Universität Bremen

Projekt DeepAnatomy

Projektbericht

Teilnehmer:

Karol Baginski	kbaginsk@uni-bremen.de
Joana Becker	beckerj@uni-bremen.de
Daniel Bröcker	broecker@uni-bremen.de
Jan-Gerrit Göbel	jgoebel@uni-bremen.de
Philipp Haker	phaker@uni-bremen.de
Johann Joachim	keuneke@uni-bremen.de
Tom Köhler	tom_koe@uni-bremen.de
Thorben Lorenzen	tho_lor@uni-bremen.de
Max Meyer	max_mey1@uni-bremen.de
Clara Maria Odinius	odinius@uni-bremen.de
Markus Rink	marink@uni-bremen.de
Marvin Alexander Schäcke	schaecke@uni-bremen.de
Lukas Schäfer	luksch@uni-bremen.de
Artem Tscherwinski	arts@uni-bremen.de
Nina Unterberg	nin_unt@uni-bremen.de

Betreut durch:

Hans Meine	Hans.Meine@mevis.fraunhofer.de
Felix Thielke	felix.thielke@mevis.fraunhofer.de

AG Medical Image Computing

Universität Bremen

WiSe19/20 & SoSe20

Inhaltsverzeichnis

1 Einleitung

Kapitel Autor: Joana Becker

Der folgende Projektbericht ist in dem Bachelorprojekt `Deep Anatomy` im Wintersemeter 2019/20 und Sommersemester 2020 entstanden. Er enthält Informationen zur Entstehung, der Organisation und den Zielen des Projekts. Die einzelnen gebildeten Teams beschreiben ihre Visionen, Arbeit und Ergebnisse. Darauf aufbauend wird beschrieben, ob die gesetzten Ziele erreicht wurden und wie ein folgendes Bachelor- oder Masterprojekt darauf aufbauen könnte.

Die Projektgruppe setzt sich aus 15 Studierenden des Bachelorstudiengang Informatik der Universität Bremen zusammen und wird hauptsächlich von Hans Meine und Felix Thielke betreut.

1.1 Geschichte

Autor: Joana Becker nach Informationen von Hans Meine

Das Bachelorprojekt `Deep Anatomy` ist aus der Zusammenarbeit zwischen Mitarbeitern des Fraunhofer MEVIS und Universitäten wie der Universität Bremen entstanden. An der Uni Bremen existiert um den Professor Ron Kikinis die Arbeitsgruppe Medical Image Computing (siehe `http://www.mic.uni-bremen.de/`). Im Zusammenhang mit der Arbeitsgruppe und dem Konzept der Lehre kam der Gedanke auf, dass ein Bachelor- und/oder Masterprojekt dazugehören würde. Der erste Versuch, ein Masterprojekt zum Thema Knie-Knorpel-MRT-Analyse stellte sich als zu spezifisch heraus, ebenso mangelte es an Interessenten, weil viele Studenten direkt aus

ihrem Bachelorprojekt in das fortführende Masterprojekt wechselten, was in diesem Fall nicht möglich war.

Der folgende Vorschlag von Horst Hahn, die Arbeit an der Online-Plattform im Fraunhofer MEVIS mit den Studenten im Projekt zu teilen, war zwar eine spannende, jedoch sehr große Vision, die zu einem vermittelbaren Projekt bearbeitet werden musste. Für die Vision der vorzeigbaren Online-Plattform im Bachelorprojekt hatte sich Hahn von dem Bachelorprojekt rund um den Roboterfußball inspirieren lassen. Dort gibt es überregionale Wettbewerbe und eine ordentliche Internetpräsenz. Diese Merkmale wurden im folgenden Bachelorprojekt umgesetzt. Dazu passend ist die Übernahme der Betreuerrolle durch Felix Thielke, der selbst Erfahrungen im Roboterfußball sammeln konnte. Unterstützt wird das Bachelorprojekt vonseiten der Uni Bremen und dem Fraunhofer MEVIS. Hans Meine beispielsweise besitzt neben seiner Stelle bei MEVIS eine halbe Stelle an der Uni Bremen im Fachbereich 3. Das und die enge Vernetzung zu weiteren Universitäten wie der Jacobs-Universität ermöglichen Erfahrung im Umgang mit Studentenprojekten und den dazugehörigen Aufgaben, sowie Wissen über Industrieprojekte und Projekte am Fraunhofer MEVIS.

