TÜRK-ALMAN ÜNİVERSİTESİ
TÜRKISCH-DEUTSCHE UNIVERSITÄT

Erkennung von Android-Malware mithilfe von Maschinellem Lernen

Bachelorarbeit
Student : Bilal Yıldız
Betreuer : Prof. Dr. Ali Gökhan Yavuz

Gliederun

1. *Einleitung*
2. *Datensatz*
3. *Merkmale extrahieren*
4. *Modelle und experimente*
5. *Vergleich*
6. *Fazit*

Einleitung

- Was ist Malware?
- Warum Android ein großes Ziel für Malware ist?

Einleitung

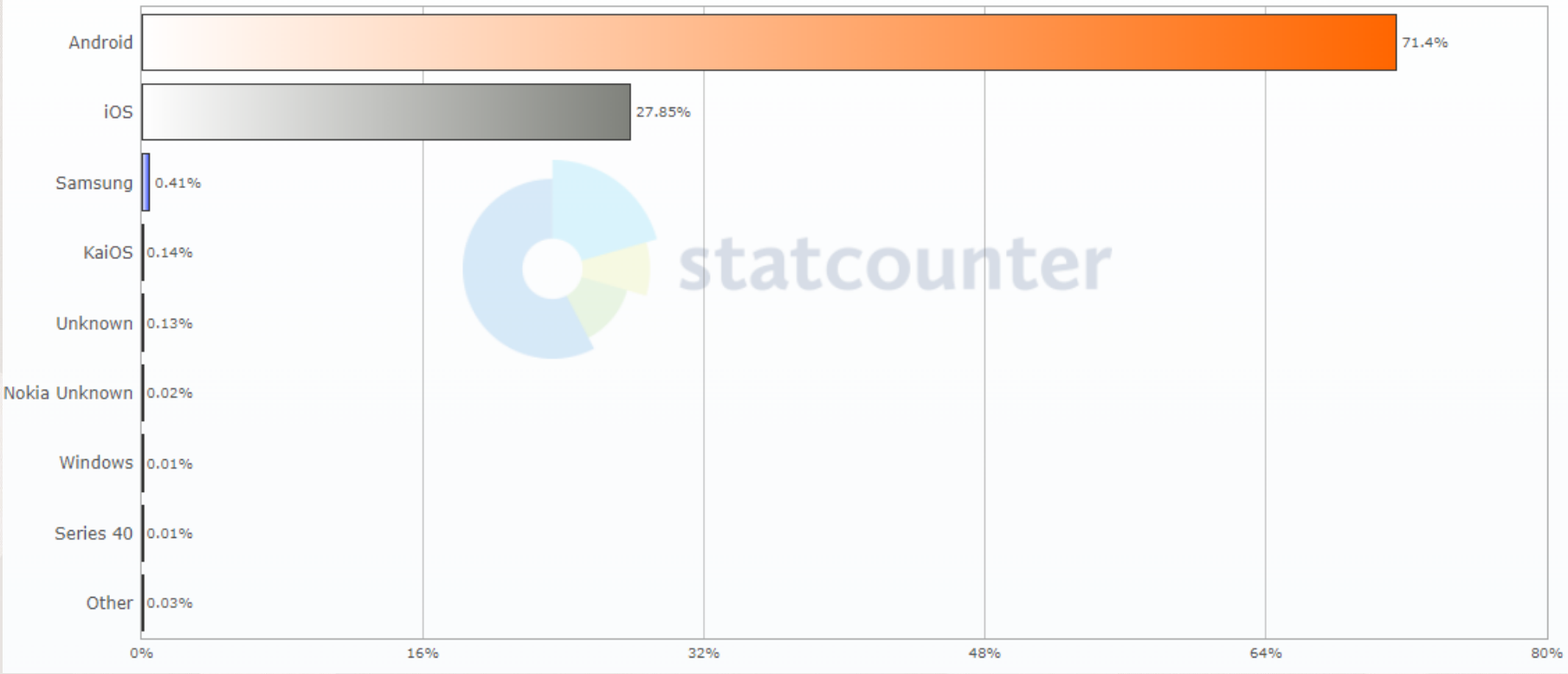

Abbildung 1: Der Prozentsatz mobiler Betriebssysteme auf der ganzen Welt [1]

Datensatz

- Der Kanadisches Institut für Cybersicherheit Datensatz[2] besteht aus 17.341 APP
- Androguard
- Anzahl der Apps nach der Analyse
  - 4758 SMS Malware
  - 2207 Banking Malware
  - 1487 Adware
  - 3236 Riskware Malware
  - 2341 Benign APP

