Eingehende Darstellung

FKZ: 03VP09292

Laufzeit des Vorhabens: 01.09.2021 – 31.12.2024

Vorhabenbezeichnung

SaMoA: Selbst-adaptives Monitoring-System für industrielle Anwendungen

1 Verwendung der Zuwendung und des erzielten Ergebnisses im Einzelnen, mit Gegenüberstellung der vorgegebenen Ziele

Die erzielten Ergebnisse und deren Gegenüberstellung mit den vorgegebenen Zielen werden im Folgenden anhand der definierten Arbeitspakete (AP) beschrieben. Hierbei bezieht sich jedes Unterkapitel auf ein AP. Die Aufgaben und Ziele jedes AP sind weiterhin auch als erstes zur Referenz gegeben.

1.1 AP 1.3 Festlegung der Bewertungsmaße

Für dieses Arbeitspaket sollen relevante Parameter (Modell-Komplexität, Datenaufkommen, Prozesskosten, Konzept und Data-Drift, etc.) zur späteren Bewertung des Innovationspotentials (vgl. AP 4) zusammen getragen werden. Hierzu zählt ebenfalls die finale Festlegung der Datengrundlage und Anwendungsszenarien für die Validierungsversuche, so dass ein sinnvoller Vergleich mit dem State-of-the-Art möglich ist.

- *Vergleich von Metriken auf Beispieldaten*
- *Programmierung von Auswerteroutinen*
- *Zusammenfassung von Ergebnissen*

Die untenstehende Auflistung listet alle Bewertungsmaße auf:

- Wirtschaftlichkeitsbetrachtung
- Reduzierung Personenstunden
- Reduktion der Rechenperformance
- Genauigkeit der Detektion
- TRL-Taxonomie Level
- Experimenteller Vergleich „manuelles Programmieren“ vs. „automatisiert“
- Vermeidung von Missverständnissen bei Problemstellungen
- Time-to-Production
- Zuverlässigkeit mind. wie bei AlexNet oder Vergleichbar
- Flexibilität
- Wartbarkeit

1.2 AP 2.3 Formale Spezifikation des Algorithmus

Ziel dieses Arbeitspaketes ist die Anpassung des Algorithmus (CGP) für die Ausführung auf einem Server. Zudem soll die Anbindung zu geeigneten Filterbibliotheken geschaffen werden. Weiterhin sollen justierbare Parameter und Operatoren definiert werden und anschließend alles Dokumentiert werden.

- *Recherche zur Auswahl von Operatoren (Abgleich mit bereits bestehendem Stand)*

- *Implementierung von Code für Vorversuche*

Die Anwendung wird momentan auf einem dafür dedizierten Rechner als Stand-Alone Programm betrieben. Eine Cloud-Anbindung, damit das Programm auf einem Server ausgeführt werden kann, ist aus diversen Gründen nicht umgesetzt. Zum einen würde die benötigte, dauerhafte Internetverbindung sich beispielsweise auf Messen als Problem erweisen. Durch die Stand-Alone Lösung lässt sich die Anwendung auch in einer abgekapselten Umgebung und in Verbindung mit einem als Demonstrator aufgebauten Kamera-Prüfstand nutzen. Zudem würde durch eine unnötige Cloud-Anwendung laufende Kosten anfallen.

Allerdings ist eine Portierung zur Ausführung auf einem Server jederzeit möglich. Der Kern der Anwendung kann problemlos als Server-basierte Anwendung integriert und portiert werden – was natürlich auch im Laufe des Projektes vorgesehen ist. Das Fraunhofer IGCV besitzt bereits die benötigte Infrastruktur, um eine Server-Anwendung zu ermöglichen. Aus Tests ergeben sich zudem, dass eine Portierung jederzeit möglich wäre.

Wie auch der vorherige Paragraf beschrieben hat, wurde der Algorithmus von Grund auf in Python Implementiert, um volle Konfigurierbarkeit und Erweiterbarkeit zu gewährleisten. Für diese Implementierung wurden zwei gängige, in der Wissenschaft und der Wirtschaft anerkannte Bibliotheken verwendet: OpenCV und SciPy. Beide Bibliotheken bieten eine ausgiebig getestete, breite Masse an diversen Bild- und Signaloperatoren. Durch die Apache 2 Lizenz sind sie frei zugänglich und ermöglichen eine einfache, schnelle und breite Nutzung.

Hierbei ist OpenCV der v.a. im universitären Umfeld der bevorzugte Weg, da die von Fraunhofer IGCV verwendete Bibliothek MVTec HALCON zwar im Prinzip die gleiche Funktionalität abdeckt, allerdings im Hinblick auf Veröffentlichungen nicht als open source freigegeben werden kann. Weiterhin entstehen durch die Verwendung von Open Source Bibliotheken keine laufenden Kosten. Vor allem in Hinblick der Projektlaufzeit wird durch diese Wahl viel Geld gespart. Daher wurde insbesondere die Anwendung insbesondere auf Basis von OpenCV und SciPy in AP 2 weiterentwickelt und optimiert.

Folgende, gängige in der Literatur zu findende Bildoperatoren werden mithilfe von OpenCV in SaMoA verwendet:

- dilate
- erode
- threshold
- Laplacian
- canny
- Gaussian blur
- Sobel
- Warp Affine
- Gabor Filter
- Median Blur
- Bilateral Filter
- Blur
- Morphology Ex

Für Signale werden folgende Operatoren aus SciPy verwendet:

- CombineTwo NormalizedSignals
- CombineTwo Signals
- ComplexRotation
- Derivative
- DifferenceFrom Avg
- FFT
- HighPass
- IFFT
- InverseGradient
- Lowpass
- MovingAvg
- MovingMax
- MovingMin
- MovingVariance
- PartitionedAvg

- PartitionedMax
- PartitionedMin
- PartitionedVar
- Rectangle
- RectangleInverted
- ThresholdAbs
- ThresholdRel
- TwoSignalsAdd
- TwoSignalsAvg
- TwoSignalsMax
- TwoSignalsMin
- TwoSignalsMultiply
- TwoSignals SelectOne
- TwoSignalsSubtract
- TwoSignalsPolar

Hierbei sind diese Operatoren nicht willkürlich gewählt. Durch ausgiebige Recherche Arbeit und damit verbundenen Experimenten haben sich diese Operatoren als vorteilhaft und positiv herauskristallisiert.

Weiterhin sind auch alle justierbaren Parameter und Variablen in einer Konfigurationsdatei abgespeichert. Diese lassen sich schnell im Text-Editor öffnen und bearbeiten; und ohne Vorkenntnisse in einer Programmiersprache besitzen zu müssen.

Zusätzlich existiert eine gute und ausführliche Dokumentation der Software, sowie Anleitungen zur Bedienung. Somit ist eine zukünftige Wartbarkeit gewährleistet und eine Bedienung von Laien.

Auch ist zu bemerken, dass sich diese Beschreibung auf eine Implementierung basierend auf Python bezieht. Dadurch ist eine Open Source Anwendung verfügbar und es fallen keine Lizenzgebühren an, wie beispielsweise durch MATLAB.

Daneben besteht auch die Möglichkeit, eine vom Fraunhofer IGCV auf C# und Halcon basierte Lösung zu verwenden – was jedoch eine kommerzielle Lösung ist. Um diese Mehrkosten zu sparen – denn es müssten neue, kostspielige Lizenzen erworben werden – verwendet die Universität Augsburg deswegen momentan die Open Source Lösung durch SciPy und OpenCV.

Eine dritte Version von CGP ist in der modernen Programmiersprache Rust gegeben. Auch diese wurde von der Universität Augsburg von Grund auf implementiert. Für diese Version können momentan noch keine Bild- oder Signalbasierte Operatoren integriert werden. Allerdings ist sie für bspw. Regressionsprobleme bestens geeignet, da sie im Vergleich zu Python oder C# um einen drei- bis vierstelligen Faktor kürzere Programmlaufzeit besitzt.

Zudem wurde auch zur Konfiguration von CGP geforscht und im wissenschaftlichen Kontext untersucht. CGP verwendet, um Lösungen zu generieren, unter anderem spezielle Mutations-Operatoren. Diese sind elementar, um erfolgreich Lösungen und Programme zu generieren. In wissenschaftlichen Arbeiten wurden verschiedene Operatoren untersucht und veröffentlicht. Eine genauere Beschreibung kann im Kapitel zu AP 5.11 „Publikation und Diskussion der Ergebnisse“ nachgeschlagen werden.