1.2 Fraunhofer MEVIS

Die Fraunhofer Gesellschaft zeichnet sich dabei als führende Forschungsorganisation in anwendungsorientierter Forschung in ganz Europa aus. Gegenüber anderen Forschungsorganisationen hat sie dadurch die geringste Grundfinanzierungsquote, erhält nur ein Drittel des Geldes von Bund und Ländern, ein Drittel aus öffentlichen Fördermitteln und ein Drittel aus der Industrie. Diese Aufteilung wird angestrebt, um der angewandten Forschung gerecht werden zu können. Das ermöglicht qualitätsgesicherte Entwicklung für beispielsweise Medizinproduktkomponenten, weitere Forschung und überbrückt zugleich das *Valley of Death* zwischen Grundlagenforschung und der Anwendung in der Gesellschaft.

Das Fraunhofer MEVIS hat sich dabei als Teil der Fraunhofer Gesellschaft auf bild- und datengestützte Softwaresysteme zur Früherkennung, Diagnose

und Therapie im medizinischen Bereich spezialisiert, sowie auf die Auswertung bildbasierter Studien zur Wirksamkeit von Medikamenten und Kontrastmitteln. Dafür findet eine enge Zusammenarbeit mit klinischen und akademischen Partnern statt [vgl. https://www.mevis.fraunhofer.de/, letzter Zugriff am: 26.05.20].

2 Projektziel

Autor: Johann Keuneke

2.1 Anfängliche Ziele

Im Bachelorprojekt Deep Anatomy haben sich die Mitglieder des Projektes in 3 Gruppen aufgeteilt, welche an verschiedenen Aufgaben im Bereich der medizinischen Bildverarbeitung arbeiteten. Hierbei hatte sich jedes Team ein Ziel vorgenommen, dass im Verlaufe des Bachelorprojekts erreicht werden sollte. Dabei entstanden folgende Teams: Trainings Loop, Front-/Backend und Challenges.

2.1.1 Team-Trainings Loop

Das Team Trainings Loop befasste sich mit der Optimierung der Trainingsschleifen. Hierbei sollte ein Modul entwickelt werden was dabei hilft, die Unterschiede zwischen originalen und trainierten Daten, zu visualisieren und zu analysieren.

2.1.2 Team-Front-/Backend

Das Team Front-/Backend beschäftigte sich mit einer Visualisierungsmöglichkeiten der DICOM Tags. Hierzu sollte in Dashboard erstellt werden, das über eine Weboberfläche erreicht werden kann. Dort sollen die Informationen der DICOM Daten Visualisiert werden.

2.1.3 Team-Challenges

Das Team Challenges hat sich mit Projekt externen Challenges befasst, welche im Themenbereich der medizinischen Bildverarbeitung lag. Das Ziel des Team-Challenges bestand darin, die vorgegebene Aufgaben der gewählten Challenge zu bearbeiten und ein Resultat zu liefern welches die Anforderung so gut es ging erfüllte.

2.1.4 Projekttag

Der Projekttag war eine geplante Veranstaltung, welcher dazu diente die erarbeiteten Ergebnisse der Teams vorzustellen. Jedoch ist der Projekttag der Uni-Bremen aufgrund des COVID-19 Virus ausgefallen, weswegen eine andere Darstellungsmöglichkeit für die Resultate der Projektgruppen gewählt werden musste.

2.2 Endgültiges Projektziel

Als Kompromiss für den Ausfall des Projekttags hat sich die Projektgruppe dafür entschieden die Resultate der Gruppen auf einer Homepage zu vereinen und zu veröffentlichen. Diese Homepage dient als Vorstellung und Informationsquelle der geleisteten Arbeit für das Bachelorprojekt Deep Anatomy und ist über diesen Link erreichbar: `http://www.deepanatomy.de`.

