Karlsruhe Institute of Technology

Pervasive Computing Systems · www.teco.kit.edu

Verschnittoptimierung für 2D-Laserschneidmaschinen mit Methoden des maschinellen Lernens

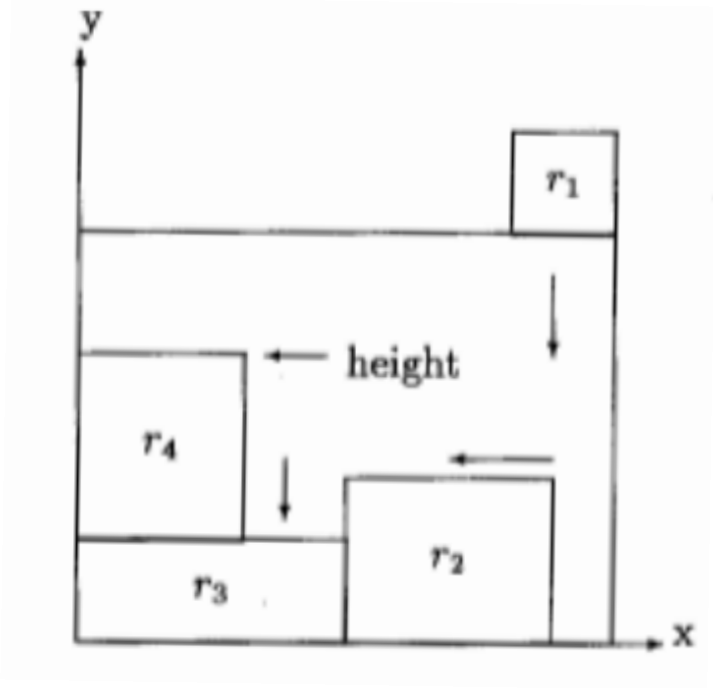
Dr. Johannes Riesterer, Yiran Huang, Yexu Zhou.