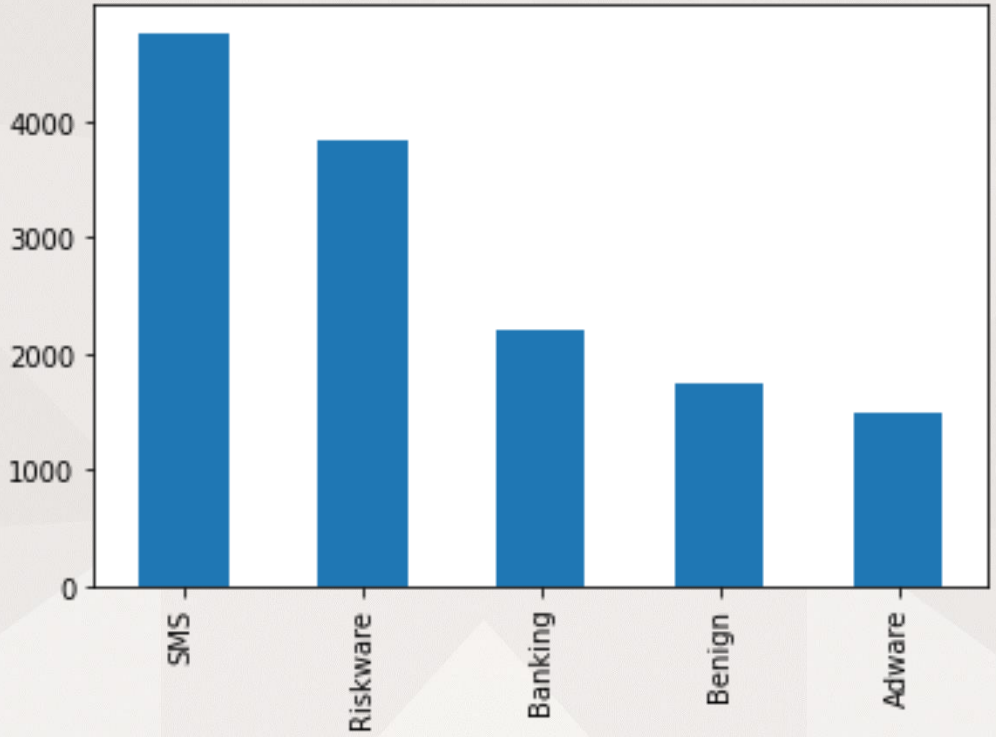

Abbildung 2: Anzahl der Apps und ihrer Klassen im Datensatz nach der Analyse

Merkmale extrahieren

- Was ist die unterschied zwischen dynamischer und statischer Analyse?

Model	Permissions	Opcode Sequence	API	Actions	Accuracy
[4]			✓		99%
[5]	✓		✓		94%
[6]	✓		✓		97%
[7]	✓				74.40%
[8]	✓		✓	✓	99.4%
[9]		✓			97%
[10]			✓		98%
diese vorgeschlagene arbeit	✓	✓			99.29%

Tablle 1: Studien, die statische Merkmale zur Klassifizierung von Android-Apps verwendet haben

Merkmale extrahieren

- Was ist Opcode?

Opcode (hex)	Opcode name	Explanation	Example
00	nop	No operation	0000 - nop
01	move vx,vy	Moves the content of vy into vx. Both registers must be in the first 256 register range.	0110 - move v0, v1 Moves v1 into v0.
02	move/from16 vx,vy	Moves the content of vy into vx. vy may be in the 64k register range while vx is one of the first 256 registers.	0200 1900 - move/from16 v0, v25 Moves v25 into v0.
03	move/16		
04	move-wide		
05	move-wide/from16 vx,vy	Moves a long/double value from vy to vx. vy may be in the 64k register range while wx is one of the first 256 registers.	0516 0000 - move-wide/from16 v22, v0 Moves v0 into v22.
06	move-wide/16		
07	move-object vx,vy	Moves the object reference from vy to vx.	0781 - move-object v1, v8 Moves the object reference in v8 to v1.
08	move-object/from16 vx,vy	Moves the object reference from vy to vx, vy can address 64k registers and vx can address 256 registers.	0801 1500 - move-object/from16 v1, v21 Move the object reference in v21 to v1.
09	move-object/16		
0A	move-result vx	Move the result value of the previous method invocation into vx.	0A00 - move-result v0 Move the return value of a previous method invocation into v0.
0B	move-result-wide vx	Move the long/double result value of the previous method invocation into vx,vx+1.	0B02 - move-result-wide v2 Move the long/double result value of the previous method invocation into v2,v3.