3 Organisation im Projekt

Autor: Nina Unterberg

Das Projekt Deep Anatomy wurde mit 15 Teilnehmern geführt. Diese wurden von Hans Meine und Felix Thielke als Betreuer unterstützt.
Das Bachelorprojekt ist eine zweisemestrige Veranstaltung der Universität Bremen. Es begann offiziell mit dem Wintersemester 19/20. Projekttreffen fanden allerdings schon ab Anfang September statt. Das Projekt endet im Mai 2020.
In diesem Zeitraum haben wöchentlich gemeinsame Plenen stattgefunden. Diese wurden teilweise in den Räumen des Fraunhofer Mevis Institutes abgehalten, teilweise in dem, von der Uni zur Verfügung gestellten, Projektraum im Cartesium und über Zoom.
In den Plenen wurden aktuelle Themen besprochen und regelmäßige Updates zu den Aufgaben der Teams und Einzelmitglieder gegeben. Des weiteren wurden aufkommende Fragen beantwortet und diskutiert.

3.1 Aufteilung in Teams

Autor: Clara Odinius

Zunächst wurde die gesamte Projektgruppe mit Hilfe von Hans Meine und Felix Thielke in die Thematik der medizinischen Bildverarbeitung und Deep Learning eingeführt. In diesem Zusammenhang wurden relevante Technologien vorgestellt und erklärt, um jeden Studenten auf den gleichen Stand zu bringen. Mit diesem Wissen sind eigene Ideen über mögliche Projektziele entstanden. In einer Runde konnte jeder Student darüber reden, welches Thema sein Interesse am meisten geweckt hat und an welcher Stelle er am Projekt mitwirken möchte. Im Anschluss daran, hat sich herausgestellt, dass

sich die Projektgruppe in drei Teams einteilen kann. Von da an wurden auf die jeweiligen Projektziele in den Teams hingearbeitet. An den wöchentlichen Plenen, haben sich die Teams gegenseitig über den aktuellen Stand informiert. Parrallel dazu wurde gemeinsam darüber gespochen, wie sich die Projektziele am Ende vereinen könnten.

3.2 Kommunikationskanäle

Autor: Thorben Lorenzen

Um die Kommunikation so einfach wie möglich zu gestalten, wurden verschiedene Plattformen und Tools eingesetzt.
So wurde Mattermost als Instant-Messengerdienst genutzt. Hier wurden teaminterne Nachrichten, Nachrichten zwischen einzelnen Personen, sowie Nachrichten im großen Plenum ausgetauscht.
Als Wiki-Plattform wurde das interne Confluence von Fraunhofer benutzt. In Confluence wurden Team-Meetings und organisatorische Themen, wie zum Beispiel Abstimmungen oder der Kassenbestand dokumentiert. Auch galt es als Plattform für allgemeinen Wissensaustausch, auf der Projektmitglieder eine gut dokumentierte Confluence-Seite über ihr projektspezifisches angeeignetes Wissen veröffentlichen und so mit anderen Projektteilnehmern teilen konnten. Des Weiteren wurde Confluence zum Pflegen der Diaries benutzt, in welchen jeder seine in das Projekt investierte Zeit dokumentieren sollte. Zum Verteilen und Zuordnen von teaminternen Aufgaben wurde die Plattform Jira benutzt. Dieses Tool bot eine gute Planungsbasis für kleinere und größere Projekte, besonders bei den Teilnehmern, die in Teams programmiert haben.
Für die in den letzten Monaten des Projekts online geführten Plenen, wurde das Tool Zoom verwendet. Zoom ist ein Video-Kommunikationstool. Hier wurden die Plenen als wöchentliche Video-Konferenzen durchgeführt.

3.3 Technologien und Infrastruktur

Autor: Thorben Lorenzen

Von Fraunhofer wurden einige Technologien und eine Infrastruktur zur Verfügung gestellt. So wurde eine Lizenz für MeVisLab und das Framework RedLeaf (**Re**mote **D**eep **Lea**rning **F**ramework) für jeden Projektteilnehmer bereitgestellt.
Außerdem wurden die CNNs nicht lokal trainiert, sondern hauptsächlich über einen Cluster, dass zur Fraunhofer-Infrastruktur gehört. Durch den leistungsstarken Cluster, konnten die CNNs deutlich schneller trainiert werden.
Auch ein Datei-Server wurde uns zur Verfügung gestellt. Um CNNs auf dem Cluster trainieren zu können, mussten die entsprechenden Konfigurations-Dateien und Datensätze auf dem Datei-Server „Gaia“ liegen.
Für das Annotieren von ungelabelten Bild-Daten wurde „Satori“ benutzt.