1.3 AP 2.4.2 Datenbankmodell und API-Implementierung

Implementierung der Datenbank; Implementierung des Backends (mit Django-Rest, OpenAPI o. a.); Integration und Test der Schnittstellen, Kopplung mit Cloud-Anwendung des IGCV;

- *Beitrag zu Spezifikationen von Cloud-Servern und OS*
- *Bearbeitung von Programmieraufgaben*

Aktuell werden Modelle und Datensätze in einem Datenbank-ähnlichen Format abgespeichert. Da vor allem Bilder- und Binärdateien abgespeichert werden müssen –

und keine numerischen Werte / Texte – eignet sich eine minimale, strukturierte Organisation besser.

Die erste Aufgabe dieses AP besteht aus der Spezifizierung und Implementierung des Datenbankmodelles. Hierbei müssen wir zwei verschiedene Szenarien Unterscheiden: Die Speicherung von gelernten KI-Modellen, sowie die Sicherung von restlichen relevanten Informationen. Die Recherche hat ergeben, dass es nicht sinnvoll ist, ausgelernte KI-Modelle in Cloud-Servern zu speichern. Der Grund dafür sind die großen Speichermengen, die ein KI-Modell benötigen würde. Dafür wäre für die effiziente und Nutzerorientierte Speicherung eine schnelle Internetverbindung, sowie größere Speichermengen notwendig. Weiterhin wäre auch eine Versionierung nicht möglich, da das Speicherformat von KI-Modellen keine Versionierung zulässt. Somit erhöht sich nochmals der Speicherbedarf, sollte das Modell geändert werden. Das alles führt dazu, dass sich die laufenden Kosten erhöhen – wodurch eine Nutzung von Cloud-Servern nicht sinnvoll wäre. Im Gegensatz dazu speichern wir unsere Lösung in einem Lokal auf dem Rechner betriebenen, speziell dafür entwickelten Datenbank. Hierfür fallen keine Kosten an. Weiterhin können wir somit Bilder-, Signal-, und Binärdateien in speziellen Speicheroptimierten Datenstrukturen abspeichern und laden.
Dieser lokale Rechner wird beim Fraunhofer IGCV betrieben und über eine REST-API angesprochen. Somit haben wir eine kostengünstige Äquivalenz geschaffen.
Eine auf einem Cloud-Service basierte Datenbank Lösung kann problemlos in kürzester Zeit als Cloud-Anwendung portiert werden.

Das zweite große Ziel dieses Arbeitspaketes ist die Bearbeitung von Programmieraufgaben sowie der API-Implementierung.
Für das Programm selbst muss Python installiert sein. Nun können alle benötigten Bibliotheken mithilfe einer von uns erstellten Anforderungs-Datei auf einmal installiert werden. Anschließend kann unser Algorithmus heruntergeladen und gestartet werden. Durch die manuelle Angabe des zu verwendeten Datensatzes, Speicherorte, etc. können Änderungen vorgenommen werden.

1.4 AP 2.5.1 Pipeline-Generierung auf Server Portieren

Einrichtung einer automatisierten Testumgebung für den Algorithmus (MLOps); Implementierung des auf CGP basierenden Algorithmus auf Bibliothek/Paket-Level; Hyperparameter-Optimierung

- *Schreiben einzelner Unit-Tests*
- *Zusammentragen von Testdaten*
- *Durchführen manueller Tests (Debugging)*

Eine automatisierte Testumgebung wird mittels Unit-Tests umgesetzt, welches die grundlegenden Funktionalitäten testet.
Weiterhin wurde bei Fraunhofer eine grundlegende MLOps-Umgebung angelegt, in die der Algorithmus im folgenden Berichtszeitraum integriert wird.
Allerdings sehen wir erst bei einer gewissen technologischen Reife den Mehrwert einer MLOps-Umgebung, in der dann nur noch Hyperparameter und Daten-management betrieben wird und weniger der Algorithmus verändert wird. MLOps ist vor allem zum Verwalten von KI/ML-modellen sinnvoll.
Unsere Algorithmen werden jedoch stetig verbessert und erweitert. Durch diese konstante Veränderungen haben wir deswegen keine ausführlichen Integrationen in die MLOps-Umgebungen vorgenommen.

Bezüglich einer CGP-Implementierung auf Bibliothek / Paket-Level existiert eine fertige Version in Python und Rust. Durch die Verwendung von OpenCV werden keine proprietäre (Software-)Pakete verwendet.
Diese Basis-Implementierung wurde in diversen Papern (bspw.: DOI: 10.1007/978-3-031-21094-5_14) durch Erweiterungen verbessert. Eine genauere Beschreibung des Inhaltes wird in Kapitel 1.2. gegeben.
Weiterhin ist auch eine automatisierte Hyperparameter Optimierung möglich. Hierfür verwenden wir die Open-Source Software Optuna[1]. Durch eine vollständige Integration dieser Bibliothek in unsere Software können beliebige Hyperparameter optimiert und dem User ausgegeben werden.

1.5 AP 3.4.1 Analyse erzeugter Pipelines

Analyse der Pipelines; Erstellen eines Überblicks über verwendete Filter und Operatoren; Beitrag zu Spezifikationen von Cloud-Servern und OS

- *Analyse und Vergleich von Ergebnissen*
- *Hilfe bei kritischer Betrachtung*

Die Analyse erzeugter Pipelines wurde erfolgreich durchgeführt und veröffentlicht[2].

Die Evaluation wurde hierbei mithilfe von zwei Sensor-Datensätzen durchgeführt, und es konnten durchweg positive Ergebnisse präsentiert werden.

Weiterhin geht die Arbeit auch sehr stark auf die Einflüsse der unterschiedlichen Hyperparameter ein. Somit kann ein Überblick über deren Auswirkungen gegeben werden, wodurch auch eine Empfehlung von Standardparametern gegeben werden kann.

Im selben Zuge wurde auch in einer nicht veröffentlichten Arbeit der Einfluss der jeweiligen Operatoren ausgewertet, wie in Abbildung 1 zu sehen ist.

Beispielsweise haben bestimmte Formen des Hochpassfilters einen sehr starken Einfluss auf das Ergebnis.

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.
[2] Andreas Margraf, Henning Cui, Stefan Baumann and Jörg Hähner. 2023. Filter evolution using Cartesian genetic programming for time series anomaly detection. In Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, Joaquim Filipe (Eds.). Proceedings of the 15th International Joint Conference on Computational Intelligence - ECTA, November 13-15, 2023, in Rome, Italy. SciTePress, Setúbal, 300-307 DOI: 10.5220/0012210700003595

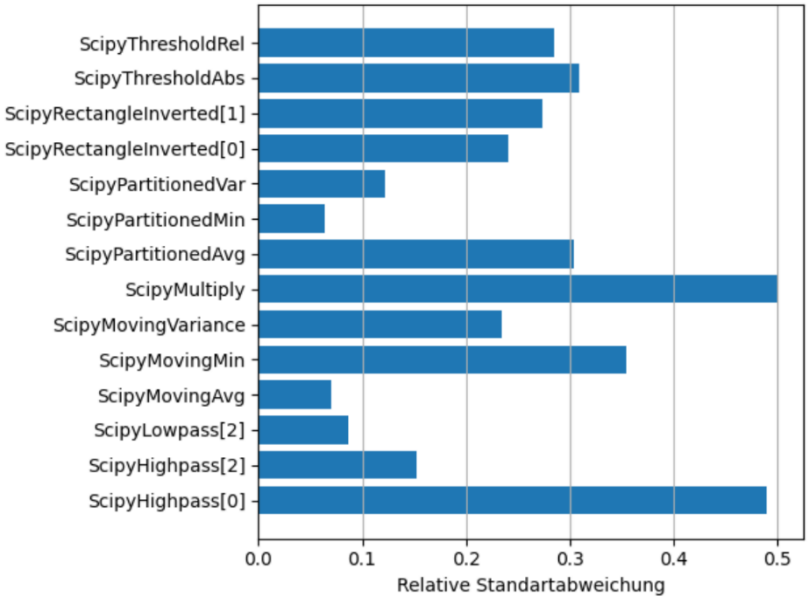

Abbildung 1: Einfluss verschiedener Operatoren auf die automatisierte Signalverarbeitung

In dieser Arbeit wurde weiterhin auch die Anzahl der Vorkommnisse der Operatoren betrachtet (vergleich Abbildung 2). So wird von unserem Algorithmus beispielsweise sehr oft ein relativer Threshold angewandt.