Abbildung 3: Ein Beispiel für Opcode-Befehle [3]

Merkmale extrahieren

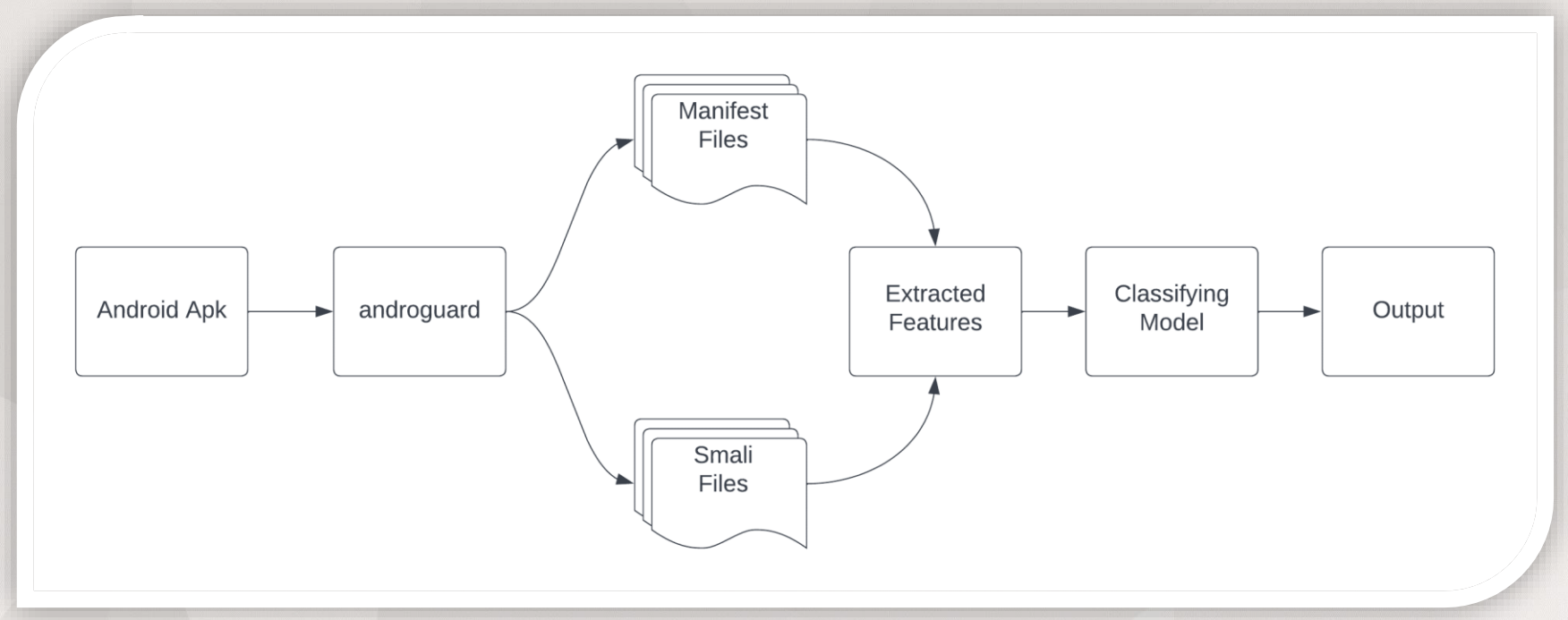

Abbildung 4: Gesamtansicht des Systems

Die Datensatz Shape nach dem Codieren aller Berechtigungen in One-Hot-Encoder und dem Extrahieren aller Opcode-Sequenzen ist 14029 x 891

```
Train shape ist (9820, 891), valX shape ist (2104, 891), testX (2104, 891)
```

Modelle und experimente

Nach mehreren Experimenten habe ich die beste Architektur für mein Basismodell gefunden :