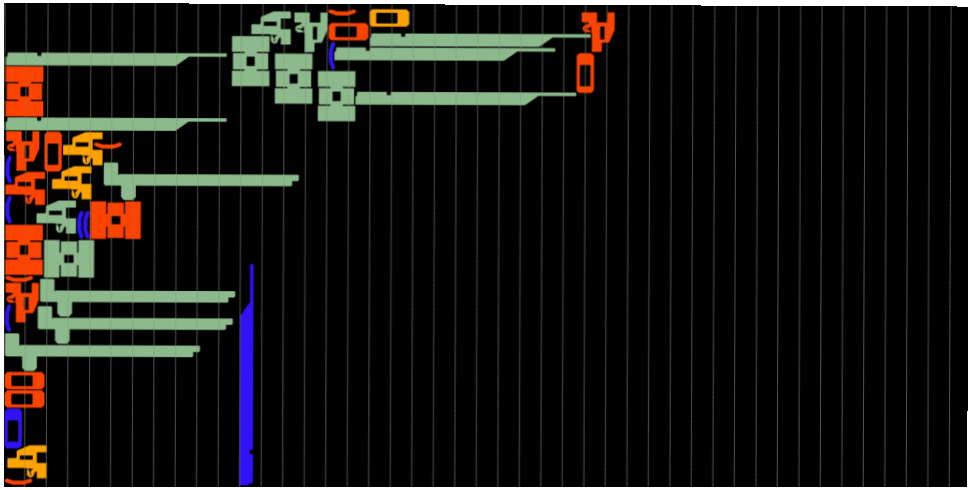
INSTITUTE OF TELEMATICS,
CHAIR FOR PERVASIVE COMPUTING SYSTEMS / TECO

TRUMPF

Modellierung

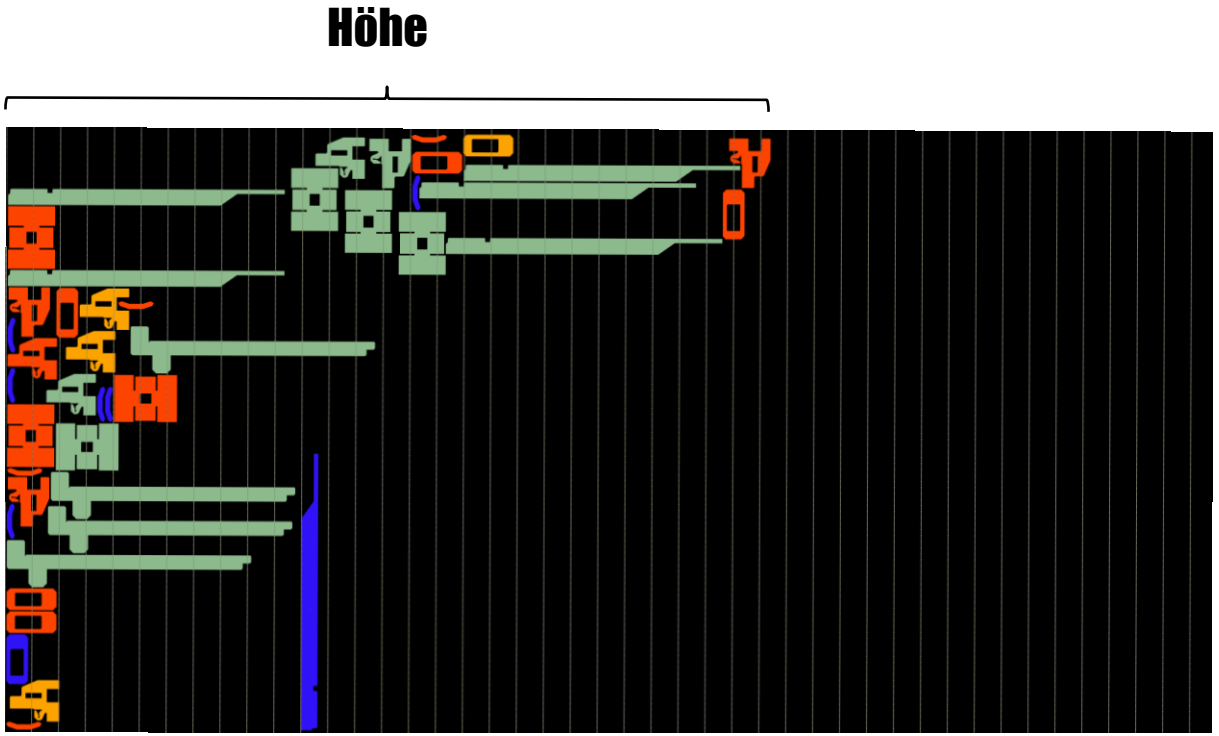
Gegeben: Nester N, der eine geordnete Schablonenkonfiguration $D = (S_1, \ldots, S_n)$ mit dem Bottom-Left-Algorithmus anordnet und mit Kipp-Codierung (2 = stabil, 3 = unsicher, 4 = kippt, 5 = fällt) bewertet.





Modellierung

Gesucht: Für gegebene Konfiguration D eine Permutation $\sigma(D) \coloneqq (S_{\sigma(1)}, \dots, S_{\sigma(n)})$ der Schablonen, so dass eine Bewertungsfunktion $\mathrm{L}(N(\sigma(D)))$ minimal wird.



Modellierung

„Klassisches“ Optimierungsproblem:

Optimierungsverfahren $O(D) \rightarrow \min_{\sigma} L(N(\sigma(D)))$.

Zum Beispiel O:=genetischer Algorithmus - „kleine Variationen der Eingabe“.

Problem: Lange Rechenzeiten bei jeder neuen Konfiguration D.