Diese Erkenntnisse ermöglichen eine genauere Untersuchung generierter Pipelines, und erlauben eine gezielte händische und/oder automatisierte Optimierung. Hiermit wird auch eine kritische Betrachtung der erzeugten Pipeline ermöglicht. Durch ein besseres Verständnis der Funktionen, ihrer Wichtigkeit, sowie dessen Einflüsse auf das Endergebnis, können auch bessere Lösungen generiert werden.

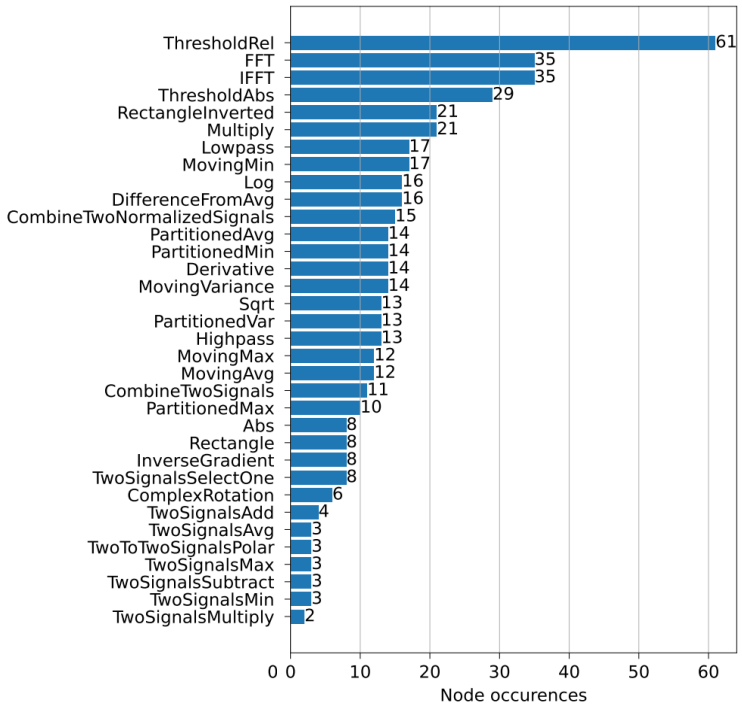

Abbildung 2: Häufigkeit der Operatoren. Sortiert nach Häufigkeit.

Eine andere Publikation[3] beschreibt unter anderem die Analyse von Pipelines für die Bildverarbeitung. Hierbei wurde sich auch vor allem auf die Einflüsse der verschiedenen Hyperparameter berücksichtigt. Das dient erneut dem besseren Verständnis, wodurch wir die Algorithmen besser auf ihre Problemstellungen abstimmen können. Außerdem können im Falle einer neuen Problemstellung von vornherein gute Hyperparameter verwendet werden. Dadurch kann auch die weitere Hyperparameteroptimierung schneller und effizienter durchgeführt werden, da schon ein passabler Ausgangszustand vorherrscht. Somit verkürzt sich auch die Zeit, bis ein neuer Algorithmus finalisiert ist.

1.6 AP 3.4.2 Bewertung anhand von Metriken

Bewertung der Güte, Performance der Ergebnisse, Vergleich zwischen Standard-Klassifikator (ANN) und des SaMoA-Algorithmus; Analyse des Potentials auf industrienahen Datensätzen (AP 1.2, 1.3); Beitrag zu Spezifikationen von Cloud-Servern und OS

- *Analyse und Vergleich von Ergebnissen*
- *Hilfe bei kritischer Betrachtung*

Um die Güte unseres SaMoA Algorithmus bezüglich der Bildverarbeitung zu bewerten, haben wir es mit einem gängigen Neuronalen Netzwerk (NN) verglichen: dem ResNet50. Wir haben dieses Neuronale Netzwerk gewählt, da es frei verfügbar und vergleichsweise niedrige Hardware-Anforderungen besitzt. Weiterhin ist es auch ein gängiges Model, welches in der Industrie des Öfteren verwendet wird.

Evaluation anhand CampusCat Datensatz

Zum einen haben wir beide Algorithmen auf einen eigenen kleinen Test-Datensatz namens CampusCat angewandt. Dieser Datensatz besteht aus 50 Hand-gelabelten Bildern einer Plüschkatze auf weißem Hintergrund. Durch eine seitliche Lichtquelle können die Bilder unterschiedliche Schattierungen besitzen.

Abbildung 3: Trainingsverlauf vom ResNet 50 auf dem CampusCat Datensatz zeigt den Trainingsverlauf vom ResNet50, welches bis zur Konvergenz trainiert wurde. Es erreicht eine Intersection Over Union (IOU) von 0.934 – was bedeutet, dass ungefähr 93% aller Pixel richtig zugeordnet sind.

[3] Andreas Margraf, Henning Cui, Simon Heimbach, Jörg Hähner, Steffen Geinitz and Stephan Rudolph. 2023. Model-driven optimisation of monitoring system configurations for batch production. In Francisco José Domínguez Mayo, Luís Ferreira Pires, Edwin Seidewitz (Eds.). Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering MODELSWARD 2023, February 19-21, 2023, in Lisbon, Portugal. SciTePress, Setúbal, 176-183 DOI: 10.5220/0011688900003402

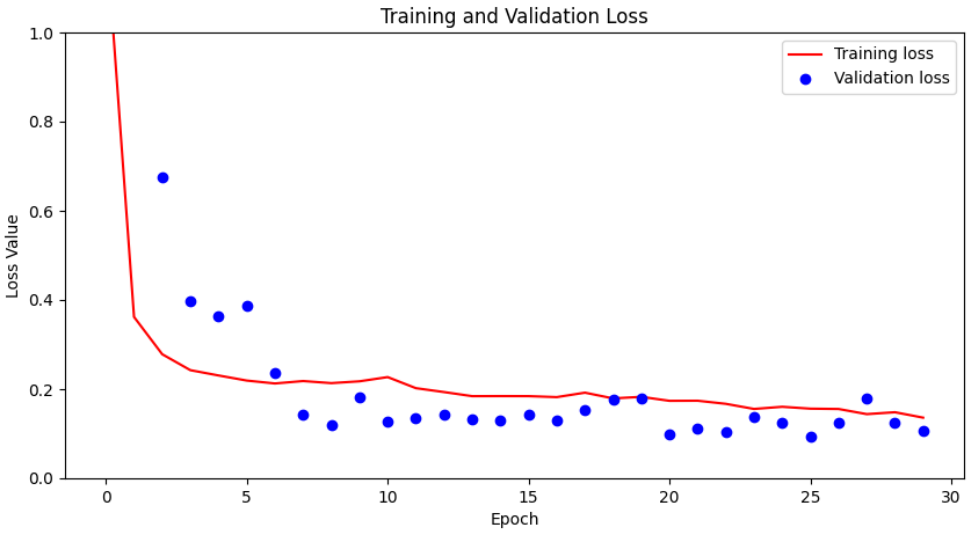

Abbildung 3: Trainingsverlauf vom ResNet 50 auf dem CampusCat Datensatz

Eine beispielhafte Abbildung der Ausgabe wird in Abbildung 4: Zufällig gezogene Ausgabe vom fertig trainierten ResNet50 dargestellt, welches ein Eingabebild, das dazugehörige Label, und die ausgegebene Vorhersage abbildet. Die hier zu sehende Vorhersage stammt aus einer zufällig gezogenen Abbildung – stellt allerdings sehr gut auch die restlichen Vorhersagen dar. Hierbei besteht das Problem, dass alle Vorhersagen diesen „unförmigen Kreis“ abbilden. Details wie das „Diplom mit der roten Schleife“ (links im Ursprungsbild zu sehen), oder der Schwanz der Katze, gehen bei der Segmentierung verloren.