1. Die erste Hidden Schicht mit 500 Neuronen und ohne Aktivierungsfunktion
2. Eine BatchNormalisierungsschicht
3. Dense Schicht mit 400 Neuronen und Tan-h-Aktivierungsfunktion
4. Eine DropOut-Schicht
5. Dense Schicht mit 250 Neuronen und Tan-h-Aktivierungsfunktion
6. Eine DropOut-Schicht
7. Dense Schicht mit 100 Neuronen und Tan-h-Aktivierungsfunktion
8. Dense Schicht mit 10 Neuronen und Tan-h-Aktivierungsfunktion
9. Output Layer mit 5 Neuronen und Softmax-Aktivierungsfunktion

Trainable params: 773,815

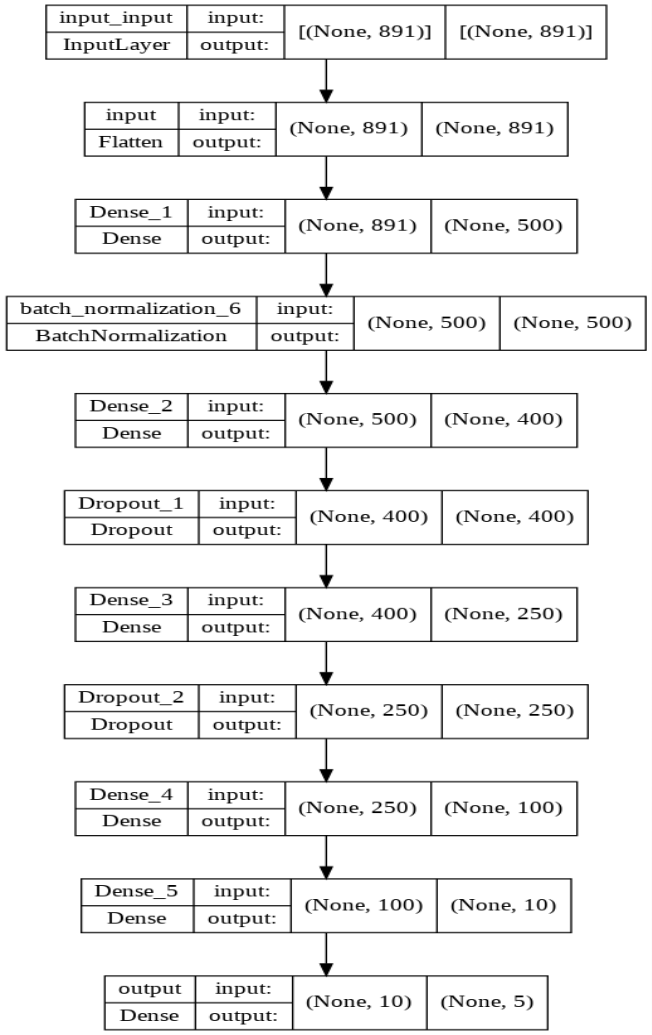

Abbildung 5: Architektur des Basismodells

Modelle und experimente

Accuracy 99.29%
Recall : 99.28%
Precision : 99.29%
F1-Score : 99.28%

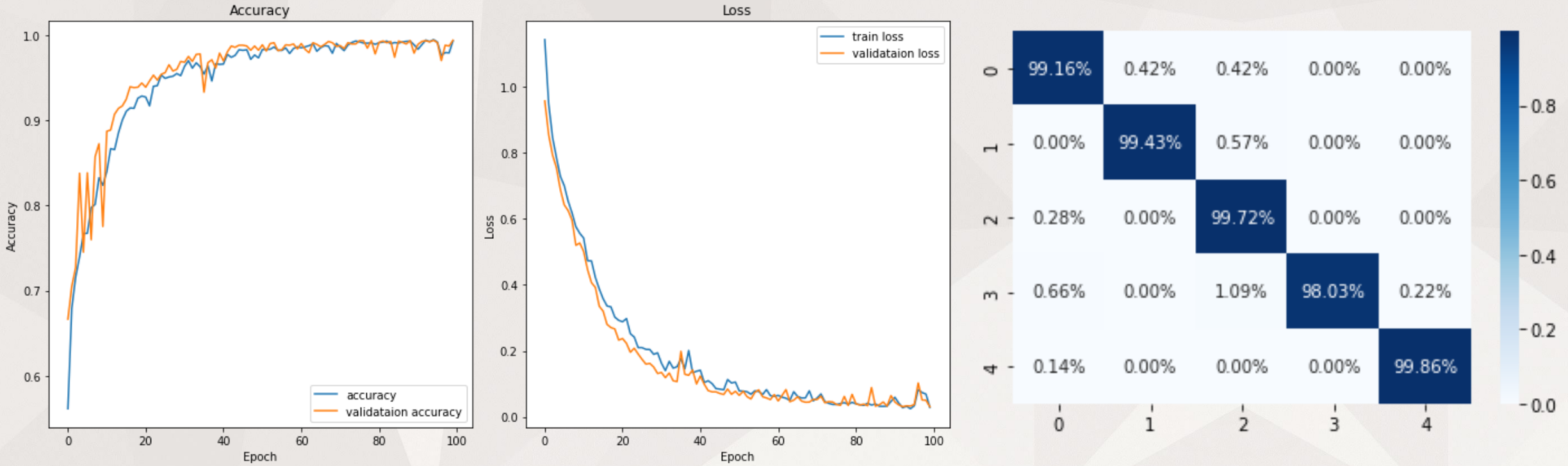

Abbildung 6: Verhalten des Basismodells Modells und Konfusionsmatrix

Methods

Machine Learning:

Wir können aus der Tabelle ersehen, dass Random Forest eine bessere Genauigkeit liefert als mein Basismodell