3.4 Teambuilding

Autor: Artem Tscherwinski

Als Teambuilding-Maßnahme wurde in einer Umfrage entschieden, sich an einem Abend zum Bowling zu verabreden. Es wurden interessante Gespräche geführt und so kamen sich die Studenten untereinander und auch den Betreuern näher. Durch den Bowlingabend wurde das Fundament für eine effektive Zusammenarbeit geschaffen.

4 Projektmanagement

Kapitel Autor: Joana Becker

4.1 Einleitung

Um ein Projekt dieser Größe abschließen zu können, wird eine gut strukturierte Planung von Zeit, Aufgaben und Ressourcen benötigt. Für das Projektmanagement wurde eine kleine Gruppe gebildet, dessen Mitglieder zwar hauptsächlich Mitglieder in anderen Gruppen waren, sich aber gelegentlich für die Projektplanung trafen. Im Verlauf dieser Treffen ist ein Zeitplan entstanden und es wurde eine organisatorische Übersicht über die einzelnen Teilgruppen grafisch erstellt. Das Ziel dieser Gruppe war das Management der zur Verfügung stehenden Zeit und der einzelnen Teilgruppen. Da sich im Verlauf der Gruppenbildung jedoch herausgestellt hat, dass eine Zusammenarbeit der einzelnen Teilgruppen nicht von komplizierteren Abhängigkeiten bestimmt wurde, wurde das Management der Gruppen jeweils den einzelnen Teilgruppen selbst überlassen.

4.2 Übersicht

Aufgabe	Mitglieder	Beschreibung
Außenminister	Lukas Schäfer	Kommunikation mit den Außenministern der anderen Projekte zur Organisation des Projekttags.
Moderation des Plenums	Artem Tscherwinski, Jan-Gerrit Göbel	Führen des Projektteams durch die wöchentlichen Plenen.
Projektplanung	Karol Baginski, Joana Becker, Marvin Alexander Schäcke	Erstellung eines Projektstrukturplans, eines Zeitplans und Meilensteinen.

4.3 Projektplanung

Für die Planung wurden mehrere Leitfragen entwickelt und auf das Projekt angewandt. Diese Fragen beinhalten Motivation, Ziel, Zweck und Beschreibung der Aufgaben und Organisation. Nachdem die Ziele und Aufgaben formuliert wurden und eine Aufteilung der Aufgaben auf die Teilgruppen stattfand, wurde ein Projektstrukturplan entwickelt. Dieser stellt grafisch die Aufteilung der Projektgruppe in die einzelnen Untergruppen und deren mögliche Aufgaben dar. Für eine bessere Lesbarkeit ist der Strukturplan als Aufzählung dargestellt und nicht als Grafik.

Deep Anatomy:

- Plattform Entwicklung
  - User Experience - Optimierung des Datenimports
    - Import von Daten mit und ohne Annotation
    - browserbasiert/komfortabel
    - Integration DICOM-Anonymisierung
    - Externe Nutzbarmachung
  - Optimierung Trainingsschleife

        * Visualisierung von Unsicherheiten
        * Visualisierung von Unterschieden zwischen Annotation und Output
        * Confusion Matrix
    - Arichtekturoptimierung
        * nnU-Net
- Challenges
    - StructSeg
    - CHAOS
        * Abdominale Organe
        * Leber (CT)
        * Leber (MRI)
    - xVertSeg
- Projektmanagement
    - Projektstrukturplan
    - Zeitplan
    - Risikoanalyse
        * Allgemeine Risiken
        * Teamspezifische Risiken
            · UserExperience/Optimierung
            · Challenges
            · Optimierung Trainignsschleife
            · Architekturoptimierung

4.4 Zeitplan

Um einen Überblick über die Deadlines zu behalten, wurden mehrere Zeitpunkte gesetzt, die als Meilensteine dienen sollen. Zu diesen Zeitpunkten sollen einzelne Aufgaben erledigt werden oder Besprechungen über die Benotungen der Projektteilnehmer erfolgen. Der Zeitplan und der Projektstrukturplan wurden über Confluence veröffentlicht und so der gesamten Projektgruppe zugänglich gemacht.