Abbildung 4: Zufällig gezogene Ausgabe vom fertig trainierten ResNet50

Im Gegensatz dazu steht der SaMoA-Algorithmus, dessen Trainingsverlauf in Abbildung 5 zu sehen ist. Ähnlich zum NN wird sehr schnell eine vergleichsweise gute Güte erreicht.

Bezüglich der IOU werden 85% aller Pixel richtig klassifiziert.

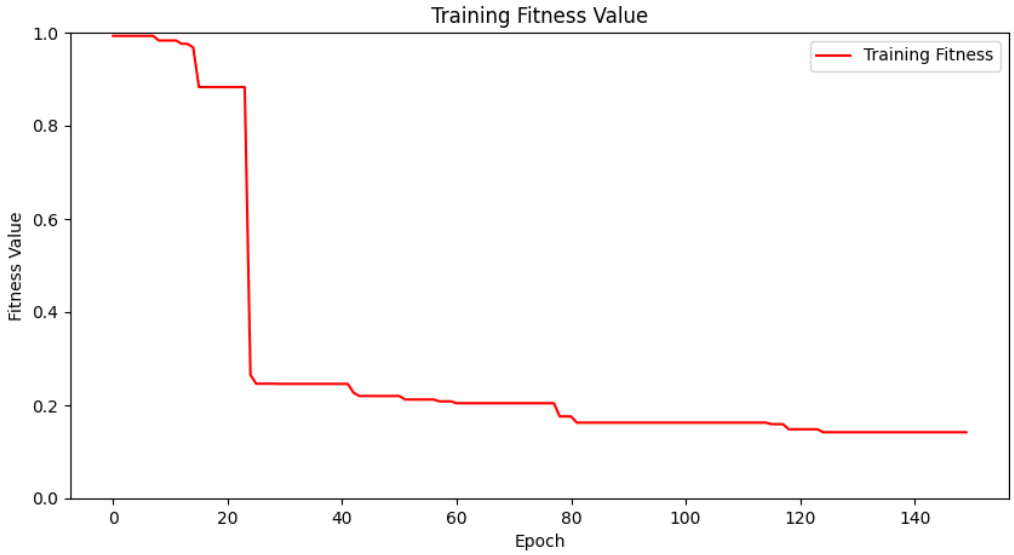

Abbildung 5: Trainingsverlauf vom SaMoA-Algorithmus auf dem CampusCat Datensatz

Im Gegensatz zum ResNet50 sind die Segmentierungen jedoch keine unförmigen Kreise, sondern ähneln dem Eingabebild sehr stark. Formen und Konturen sind auch eindeutig erkennbar. Allerdings kann die ungleiche Schattierung des Hintergrundes leichte Probleme bereiten, wie in der Ausgabe des Algorithmus der beiden rechten Bilder zu sehen ist.

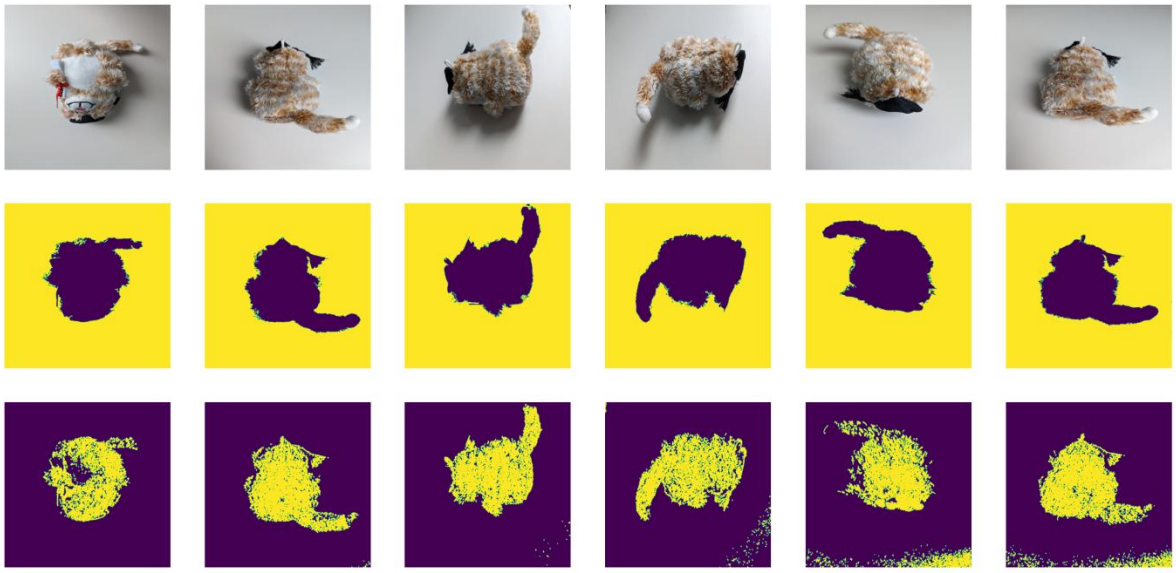

Abbildung 6: Beispielhafte Ausgaben vom SaMoA-Algorithmus. Oben: Eingabebild, Mitte: Label, Unten: Ausgabe des Algorithmus

Weiterhin besteht der Vorteil, dass unser Algorithmus „Shift-Invariant“ ist. Ein großes Problem von Neuronalen Netzen ist, dass sie *nicht* Shift-Invariant sind[4]. Das bedeutet, dass die Performanz stark von der Position des zu erkennenden Objektes im Bild abhängt. Befindet sich die Plüschkatze beispielsweise rechts oben im Bild, anstatt in der Mitte, kann das die Güte vom ResNet50 bemerkbar senken. Im Gegensatz dazu hat unser Algorithmus das Problem nicht. Die

[4] Zhang, Richard. "Making convolutional networks shift-invariant again." *International conference on machine learning*. PMLR, 2019.

Position des Plüschtieres kann beliebig innerhalb des Bildes verschoben werden, ohne die Performanz negativ zu beeinflussen.

Ein anderer Vorteil ist, dass unser Algorithmus schneller Segmentierungen erzeugen kann. Die Trainingszeit beider Modelle ist ungefähr gleich lang. Allerdings benötigt das fertig trainierte ResNet50 mehr Zeit, ein einzelnes Bild zu segmentieren als unser SaMoA-Algorithmus. Der Grund dafür ist, dass das Bild für das NN im Vorhinein herunter skaliert werden muss, und anschließend werden sehr viele Matrixmultiplikationen durchgeführt. Im Gegensatz dazu müssen die Bilder für unseren Algorithmus nicht herunterskaliert werden. Durch die zusätzliche Möglichkeit der Parallelisierung von Operatoren können auch Teile von unserem Algorithmus gleichzeitig ausgeführt werden, wodurch noch mehr Zeit gespart werden kann.

Diese Zeitersparnis ist für unser Einsatzgebiet besonders interessant, da somit am Ende ein höherer Durchsatz an geprüften Produkten möglich ist.

Evaluation anhand diverser Industrie-Datensätze

Vergleicht man nun die verschiedenen Lernalgorithmen anhand verschiedener Industrie-Datensätzen, kommt es zu einem durchwachsenen, aber Interessanten Ergebnis.

Die folgende Tabelle enthält den Vergleich von CGP mit einem Fully Connected Network (FCN) sowie einem U-Net. Die Werte darin geben die Accuracy des Modells an (höher ist besser).

Datensatz	FCN	U-Net	CGP
Metallmuttern[5]	0.88	0.97	0.82
Fließen[5]	0.69	0.88	0.52
Textilien[6]	0.00	0.12	0.09
Magnetplatten[7]	0.67	0.78	0.24
Stahl[8]	0.39	0.59	0.49

Insgesamt lässt sich schlussfolgern, dass die Qualität von CGP stark vom Datensatz abhängt. Für „Metallmuttern“ beispielsweise erreicht CGP eine hohe Güte – und Punktet vor allem dadurch, dass das Training des Modells weniger Aufwendig ist als für ein FCN oder U-Net.