Abbildung 7: Random Forest Konfusionsmatrix

Model	Accuracy	Recall	Precision	F1-Score
Naive Bayes	57.40%	57.40%	58.03%	52.68%
SGD Classifier	75.67%	75.67%	78.58%	76.13%
Logistic Regression	65.34%	65.34%	66.58%	64.09%
SVM	63.86%	63.86%	66.71%	60.42%
Random Forest	99.79%	99.79%	99.79%	99.79%
Basismodell	99.29%	99.28%	99.29%	99.28%

Tabelle 2: Ergebnisse des maschinellen Lernens

Modelle und experimente

Merkmalsauswahl:

- Boruta-Algorithmus zur Funktionsauswahl verwendet

Model	Ohne Merkmalsauswahl Accuracy	Mit Merkmalsauswahl Accuracy
Naive Bayes	57.40%	59.25%
SGDClassifier	75.67%	71.61%
Logistic Regression	65.34%	64.20%
SVM	63.86%	63.84%
Random Forest	99.79%	99.81%
Basismodell	99.29%	96.63%

Tabelle 3: Architektur des Basismodells mit und ohne Merkmalsauswahl

Vergleich

DL-Droid:

DL-Droid Model besteht aus 3 verborgenen Schichten mit jeweils 200 Neuronen. dieses Modell wurde gemeinsam auf einem komplexen Datensatz (statisch und dynamisch) trainiert.