Modellierung

Modellgestützte Optimierung:

- Trainiere Modell M_k für den Konfigurationsraum $K \coloneqq \{\sigma(D)\}_{(\sigma,D)}$, welches für Eingabe $\sigma(D)$ Bewertung $\mathrm{M_K}\big(\sigma(D)\big) \coloneqq \mathrm{f(L}(N(\sigma(D)))$ ausgibt. $|K| \sim (44 \cdot 2)^{50}$ (Anzahl versch. Schablonen, Rotation, Nesting)

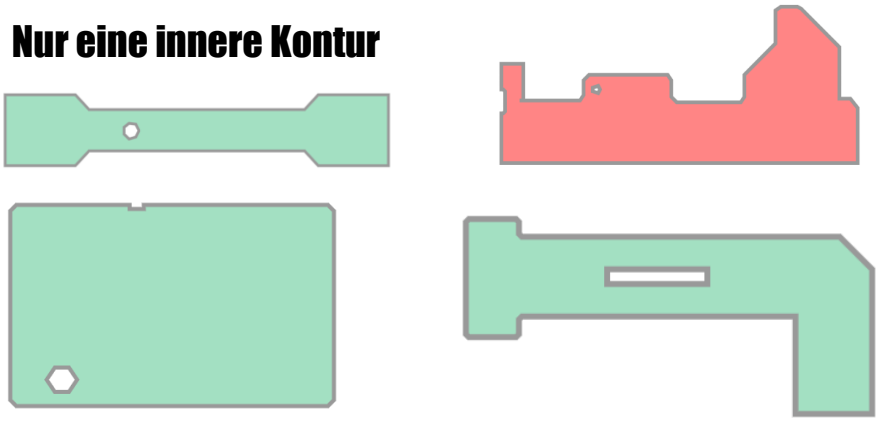
- Ziehe zufällig 100 Permutationen $\{\sigma_i\,(D)\}_{i=1,100}$ aufsteigend sortiert nach Bewertung $\mathrm{M_K}\big(\sigma_i(D)\big)$. Definiere $O_{M_k}(D) \coloneqq \sigma_1(D)$.

- **Hypothese**: Modellgestützte Optimierung O_{M_k} liefert bessere Resultate als zufällig gezogene Permutation.

Bauteile

48 neue Bauteile + 6 vorhandene Bauteile

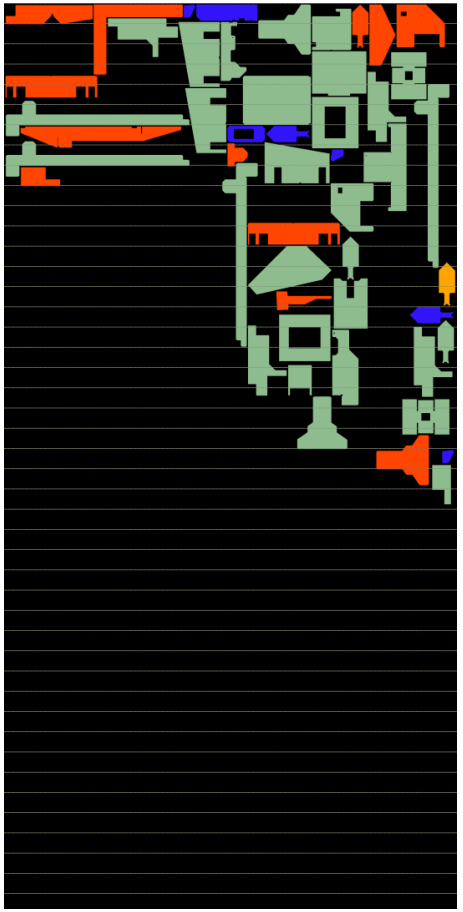
- **Nur eine innere Kontur**



- **Flächenskalierung**

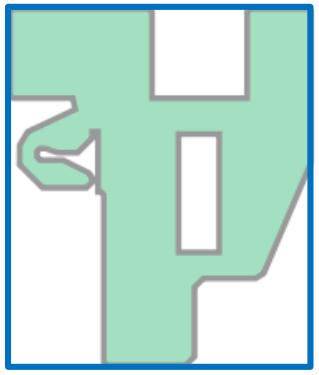
  Innerhalb (Mindestfläche von 6 Bauteile, Maximale Fläche von 6 Bauteile)