5 Team Challenges

5.1 Einleitung

Autor: Joana Becker

In der Aufteilung des Projektteams in die einzelnen Arbeitsgruppen, hat sich eine Gruppe das Ziel gesetzt, an bestehenden Segmentation Challenges teilzunehmen. Solche Challenges sind öffentliche Wettbewerbe, bei denen annotierte und nicht-annotierte Bilder mit einer Aufgabe veröffentlicht werden. Das Ziel ist es, Netze für die Segmentierung von Elementen wie z.B. Organen zu entwickeln und auf den annotierten Bildern zu trainieren, um diese möglichst erfolgreich auf den nicht-annotierten Bildern anzuwenden. Die Ergebnisse können dann eingesendet und mit anderen Teilnehmern verglichen werden.

Das Challenge Team hat sich in weitere Gruppen aufgeteilt, die jeweils verschiedene Segmentation Challenges bearbeitet haben. Die einzelnen Challenges waren CHAOS, StructSeg und xVertSeg und behandeln jeweils unterschiedliche Bereiche im Körper. StructSeg und xVertSeg wurden in Einzel- und Partnerarbeit bearbeitet, die CHAOS-Gruppe bestand aus 4 Mitgliedern.

Die Mitglieder des Team Challenge waren Max Meyer, Joana Becker, Artem Tscherwinski, Daniel Bröcker, Johann Joachim Keuneke, Thorben Lorenzen und Tom Köhler.

5.2 Trainings-Pipeline

Autor: Artem Tscherwinski

5.2.1 Einleitung

Im Folgenden wird eine relativ simple Datenpipeline erläutert, welche für den Import von Leber-CT-Daten aus der CHAOS-Challenge verwendet wurde.

5.2.2 Der Datenimport

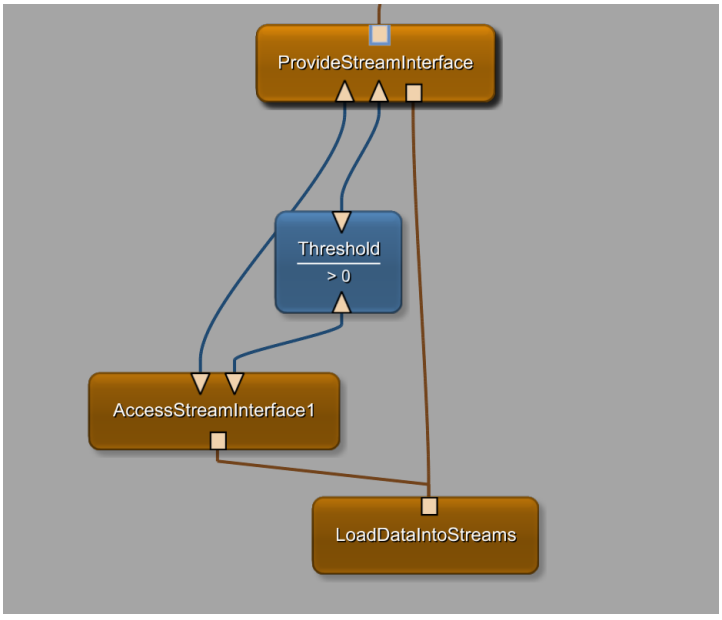


Abbildung 5.1: (Datenimport)

Die in Abb. 5.1 gezeigten Module dienen dem Datenimport. Während im `LoadDataIntoStreams`-Modul die Daten aus den Datenreferenzen einer .lst-Datei in einen Stream gewandelt werden, sorgen `AccessStreamInterface` und `ProvideStreamInterface` für die Zugänglichkeit der Daten, es lassen sich so verschiedene Daten auswählen. Das `Threshold`-Modul gibt allen Grauwerten, die größer als 0 sind, den Wert 1.