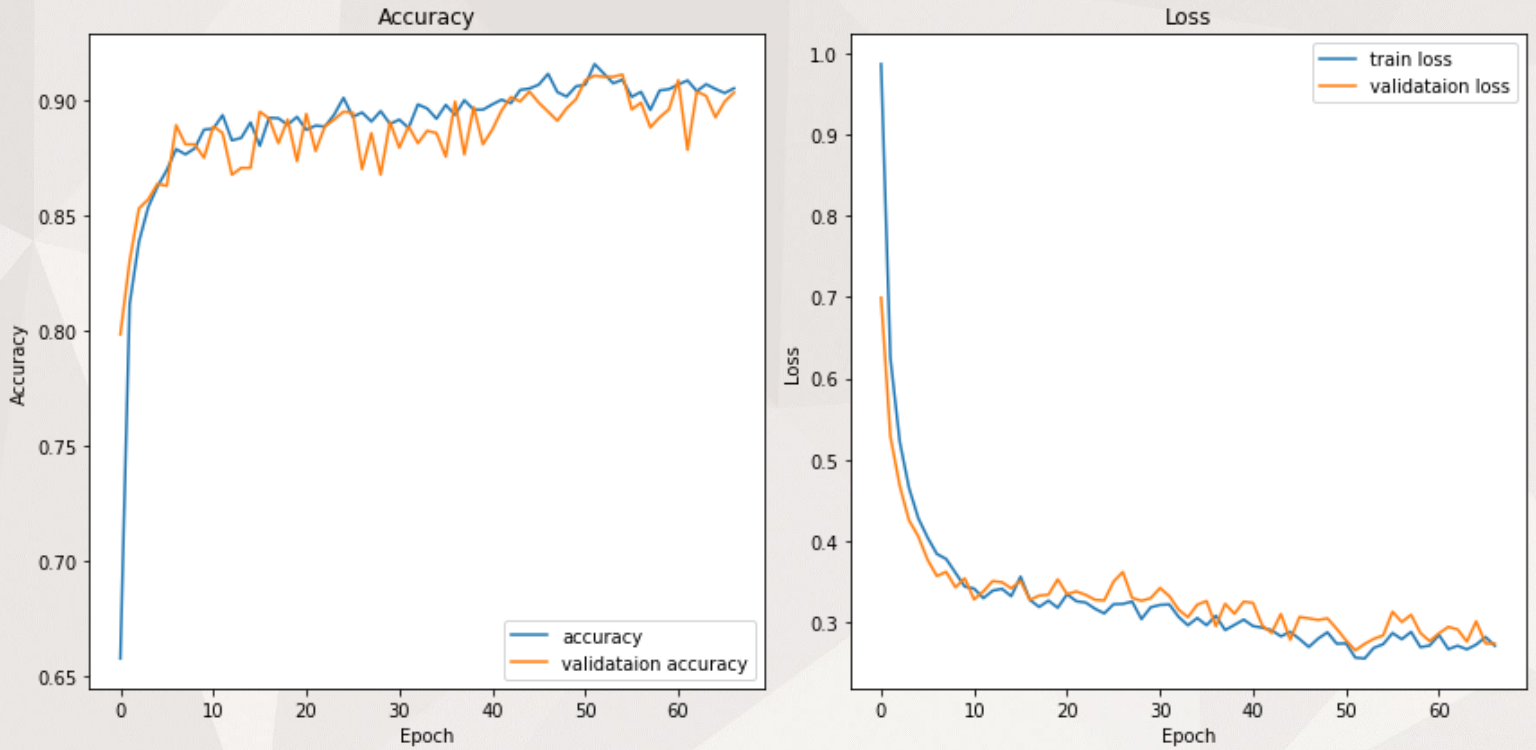

Abbildung 8: Verhalten des DL-Droid Modells

Vergleich

Droid-Sec:

Ihr Modell hat 3 Schichten mit 150 Neuronen für jede Schicht und es gab ihnen eine Genauigkeit von 96 %, aber als ich ihr Modell mit unserem Datensatz benutzen, konnte dasselbe wie das vorherige Modell keine höhere Genauigkeit als 90% erreichen.