  Skalar : Zufällig erzeugt

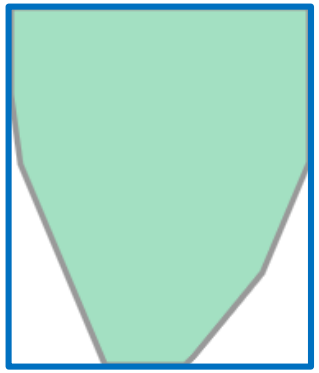


Feature Engineering

- **Flächengröße : 7228.38**
- **Länge des Umfangs : 901.58**
- **Anzahl der Ecken : 53**
- **Lange geteilt durch Breite : 1.19**
- **Fläche geteilt durch Umfang : 8.02**
- **Fläche geteilt durch die Fläche von Bounding box : 0.57**
- **Convex Fläche : 10236.16**



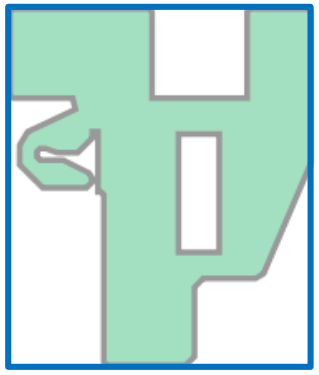
Flächengröße
Länge des Umfangs
Bounding box



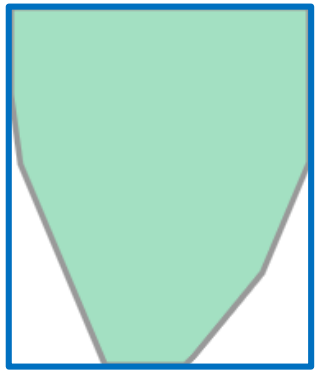
Convex Flächengröße

Feature Engineering

- **Fläche geteilt durch Convex Fläche : 0.71**
- **Convex Fläche geteilt durch die Fläche von Bounding box : 0.8**
- **Convex Fläche geteilt durch die Fläche von Bounding box : 0.8**
- **Fläche durch Convex Fläche: 0.71**
- **Ist Rotation gleich 0: 0/1**
- **Ist Rotation gleich 90: 0/1**