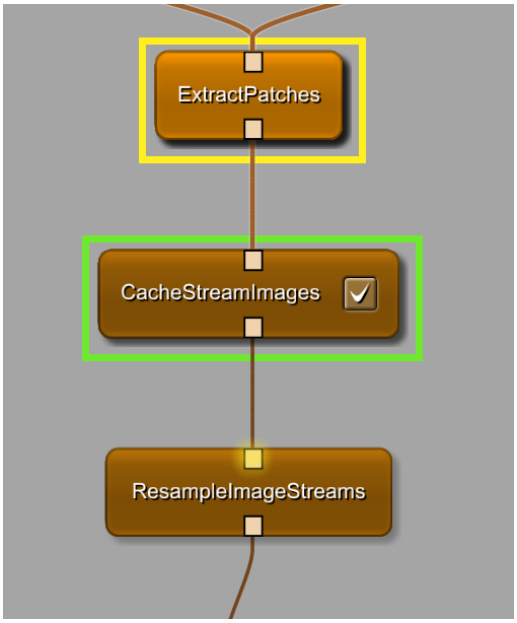


Abbildung 5.2: (Größenparameter)

5.2.3 Die Größenparameter

In Abb. 5.2 werden verschieden Größenparameter gesetzt, darunter die Patch-Size. Ein Patch ist ein Teil der aktuell betrachteten Schicht. Auch die Voxel-Size wird hier gesetzt, Voxel sind Bildpunkte. Das `CacheImageStreams`-Modul speichert die geladenen Slices im Cache, damit schneller darauf zugegriffen werden kann.

5.2.4 Randomisierung

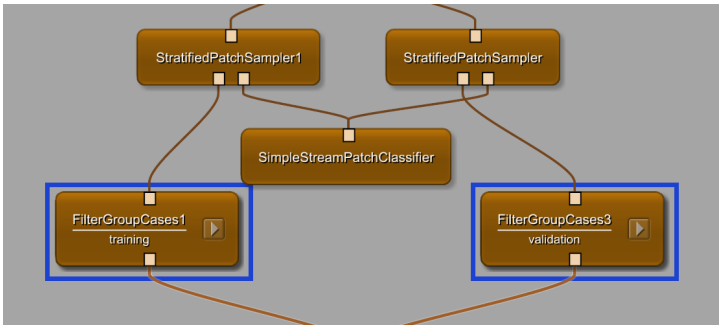


Abbildung 5.3: (Randomisierung)

In Abb. 5.3 sind viele Module miteinander verknüpft. Im Grunde werden hier Trainings- und Validierungsdaten getrennt und randomisiert.

5.2.5 Server

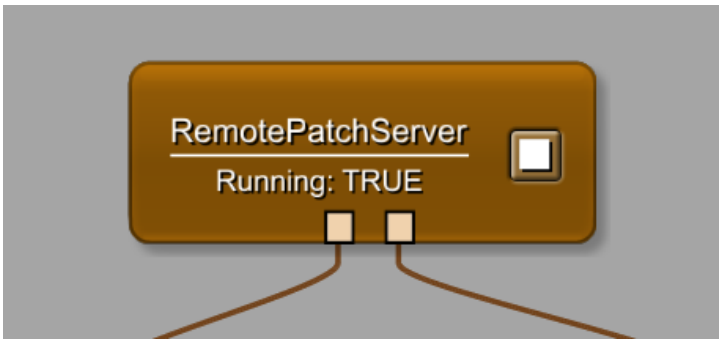


Abbildung 5.4: (Server)

In Abb. 5.4 werden die Daten mitsamt Einstellungen in den Server geladen.

5.2.6 Gesamtbild

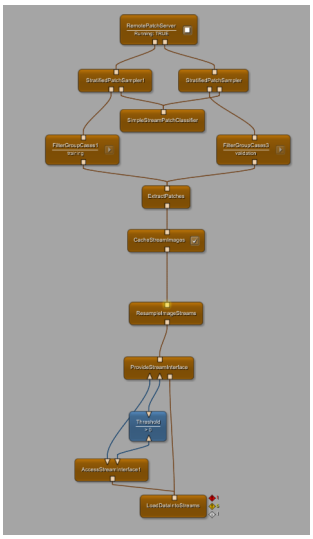


Abbildung 5.5: (Pipeline)

In Abb.5.5 ist die Pipeline im Ganzen zu sehen. Alle Konfigurationen, die hier angegeben werden, lassen sich auch in einer Config-Datei machen, ohne, dass Module dafür benutzt werden müssen.

