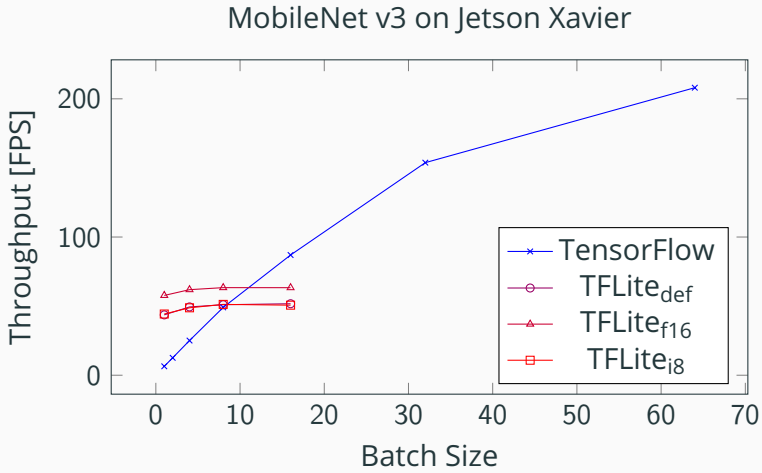
⇒ batching useless on TensorFlow Lite?

$c > \beta_0 + \beta_1$ ⇒ TensorFlow Lite faster for low batch sizes?

Batching: Observations



Batching: Observations



⇒ TFLite better than GPU-accelerated TensorFlow for low-latency inference

Conclusion

- Considering Hardware/Software interactions is important
- Black-Box Models for AI Software Systems provide valuable insights

Conclusion

- Considering Hardware/Software interactions is important
- Black-Box Models for AI Software Systems provide valuable insights

- Linear Model Trees perform best in our evaluation
  - abstraction error < 13 %
  - generalization error < 26 %
- Categorial trees are useful: more compact, ≈ same accuracy

References i

[Ban+21] Colby Banbury et al. **"MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers".** In: Proceedings of Machine Learning and Systems. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3. 2021, pp. 517–532.

[Bre+84] Leo Breiman et al. **Classification and regression trees.** Routledge, 1984. DOI: 10.1201/9781315139470.

[CG16] Tianqi Chen and Carlos Guestrin. **"XGBoost: A Scalable Tree Boosting System".** In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

References ii

[Guo+18] Jianmei Guo et al. **"Data-Efficient Performance Learning for Configurable Systems".** In: Empirical Softw. Engg. 23.3 (June 2018), pp. 1826–1867. ISSN: 1382-3256. DOI: 10.1007/s10664-017-9573-6. URL: https://doi.org/10.1007/s10664-017-9573-6.

[Kep96] Stanislav Keprta. **"Non-binary classification trees".** In: Statistics and Computing 6.3 (1996), pp. 231–243.

[LDL21] Edgar Liberis, Łukasz Dudziak, and Nicholas D. Lane. **"μNAS: Constrained Neural Architecture Search for Microcontrollers".** In: Proceedings of the 1st Workshop on Machine Learning and Systems. EuroMLSys '21. Online, United Kingdom: Association for Computing Machinery, 2021, pp. 70–79. ISBN: 9781450382984. DOI: 10.1145/3437984.3458836. URL: https://doi.org/10.1145/3437984.3458836.

References iii

[Li+21] Jinyang Li et al. **“Towards an Accurate Latency Model for Convolutional Neural Network Layers on GPUs”.** In: MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM). 2021, pp. 904–909. DOI: 10.1109/MILCOM52596.2021.9652907.

[Per+21] Juliana Alves Pereira et al. **“Learning software configuration spaces: A systematic literature review”.** In: Journal of Systems and Software 182 (2021), p. 111044. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2021.111044. URL: https://www.sciencedirect.com/science/article/pii/S0164121221001412.

[Qui+92] John R Quinlan et al. **“Learning with continuous classes”.** In: 5th Australian joint conference on artificial intelligence. Vol. 92. World Scientific. 1992, pp. 343–348.

References iv

[SSS10] Julio Sincero, Wolfgang Schröder-Preikschat, and Olaf Spinczyk. **“Approaching Non-functional Properties of Software Product Lines: Learning from Products”.** In: 2010 Asia Pacific Software Engineering Conference. 2010, pp. 147–155. DOI: 10.1109/APSEC.2010.26.

[Tan+19] Mingxing Tan et al. **“MnasNet: Platform-Aware Neural Architecture Search for Mobile”.** In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2019.