- Flächengröße
- Länge des Umfangs
- Bounding box



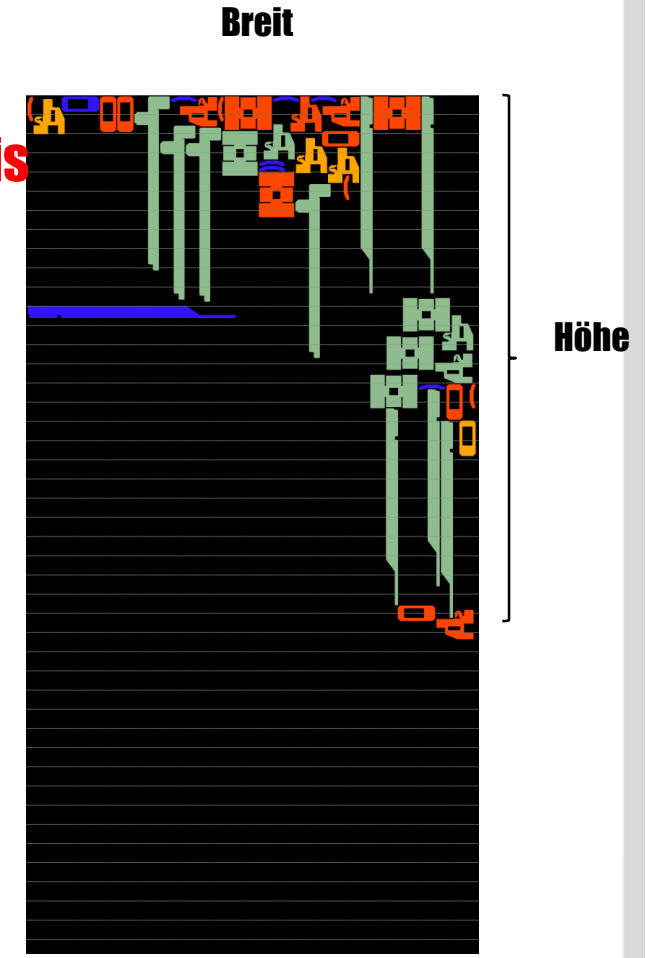
- Convex Flächengröße

Bewertung

SCORE= Mean_score + 0.5 * Höhe + Flächenverhältnis

$$L\big(N(S_1, S_2, \dots, S_n)\big) = (Score(S_1), Score(S_1, S_2), \dots, Score(S_1, \dots S_n))$$

	Stabil	Unsicher	Kippt	Fällt
Metric4	2	3	4	5
Score	0	3.5	4	0.5



- **Höhe**
- **Flächenverhältnis**

$$1 - \frac{Gesamtfläche\ der\ 50\ Bauteile}{Breit\ * Höhe}$$

Modell

- Wir benötigen den Prädiktor, um für Eingabestrings mit variabler Länge zu arbeiten
- Korreliert mit wahrer Leistung: Wir müssen nicht unbedingt einen kleiner mean squared error erreichen, aber wir wollen, dass der Prädiktor die Konfigurationen etwa in der gleichen Reihenfolge einordnet wie ihre wahren Leistungswerte.

SEQ2SEQ:

Input : 1x13 | 1x13 | ● ● ● | 1x13 | 1x13

Output: $Score(S_1)$ | $Score(S_1, S_2)$ | ● ● ● | $Score(S_1, \dots, S_{n-1})$ | $Score(S_1, \dots, S_n)$

Loss: L1

Hyper-parameter:

Von **SMAC** optimierte **seq2seq** Struktur :

LSTM: (13, 64, num_layers=2)

MLP:

(0): Linear(in_features=64, out_features=16, bias=True)

(1): Dropout(p=0.5)

(2): ReLU()

(3): Linear(in_features=16, out_features=1, bias=True)

Evaluation

- Die Platzierung der Bauteile ist stark von den Gewichtungen der **Packungsdichte**, **Verkippen**, **Steg-Beschädigung** beeinflusst.

Gewichtung	1: 1: 1	1: 8: 1	8: 1: 1

- Die Leistung des Modells sollte getestet werden, wenn es neue unbekannte Bauteile gibt.
  - Insgesamt 54 Bauteile, 44 für Training.

- Metric: **Beste Test-Konfiguration ist in top 5 der Vorhersage**
  Datenmenge = 15000 Batches = 130--150 Batch size = 100