Für den Fließen-Datensatz ist die Accuracy vergleichsweise gering. Betrachtet man allerdings die Segmentierungen genauer, fällt einem die Ursache dafür auf: Die Klasse der Klebestreifen wird nicht erkannt (siehe Abbildung 7). Da diese Klasse allerdings die Accuracy stark beeinflusst, wirkt sich das auch negativ auf die erreichte Güte aus – obwohl die anderen Klassen gut segmentiert werden können.

Trotz alledem kann die Performanz von CGP mit einem FCN verglichen werden. Dabei hat CGP den Vorteil, dass das Training und die Generierung von Segmentierungen viel Ressourcensparender ist als bei FCN’s. Dadurch wird Energie gespart und auch die Ausführung auf kleineren Mikrocontroller bzw. leistungsschwachen PC’s ermöglicht.

[5] https://www.mvtec.com/company/research/datasets/mvtec-ad
[6] https://www.aitex.es/afid/
[7] https://www.stanfordmagnets.com/classification-application-of-magnetic-tiles.html
[8] https://www.kaggle.com/competitions/severstal-steel-defect-detection

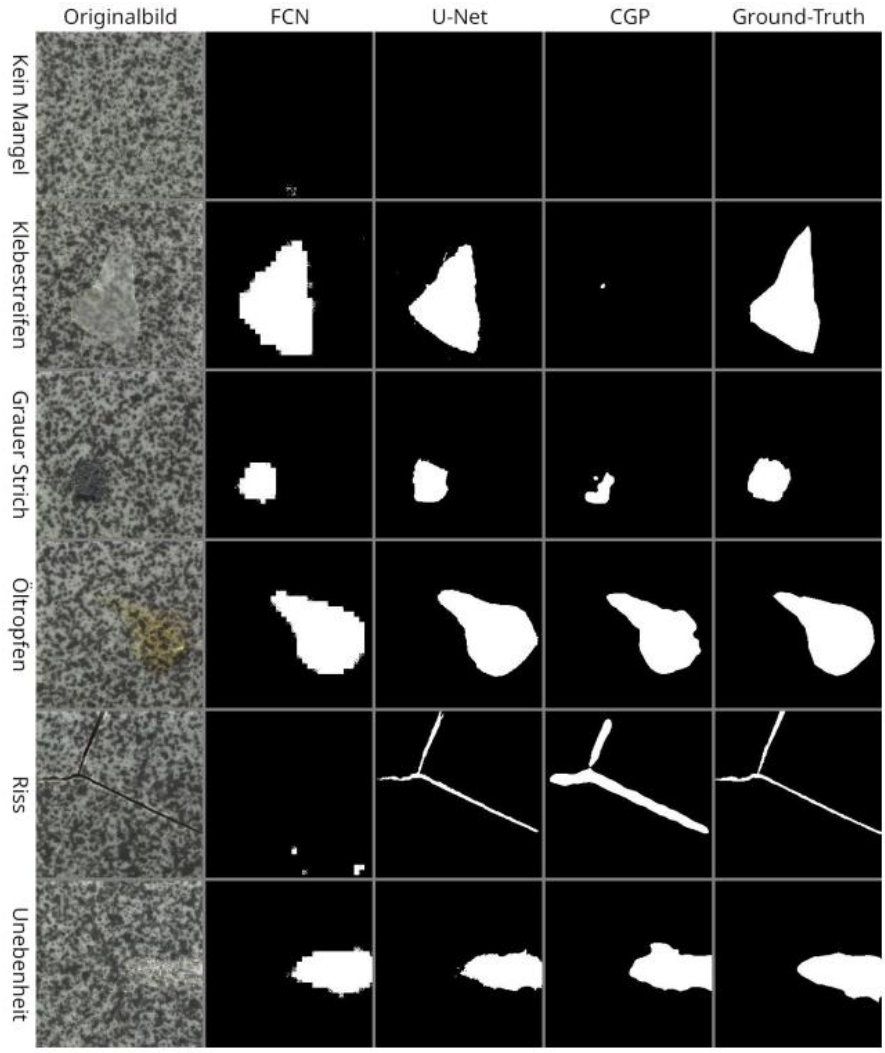

Abbildung 7: Segmentierungen vom Fließen-Datensatz

Bewertung: Benötigte Anzahl an Daten

Für CGP lässt sich beobachten, dass nur eine geringe Anzahl an Datenpunkten nötig ist, um es bis zur Konvergenz zu trainieren. Das lässt sich sehr gut durch Abbildung 8 verdeutlichen. Anhand vieler verschiedener industrieller Datensätze lässt sich gut der Vergleich zwischen erreichter Genauigkeit und benötigten Daten beobachten. Viele Datensätze Konvergieren schon mit unter 20 Datenpunkten auf eine sehr gute Genauigkeit. Durch das Hinzufügen weitere Daten kann zwar mitunter die Performanz ein bisschen verbessert werden, führt jedoch zu keiner statistisch relevanten Verbesserung. Andere Datensätze wiederum profitieren stark durch den Zuwachs an Datenpunkten. Das beste Beispiel dafür ist der „screw“ Datensatz.

Durch diese Ergebnisse lassen sich nun auch Entscheidungsbäume ableiten. Mithilfe von Komplexitätsmetriken (Bspw. Objektgröße, Varianz pro Datensatz, etc.) können Vorhersagemodelle erstellt werden. Diese können dabei helfen, die benötigte Datenmenge gegenüber einer gewünschten Genauigkeit grob abzuschätzen. Ein beispielhafter Auszug ist in Abbildung 9 gegeben.

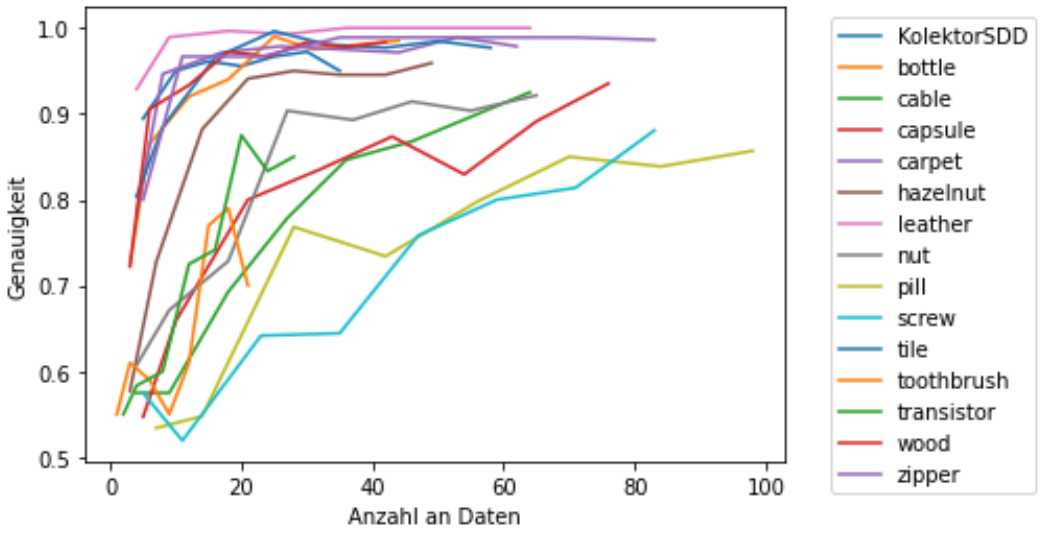

Abbildung 8: Korrelation zwischen Anzahl an Datenpunkten und erreichter Genauigkeit

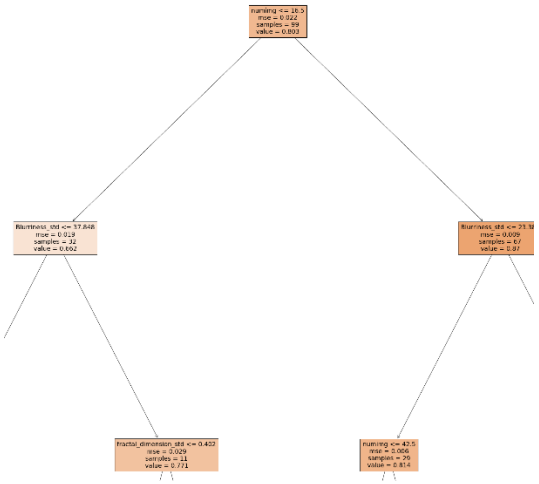

Abbildung 9: Beispielhafter Auszug eines Entscheidungsbaumes zur Bestimmung der benötigten Datenpunkte

Bewertung der benötigten Ressourcen

In diesem Bericht wird wiederholt angesprochen, dass CGP weniger Rechenleistung benötigt als Neuronale Netze. Diese Aussage kann mithilfe der Publikation von Gholami et al.[9] bestätigt werden. In ihrer Arbeit bestätigen sie, dass die neueren Architekturen und Deep-Learning Modelle immer komplexer werden. Das heißt: Sie müssen mehr Berechnungen für ein Ergebnis durchführen.