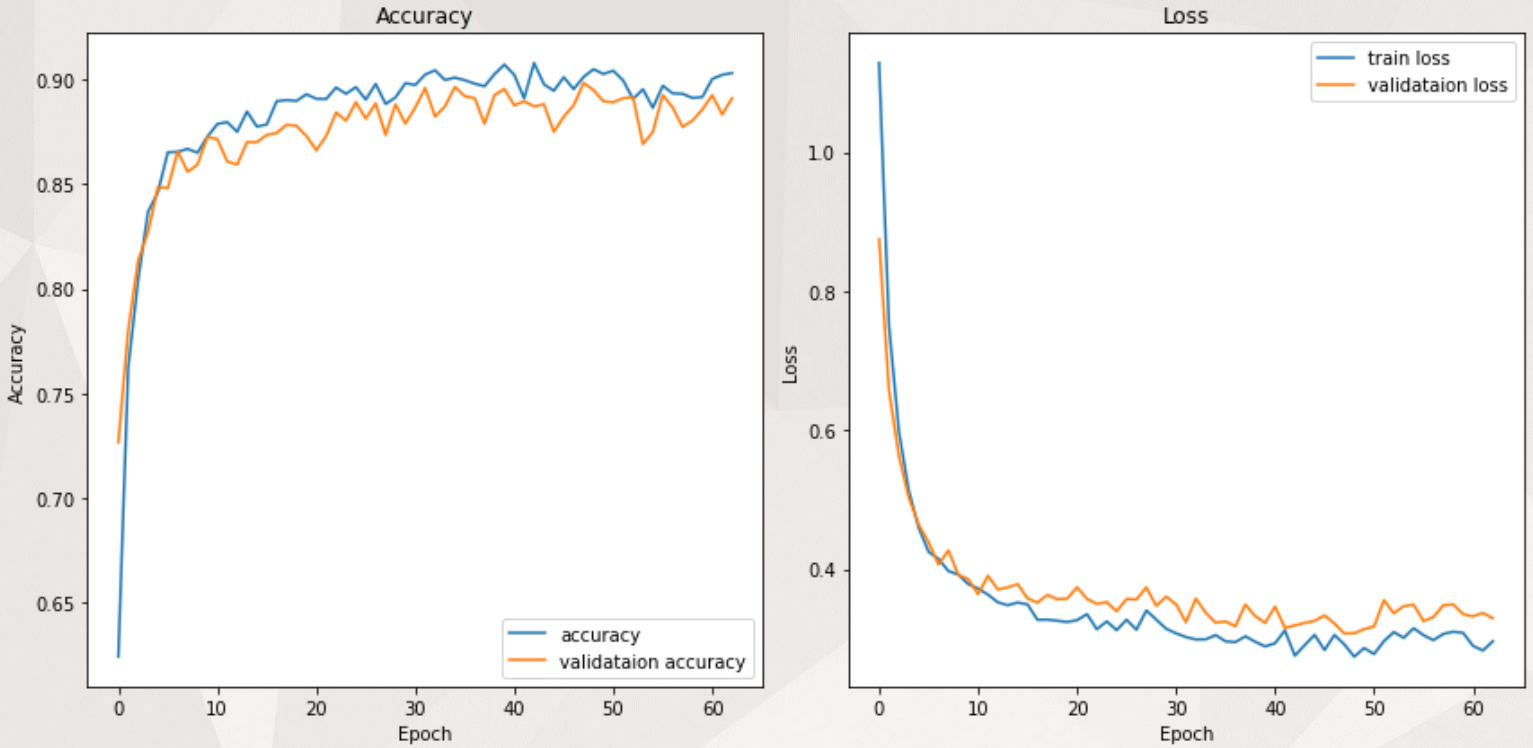

Abbildung 9: Verhalten des Droid-Sec Modells

Vergleich

ANASTASIA:

Sie trainierten das Deep-Learning-Modell mit einem ganz bestimmten Hyperparameter für 600 Epochen. Wir haben versucht, ihr Modell auf unseren Datensatz anzuwenden aber wie wir dem Verhaltensdiagramm des Modells entnehmen können, hat es bei unserem Datensatz nicht gut funktioniert. Wir haben eine Genauigkeit von nur 35%.

```
Dense(3200,activation="tanh")
Dense(1600,activation="tanh")
Dense(800,activation="tanh")
Dense(400,activation="tanh")
Dense(200,activation="tanh")
Dense(100,activation="tanh")
```