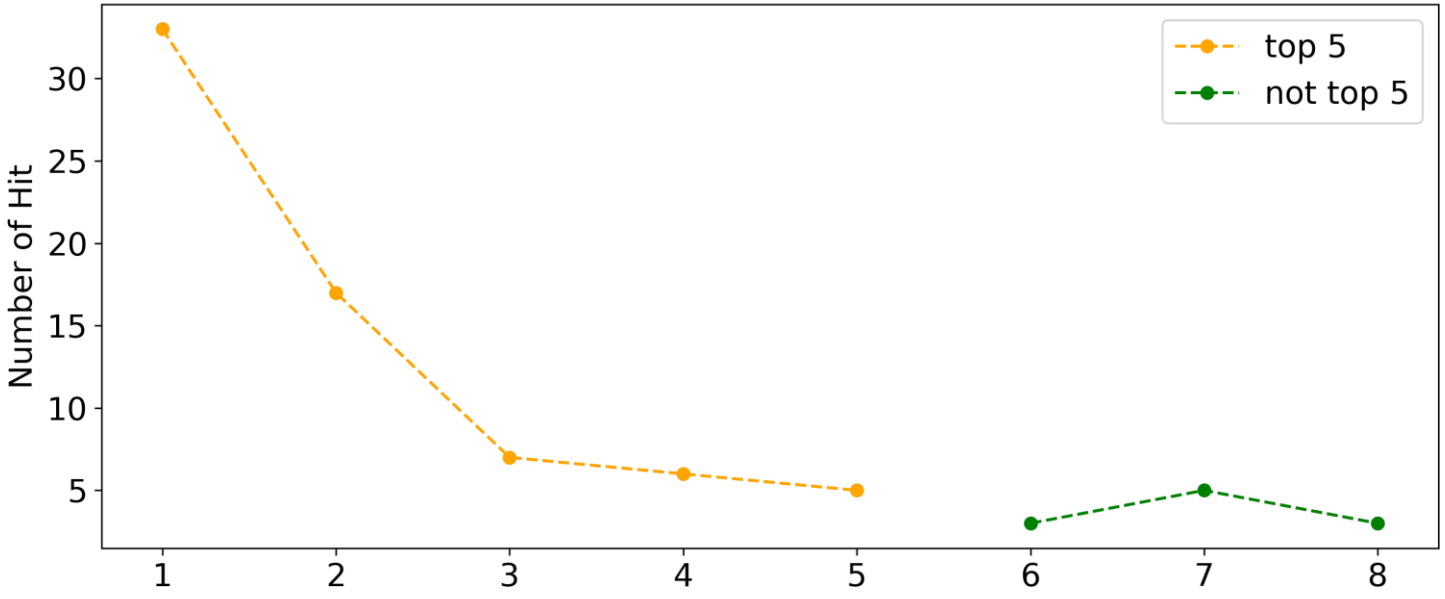
Evaluation

Experiment 1

Setup: Gewichtung ist 1: 1: 1
Anzahle der Bauteile für Training : 54
Anzahle der Bauteile für Test : 54