Die Trainingskomplexität eines ResNet aus dem Jahre 2016 benötigt beispielsweise ungefähr 10.000 Peta Flops. CGP benötigt unserer Berechnung nach nur durchschnittlich 500 Millionen Flops. Das entspricht einem Faktor von $2 \cdot 10^{10}$ weniger Flops für CGP. Dieser Vergleich, sowie der Vergleich mit anderen gängigen Architekturen, wird in Abbildung 10 verdeutlicht.

[9] Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M. W., & Keutzer, K. (2023). AI and Memory Wall. arXiv.org. https://arxiv.org/abs/2403.14123

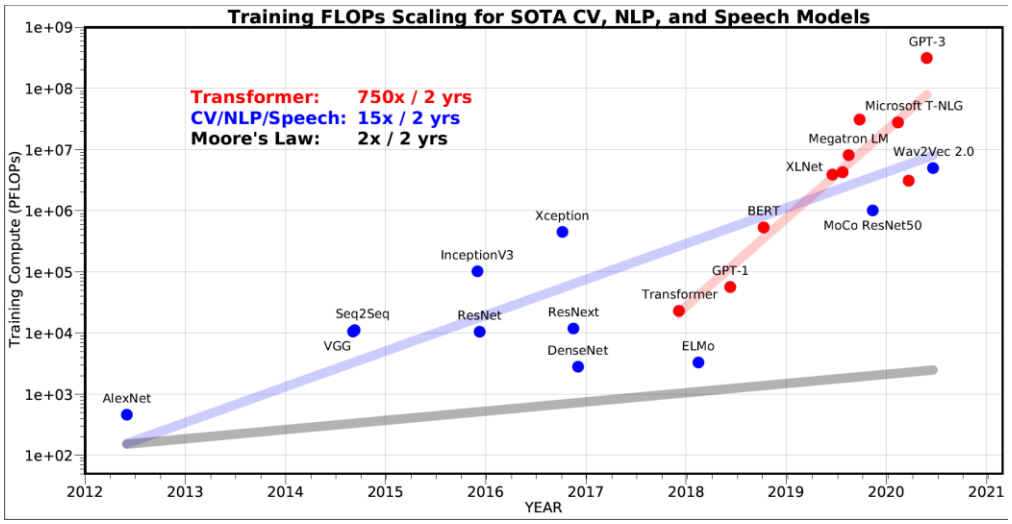

Abbildung 10: Ressourcennutzung von gängigen Deep-Learning Architekturen in Peta Flops nach Gholami et al.[10].

1.7 AP 3.5 Dokumentation der Performance / Benchmarking

Bewertung der Effizienz des Systems und des Algorithmus; Gegenüberstellung zum StotA durch quantitative Evaluierung des Algorithmus; Test der Server-Portierung; Konzept zur Skalierbarkeit im industriellen Kontext erarbeiten; Definition wirtschaftlicher Kennzahlen;

- *Testdurchlauf des Algorithmus starten und überwachen*
- *Ergebnisse Dokumentieren*

Die Effizienz des Systems und der Algorithmen wurden schon in bisherigen Arbeitspaketen erwähnt (AP 3.4.1 Analyse erzeugter Pipelines, AP 3.4.2 Bewertung anhand von Metriken).

Zusammengefasst bieten die Algorithmen einen Mehrwehrt für das System und liefern zufriedenstellende Ergebnisse. Durch den Einsatz von neuronalen Netzen kann zwar die Qualität verbessert werden, doch damit steigen auch die Hardwareanforderungen. Zusätzlich ist durch das Nutzen von neuronalen Netzen auch die Interpretierbarkeit unmöglich.

Somit bietet unser Algorithmus ausreichende Performanz, mögliche Interpretierbarkeit sowie niedrige Hardware-Anforderungen.

Eine quantitative Evaluierung des Algorithmus ist beispielsweise in einer Veröffentlichung[11] zu sehen. Darin betrachten wir Cartesian Genetic Programming (CGP) – eine der Hauptalgorithmen in SaMoA – sowie mögliche Verbesserungen davon. Wir testen unsere Algorithmen anhand von Boolean Benchmarks und evaluieren unsere Ergebnisse mit neuesten Ansätzen aus der Bayesian Stochastik. Wir zeigen in dieser Veröffentlichung, dass wir zum einen schneller und zum anderen kleinere Programme erzeugen können.

In einer veröffentlichten Arbeit haben wir auch unsere Versuchsreihe ausgeweitet. Hierbei betrachten wir zusätzlich Benchmarks aus der symbolischen Regression und führen drei neue

[10] Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M. W., & Keutzer, K. (2023). AI and Memory Wall. arXiv.org. https://arxiv.org/abs/2403.14123

[11] Henning Cui, Andreas Margraf and Jörg Hähner. 2023. Equidistant Reorder operator for Cartesian Genetic Programming. In Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, Joaquim Filipe (Eds.). Proceedings of the 15th International Joint Conference on Computational Intelligence - ECTA, November 13-15, 2023, in Rome, Italy. SciTePress, Setúbal, 64-74 DOI: 10.5220/0012174100003595

Erweiterungen für CGP ein. Auch darin zeigen wir, dass unser Algorithmus, bzw. unsere Erweiterungen, die Performanz auf vielen Ebenen verbessert.

Bezüglich des Punktes „Test der Server-Portierung“ können wir auch positive Erfolge benennen. Um die SaMoA Applikation zu installieren, werden Python 3, ein Produktschlüssel von DC43 Version 2, sowie eine Internetverbindung benötigt.

Damit lässt sich unsere Software in mehreren Einzelschritten installieren. Ein problemloser Installationsablauf hat sich bereits in der Vergangenheit gezeigt, da wir unsere Software beispielsweise auf mehrere Rechner für diverse Messen portieren mussten.

1.8 AP 4.2.1 Datenbereitstellung und Pipelineoptimierung

Annotation der Testdaten; Übertragung der Daten an die Software; Dokumentation der Pipelineoptimierung; Paralleles Trainieren von Klassifikatoren (neuronale Netze) mit entsprechendem Umfang von Trainingsdaten (mind. 100 annotierte Datenpunkte); Dokumentation der Zeitaufwände für die Validierung;

- *Unterstützung beim Anlegen einer Datenbank*
- *Recherche, Aufbereitung und Einarbeitung in Repository*
- *Einzelne Annotationsaufgaben*

Für dieses Arbeitspaket wurde sich auf die Recherche sowie Aufbereitung von Datensätzen konzentriert. Hierbei haben sich folgende Datensätze hervorgetan:

- The MVTec Anomaly Detection Dataset
- Severstal: Steel Defect Detection

Beide Datensätze sind industrieller Natur und dadurch durchaus für SaMoA geeignet. Weiterhin werden sie auch in vielen, diversen wissenschaftlichen Veröffentlichungen verwendet. Das verstärkt wiederum auch ihre Qualität und wissenschaftlichen Nutzen und Bedeutung. Dadurch, dass beide Datensätze auch industrieller Natur sind und durch führende Firmen erstellt wurden, besitzen sie auch einen starken industriellen Bezug – was wiederum für SaMoA sehr wichtig ist.

Weitere potenzielle Datensätze sind:

- „Textures Classification dataset“ aus der wissenschaftlichen Arbeit "A compact convolutional neural network for surface defect inspection"
- SDNET2018: A concrete crack image dataset for machine learning applications
- TILDA Textile Texture Database
- Textures under varying Illumination 2006
- Kylberg Texture Dataset v. 1.0

Alle gerade erwähnten Datensätze besitzen auch einen Bezug zu SaMoA, sind allerdings aufgrund qualitativen oder anderweitigen Bedenkens nicht in der oberen Liste aufgeführt.