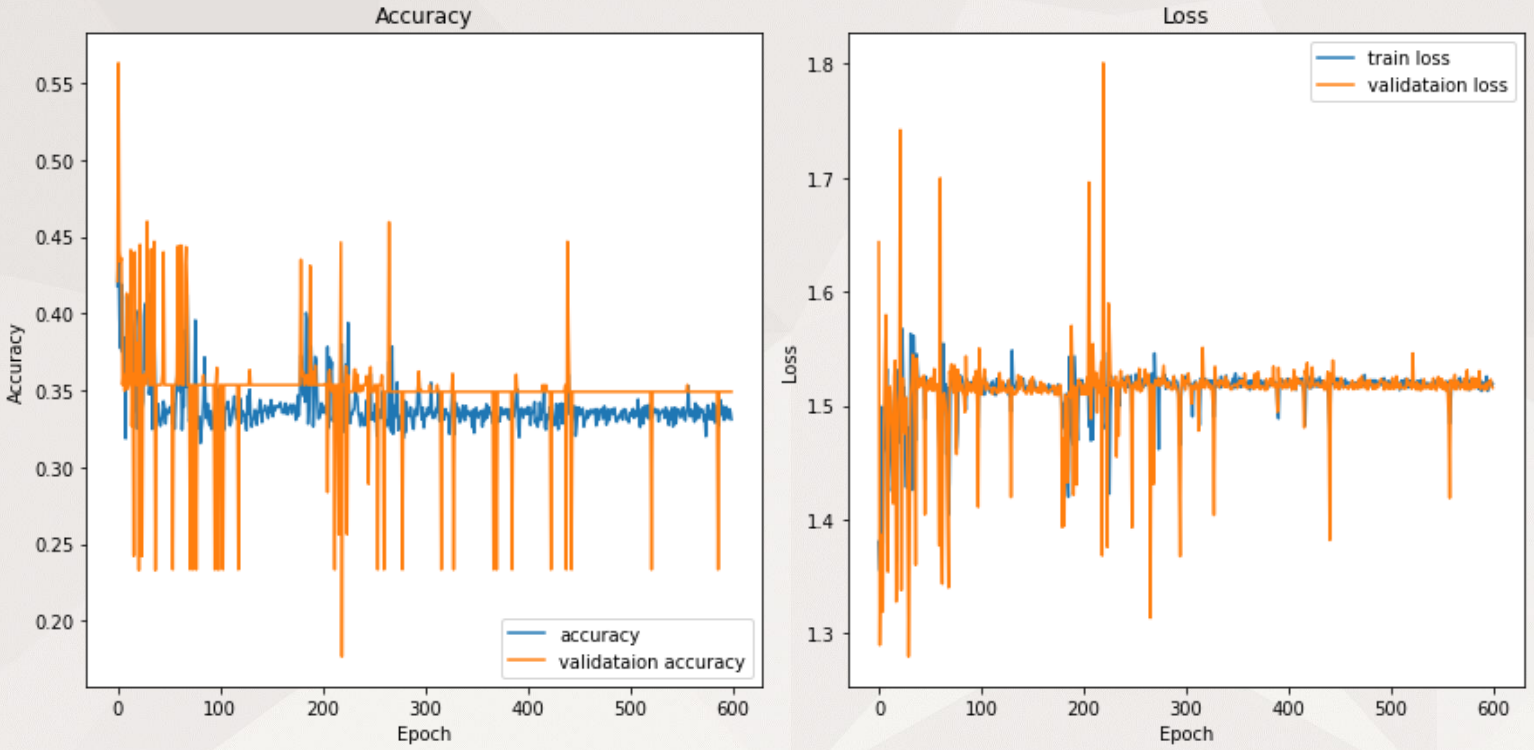

Abbildung 10: Verhalten des ANTASIA Modells

Fazit

- Die beste Leistung in Bezug auf Genauigkeit, gute Anpassung an die Daten ist das Random Forest Modell.
- Komplexeres und fortgeschritteneres Modell wie ANASTASIA nicht die bestmöglichen Leistung gegeben hat.
- Es ist nicht notwendig, immer Deep Learning zu verwenden, während wir mit einfachen Algorithmen für maschinelles Lernen dasselbe Ergebnis erzielen können..

Quellen

1. Statcounter. 2. mobile operating system market share worldwide nov 2020 – nov 2021 (percentage). @ONLINE, Dec. 2021. URL https://gs.statcounter.com/os-market-share/mobile/worldwide. Last accessed 21 December 2021.
2. DataSet Link : https://www.unb.ca/cic/datasets/maldroid-2020.html
3. Opcode Tabelle Link : http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html
4. Y. Aafer, W. Du, and H. Yin. Droidapiminer: Mining api-level features for robust malware detection in android. In T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, editors, Security and Privacy in Communication Networks, Cham, 2013. Springer International Publishing. ISBN 978-3-319-04283-1.
5. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In NDSS, 2014. DREBIN.
6. H. Fereidooni, M. Conti, D. D. Yao, and A. Sperduti. Anastasia: Android malware detection using static analysis of applications. 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pages 1–5, 2016.
7. T. N. Turnip, A. Situmorang, A. Lumbantobing, J. Marpaung, and S. I. G. Situmeang. Android malware classification based on permission categories using extreme gradient boosting. In Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology, SIET '20, page 190–194, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450376051. doi: 10.1145/3427423.3427427. URL https://doi.org/10.1145/3427423.3427427.
8. X. Su, D. Zhang, W. Li, and K. Zhao. A deep learning approach to android malware feature learning and detection. 2016 IEEE Trustcom/BigDataSE/ISPA, pages 244–251, 2016.
9. A. Yewale and M. Singh. Malware detection based on opcode frequency. In 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). IEEE, may 2016. doi: 10.1109/icaccct.2016.7831719. URL https://doi.org/10.1109%2Ficaccct.2016.7831719.
10. L. Deshotels, V. Notani, and A. Lakhotia. Droidlegacy: Automated familial classification of android malware. In Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop 2014, PPREW'14, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326490. doi: 10.1145/2556464.2556467. URL https://doi.org/10.1145/2556464.2556467.
11. Z. Yuan, Y. Lu, Z. Wang, and Y. Xue. Droid-sec: Deep learning in android malware detection. SIGCOMM Comput. Commun. Rev., 44(4):371–372, aug 2014. ISSN 0146-4833. doi: 10.1145/2740070.2631434. URL https://doi.org/10.1145/2740070.2631434.
12. M. K. Alzaylaee, S. Y. Yerima, and S. Sezer. Dl-droid: Deep learning based android malware detection using real devices. Computers Security, 89:101663, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2019.101663. URL https://www.sciencedirect.com/science/article/pii/S0167404819300161.

Danke