Metric : Beste Test-Konfiguration ist in top 5 der Vorhersage

Result : 84%



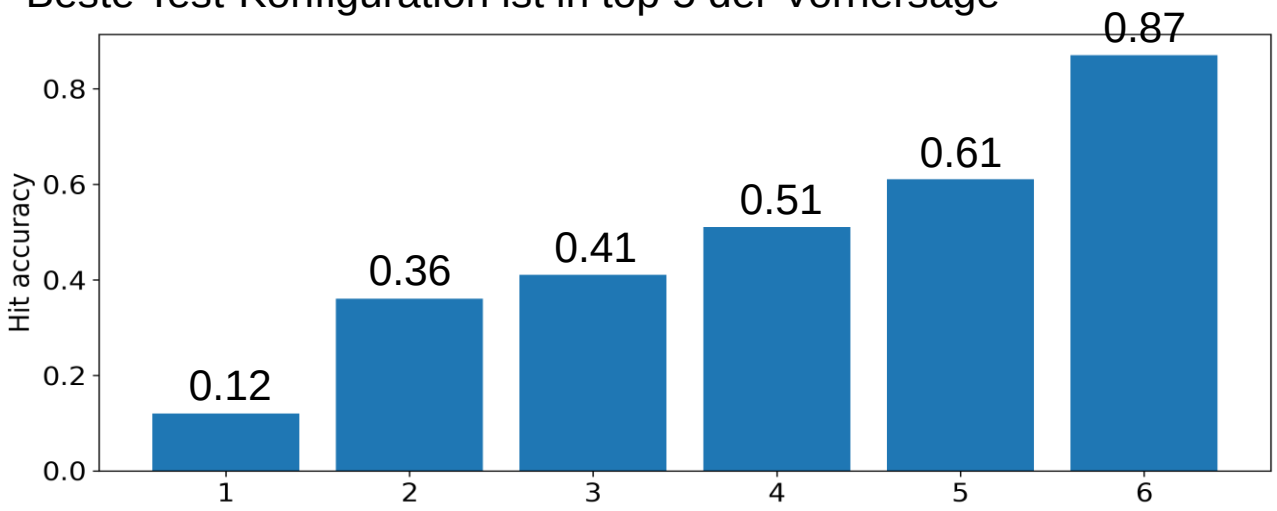
Evaluation

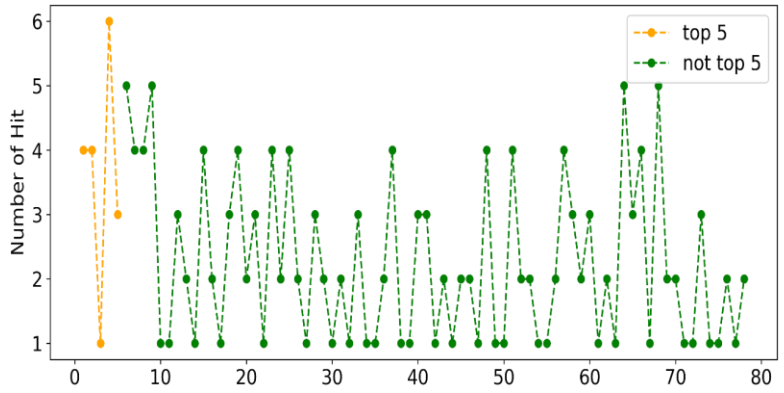
Experiment 2

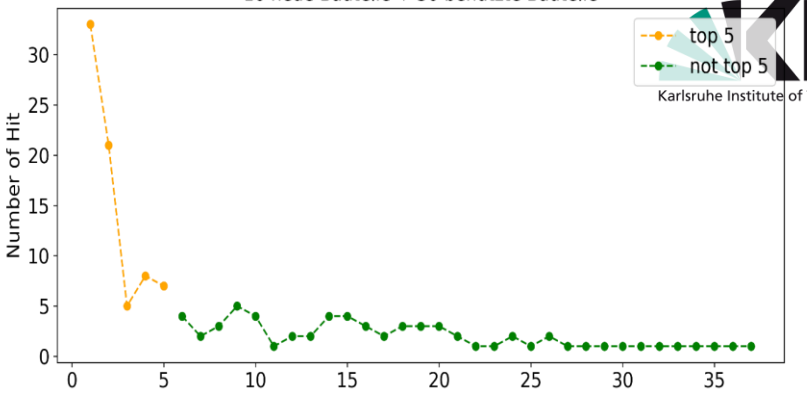
Setup: Gewichtung ist 1: 8: 1
Anzahle der Bauteile für Training : 44
6 verschiedene Datensatz:
(1) 10 neue Bauteile
(2) 10 neue Bauteile + 10 benutzte Bauteile
(3) 10 neue Bauteile + 20 benutzte Bauteile
(4) 10 neue Bauteile + 30 benutzte Bauteile
(5) 10 neue Bauteile + 44 benutzte Bauteile
(6) 44 benutzte Bauteile

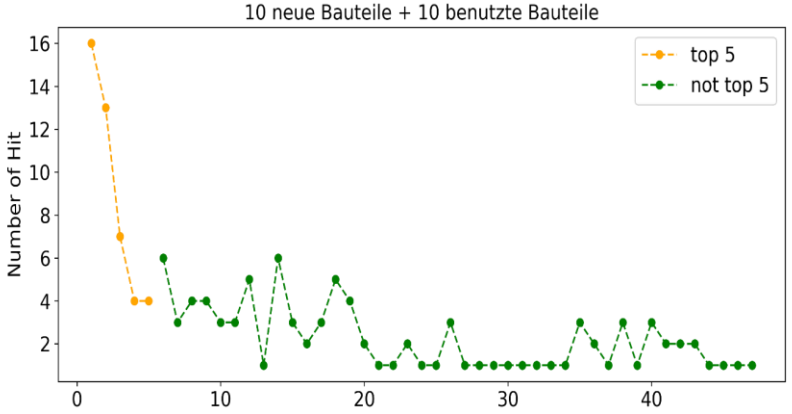
Metric : Beste Test-Konfiguration ist in top 5 der Vorhersage