Das Fraunhofer stellt zusätzlich auch eigene Datensätze bereit, die der Forschung, Optimierung und Validierung der Algorithmen und für vergleichende Experimente dient. Diese werden auch für Demonstrationszwecke verwendet, um einen Demonstrator für Messen zu erstellen.

Durch die verstärkte Zuarbeit des Fraunhofer IGCV wurde zudem ein Datensatz erstellt, welches Bilder von Kohlenstofffaserverstärkten Kunststoffen (CFK) beinhaltet. Dieser besteht aus CFK-Gewebe mit diversen Fehlern und deren Annotationen.

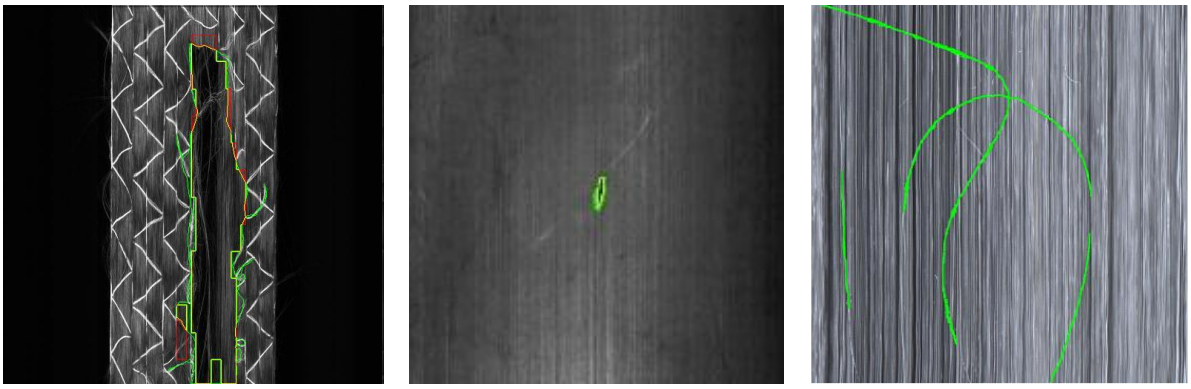

Abbildung 11: Beispielbilder vom fehlerhaften CFK inklusive Annotation

Mithilfe dieser annotierten Bilder konnte im Anschluss ein Model zum Segmentieren der Daten trainiert werden. Dafür wurde ein standardmäßiges U-Net verwendet. Da jedoch die Bilder sehr komplex sind, konnte das U-Net nur ungefähr 50% aller Pixel der Validierungsbilder korrekt segmentieren.

Der Zeitaufwand dieser Validierung ist von vielen Faktoren abhängig.

Die Annotation kann bei komplexen Bildern viel Zeit in Anspruch nehmen – im Beispiel von CFK sind es sehr komplexe Bilder mit nicht eindeutigen Fehlern. Ein Team aus Expert:innen müssen solche Bilder annotieren, wobei sie sich oft nicht einig über die Art des Fehlers sind. Durch solche Ungewissheiten kann die Annotation sehr lange dauern, sodass sich das Labeln von nur wenigen Bildern über mehrere Tage hinwegziehen kann.

Die Annotation der CampusCat Bilder benötigt wiederum sehr wenig Zeit. Für die Annotation muss nur das Areal der Plüschkatze markiert werden. Diese Arbeit lässt sich sehr einfach auf viele Personen aufteilen, da kein Fachwissen benötigt wird. So kann innerhalb einer Stunde – durch die Mithilfe von mehreren Personen – ein Datensatz mit über 100 Bildern entstehen.

Ein anderer Aspekt der Validierung ist das Training der Modelle. Auch hier spielen viele Faktoren hinein, da das Training und das Modell richtig konfiguriert werden müssen. Im Beispiel von neuronalen Netzen bedeutet das die Wahl von:

- Lernrate
- Optimierungsalgorithmus (bspw. Gradient Descend vs. AdamOptimizer vs. ...)
- Netzwerk-Architektur (Anzahl von Faltungsschichten und deren Konfiguration, Normalisierung der Batches oder Schichten, ...)
- ...

Die Konfiguration kann durch Optimierungsverfahren wie Optuna[12] bestimmt werden. Die Netzwerk-Architektur lässt sich auch durch CGP[13] optimieren. Beide Vorgehen benötigen allerdings viel Zeit und Hardware-Ressourcen. Durch Domänen- und Experten Wissen ist die Krux der richtigen Konfiguration eines NN nicht mehr vorhanden. Je nach Wissensstand kann auch eine Person das NN schnell erstellen und konfigurieren. Durch die Optimierung durch CGP

[12] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.
[13] Suganuma, M., Shirakawa, S., Nagao, T. (2020). Designing Convolutional Neural Network Architectures Using Cartesian Genetic Programming. In: Iba, H., Noman, N. (eds) Deep Neural Evolution. Natural Computing Series. Springer, Singapore. https://doi.org/10.1007/978-981-15-3685-4_7

kann die Suche nach einer geeigneten Architektur länger dauern – benötigt allerdings keinen Experten / keine Expertin, sodass Geld, Zeit und Ressourcen gespart werden.

1.9 AP 4.4.1 Überprüfung der Einhaltung der Validierungskennzahlen

Kritische Betrachtung der Übereinstimmung mit Validierungskennzahlen; Ursachenanalyse, Identifikation von weiterem Forschungspotential; Retro-Analyse der Validierungsphase, Überprüfung der Einhaltung von Anforderungen (vgl. AP 1.2, 1.3) im Hinblick auf Zielanwendungen;

- *Erstellen von Grafiken*
- *Datenanalyse*

Die meisten Validierungskennzahlen wurden schon in vorherigen AP kritisch betrachtet und besprochen. Zusammenfassend lässt sich für AP 4.4.1 sagen, dass beinahe alle gesetzten Validierungskennzahlen Übereinstimmen im Hinblick auf die Zielanwendung.

Eine genaue Auflistung wird hier nicht gegeben, um Redundanzen zu vermeiden. Wir verweisen außerdem auf die eigehende Darstellung der anderen Projektpartner (Fraunhofer IGCV; Universität Stuttgart) um einen Überblick über die hier nicht genannten Validierungskennzahlen zu bekommen.

1.10 AP 5.1 Publikation und Diskussion der Ergebnisse

Publikation auf Konferenzen; Erstellung von Artikeln für die Projektwebseite;

WEBSITE:

Seit 2022 besitzt SaMoA eine Website, die stetig aktualisiert wird.

https://www.uni-augsburg.de/de/fakultaet/fai/informatik/prof/oc/forschung/aktuelle-forschungsprojekte/samoa/

MESSEN:

Zu Beginn des Projektes wurde das Projekt schon am 15. Juli 2022 auf der KI-Summerschool 2022 in Augsburg vorgestellt. Diese Summerschool wurde von Bayern Innovativ, Bayerische Gesellschaft für Innovation und Wissenstransfer mbH, der fortiss GmbH, dem Landesforschungsinstitut des Freistaats Bayern für softwareintensive Systeme sowie dem KI-Produktionsnetzwerk Augsburg veranstaltet. Ziel dabei war es, interessierten Unternehmen einen Einblick in die KI sowie der aktuellen Forschung zu geben. Dabei wurde sich vor allem auf eine potenzielle Anwendung der KI in Unternehmen bezogen.

Hinsichtlich Konferenzen konnte SaMoA auf der FPGA Conference Europe in München am 04. Juli mit Personen aus der führenden Industrie eindrücke Sammeln. Durch Vorträge und Gespräche mit hochrangigen Wissenschaftlern und Personen aus der Industrie konnten neue Ideen generiert und der neueste Stand der Technik erschlossen werden.

Im Jahr 2023 war SaMoA noch auf zwei weiteren Messen. Zum einen konnten wir unseren Demonstrator bei der Halleneröffnung des Augsburger KI-Produktionsnetzwerkes vorstellen (https://www.uni-augsburg.de/de/forschung/einrichtungen/institute/ki-produktionsnetzwerk/).