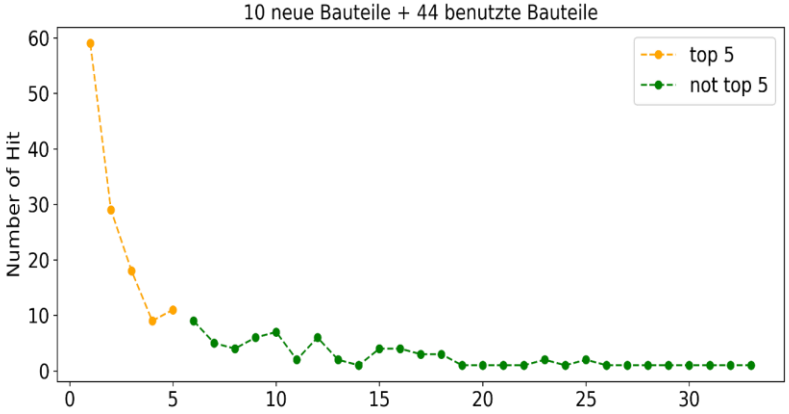
Result :

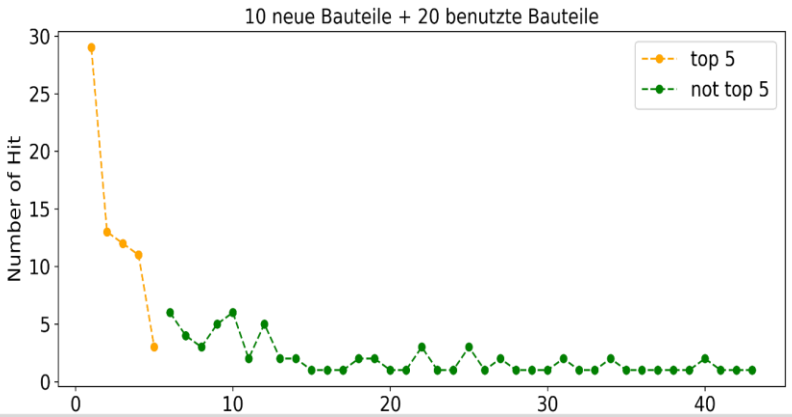


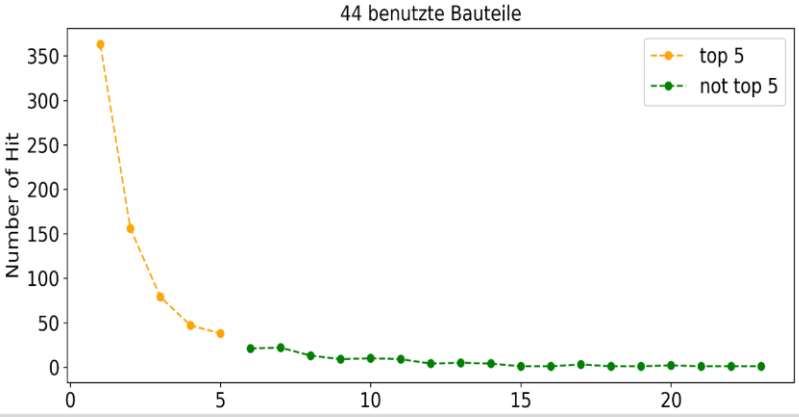












Evaluation

Experiment 3

Setup: Gewichtung ist 8: 1: 1
Anzahle der Bauteile fuer Training : 44
6 verschiedene Datensatz:
(1) 10 neue Bauteile
(2) 10 neue Bauteile + 10 benutzte Bauteile
(3) 10 neue Bauteile + 20 benutzte Bauteile
(4) 10 neue Bauteile + 30 benutzte Bauteile
(5) 10 neue Bauteile + 44 benutzte Bauteile
(6) 44 benutzte Bauteile

Metric : Beste Test-Konfiguration ist in top 5 der Vorhersage

Result :