Zur Halleneröffnung waren verschiedenste Vertreter aus der Industrie und Politik eingeladen. Zum einen wurden die Teilnehmenden dazu animiert, eine vereinfachte Variante des Questionnaires auszufüllen. Nachdem alle Angaben ausgefüllt wurden, konnte live mitverfolgt werden, wie eine neue Konfiguration für ein Monitoring System ausgegeben wurde. Zusätzlich veränderte sich der Design Graph vom DC43, sodass bestimmte Entscheidungen nachvollzogen werden konnten. Der zweite Teil des Demonstrators fokussierte sich auf die Bildverarbeitung. Hier konnten die Teilnehmenden live ein neues Modell trainieren und neue, unbekannte Objekte segmentieren lassen.

Der selbe Demonstrator wurde anschließend einige Wochen später auf der Automatica ausgestellt.

Auf beiden Messen war das Interesse an SaMoA sehr groß. Viele Vertreter:innen von (mittelständischen) Unternehmen waren von der Idee des automatisierten Monitoringsystems begeistert. Es kam zum regen Austausch mit vielen diversen Firmen, sodass Feedback sowie weiterführende Anwendungsfälle diskutiert werden konnten.

Im Jahr 2024 war SaMoA auf der Hannover Messe, der auch im Projektantrag angestrebten Industriemesse mit internationaler Bedeutung. Dort konnte der Demonstrator vor Fachpublikum ausgestellt werden, sowie Inhalte und Ziele mit diversen Personen aus der Industrie diskutiert werden.

Mentorentreffen:

SaMoA wurde allen Mentoren sowie dem Industriebeirat vorgestellt und rege diskutiert. Hierbei wurde auf Mentorenseite Prof. Dr. Gregor Schiele aus der der Universität Duisburg-Essen sowie Prof. Dr.-Ing. Markus Till aus der Hochschule Ravensburg-Weingarten getroffen. Aus dem Industriebeirat wurde sich mit Vertretern von Volavis GmbH und Chromasens GmbH getroffen. Gemeinsam kam es bei jedem Treffen zum regen Austausch.

Publikationen:

Im Rahmen von SaMoA wurden viele Publikationen veröffentlicht. Eine genaue Beschreibung sowie Diskussion würde den Rahmen dieses Dokumentes Sprengen. Deswegen werden nur alle Publikationen chronologisch aufgelistet:

1. Henning Cui, Andreas Margraf and Jörg Hähner. 2022. Refining mutation variants in Cartesian genetic programming. Lecture Notes in Computer Science 13627, 185-200. DOI: 10.1007/978-3-031-21094-5_14
2. Henning Cui, David Pätzel, Andreas Margraf and Jörg Hähner. 2023. Weighted mutation of connections to mitigate search space limitations in Cartesian Genetic Programming. In FOGA '23: Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, 30 August - 1 September 2023, Potsdam, Germany. Association for Computing Machinery, New York, NY, 50-60 DOI: 10.1145/3594805.3607130
3. Henning Cui, Andreas Margraf, Michael Heider and Jörg Hähner. 2023. Towards understanding crossover for Cartesian Genetic Programming. In Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, Joaquim Filipe (Eds.). Proceedings of the 15th International Joint Conference on Computational Intelligence - ECTA, November 13-15, 2023, in Rome, Italy. SciTePress, Setúbal, 308-314 DOI: 10.5220/0012231400003595

4. Andreas Margraf, Henning Cui, Simon Heimbach, Jörg Hähner, Steffen Geinitz and Stephan Rudolph. 2023. Model-driven optimisation of monitoring system configurations for batch production. In Francisco José Domínguez Mayo, Luís Ferreira Pires, Edwin Seidewitz (Eds.). Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering MODELSWARD 2023, February 19-21, 2023, in Lisbon, Portugal. SciTePress, Setúbal, 176-183 DOI: 10.5220/0011688900003402
5. Andreas Margraf, Henning Cui, Stefan Baumann and Jörg Hähner. 2023. Filter evolution using Cartesian genetic programming for time series anomaly detection. In Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, Joaquim Filipe (Eds.). Proceedings of the 15th International Joint Conference on Computational Intelligence - ECTA, November 13-15, 2023, in Rome, Italy. SciTePress, Setúbal, 300-307 DOI: 10.5220/0012210700003595
6. Andreas Margraf, Henning Cui, Anthony Stein and Jörg Hähner. 2023. Evolving processing pipelines for industrial imaging with Cartesian Genetic Programming. In Peter Lewis, Marin Litoiu, Ivana Dusparic, Barry Porter, Norha M. Villegas (Eds.). 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 25-29 September 2023, Toronto, ON, Canada. IEEE, Piscataway, NJ, 133-138 DOI: 10.1109/ACSOS58161.2023.00031
7. Henning Cui, Andreas Margraf and Jörg Hähner. 2023. Equidistant Reorder operator for Cartesian Genetic Programming. In Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, Joaquim Filipe (Eds.). Proceedings of the 15th International Joint Conference on Computational Intelligence - ECTA, November 13-15, 2023, in Rome, Italy. SciTePress, Setúbal, 64-74 DOI: 10.5220/0012174100003595
8. Henning Cui, Michael Heider and Jörg Hähner. 2024. Positional bias does not influence Cartesian Genetic Programming with crossover. In Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, Thomas Bäck (Eds.). Parallel Problem Solving from Nature – PPSN XVIII: 18th International Conference, PPSN 2024, Hagenberg, Austria, September 14–18, 2024, proceedings, part I. Springer, Cham, 151-167 DOI: 10.1007/978-3-031-70055-2_10
9. Henning Cui and Jörg Hähner. 2024. Cartesian Genetic Programming is robust against redundant attributes in datasets. In Francesco Marcelloni, Kurosh Madani, Niki van Stein, Joaquim Filipe (Eds.). Proceedings of the 16th International Joint Conference on Computational Intelligence - ECTA, November 20-22, 2024, in Porto, Portugal. SciTePress, Setúbal, 108-119 DOI: 10.5220/0012974600003837
10. Henning Cui, Andreas Margraf and Jörg Hähner. in press. Analysing the influence of reorder strategies for Cartesian Genetic Programming. preprint. DOI: 10.48550/arXiv.2410.00518

2 der wichtigsten Positionen des zahlenmäßigen Nachweises,

Besonders wichtig waren personelle Kosten sowie Reisekosten. Insbesondere durch die hohe Anzahl an Publikationen wurden viele Reisen zu Konferenzen getätigt.

Weiterhin ist erwähnenswert, dass ihm Rahmen von SaMoA ein Server beschafft wurde. Dieser wurde für das Training der Lehralgorithmen verwendet.

3 der Notwendigkeit und Angemessenheit der geleisteten Arbeit,

Auf verschiedenen Messen wurden mit vielen, diversen Vertretern der Industrie gesprochen. Alle haben dabei sehr großes Interesse an SaMoA gezeigt; und haben auch den Nutzen,

Notwendigkeit, sowie die Wirtschaftlichkeit angepriesen. Das selbe Feedback wurde uns auch durch die regelmäßigen Treffen mit dem Industriebeirat sowie den Mentoren vermittelt.

Das Innovationspotential ist bei SaMoA besonders groß, da es derzeit keine vergleichenden Produkte gibt.

Auch ist ein Wissens-transfer möglich. Durch Anpassungen der Software lassen sich Erkenntnisse und Ziele sehr leicht auf andere Use-Cases anpassen.

4 des voraussichtlichen Nutzens, insbesondere der Verwertbarkeit des Ergebnisses im Sinne des fortgeschriebenen Verwertungsplans,

Im Sinne der Verwertung werden viele Richtungen angestrebt.

Auf Seitens der Uni-A wird eine Ausgründung sowie eine EXIST-Förderung angestrebt. Weiterhin wird im Rahmen von SaMoA auch eine Dissertation geschrieben, und Publikationen veröffentlicht.

5 des während der Durchführung des Vorhabens dem Zuwendungsempfänger bekannt gewordenen Fortschritts auf dem Gebiet des Vorhabens bei anderen Stellen,

Uns, sowie dem Industriebeirat, sind keine Fortschritte bei anderen Stellen bekannt.

6 der erfolgten oder geplanten Veröffentlichungen des Ergebnisses nach Nr. 5 NKBF/NABF

Siehe AP 5.1