5.3 Chaos Leber CT, Abdomen CT und MRT

Autor: Artem Tscherwinski

5.3.1 Einleitung

Es wurde sich mit der CHAOS-Challenge befasst, genauer mit der Segmentierung der Leber bei CT-Scans und später dann auch mit der Segmentierung von Leber, Nieren sowie Milz bei CT- und MRT-Scans.

5.3.2 Motivation

Interessant an dieser Challenge war, dass gleich mehrere Organe aus zwei Scan-Methoden gleichzeitig segmentiert werden sollen. Aufgrund von fehlendem Fachwissen war die grundsätzliche Idee, ein Training durchzuführen und dann anhand von Veränderungen des U-Net-Levels, der PatchSizes oder auch der Anzahl der Trainingsiterationen ein Bild davon zu machen, welche Änderungen zu welchen Ergebnissen führen, um so für künftige Trainings zu wissen, welche Parameter besondere Relevanz für die Qualität des Ergebnisses haben.
So wurde für den Beginn eine relativ simple Unterchallenge von CHAOS ausgewählt: Die Segmentierung einer Leber an CT-Scans von 20 Patienten. Ein Datensatz beinhaltet annotierte Scans mit Markierungen des gewünschten Organs, mit welchen trainiert wird. Ein weiterer Datensatz von 20 Patienten beinhaltet die nicht annotierten Datensätze, auf welche das Ergebnis angewandt werden kann.

5.3.3 Implementierung

Leber (CT): Für das erste Training wurden *default*-Werte für Voxel- und PatchSize verwendet. Am Ergebnis wurde die Problematik von kleinen Patches bei großen Organen deutlich: Bei einem so großen Organ wie der Leber, welches je nach betrachteter Slice zwei Drittel des Bauchraums einnehmen kann, werden gleichfarbige, aber deutlich kleinere Strukturen im Bauchraum fälschlicherweise ebenfalls als Leber markiert. In der Hoffnung, dass das Convolutional Neural Network erkennt, dass die Leber ein großes und zusammenhängendes Organ ist, wurden die PatchSizes erhöht. Leider war das Ergebnis wieder nicht zufriedenstellend, da nun feine Leberstrukturen sehr grobe Markierungen hatten oder einfach nicht erkannt wurden.
Der nächste Schritt war, die Anzahl der Trainingsiterationen bis zur Validierung zu vergrößern. So wurde das neuronale Netz gezwungen länger zu trainieren bis validiert wurde (und somit möglicherweise beendet), in der Hoffnung, dass nun feinere Strukturen erkannt werden. Auch wurden die Validierungsalgorithmen geändert und ebenfalls die Endbedingungen für das Beenden eines Trainings wurden angepasst. Leider führte die Veränderung der Iterationsanzahl nicht zum gewünschten Ergebnis, zeigte jedoch leichte Verbesserungen. Da nun die Grundzüge eines Trainingsablaufs nachvollzogen werden konnten, sollte eine interessanteren Aufgabe bearbeitet werden: Das Trainieren eines Netzes, welches mehrere Organe erkennt. Dafür wurde ein Datensatz von CT- und MRT-Scans aus der Chaos-Challenge verwendet.

Abdomen (MRT und CT): Nun wurden mehrere Organe gleichzeitig segmentiert, daher war eine Vorbereitung der Daten notwendig. Jedes Organ hat einen eigenen Grauwert zugewiesen bekommen, damit das CNN diese besser unterscheiden kann. Wegen teils überlappender Grauwerde in CT- und MRT-Datensätzen musste das Laden dieser angepasst werden. So mussten erst beide einzeln verarbeitet werden, bevor die Daten zusammengeführt werden konnten. Mit jedem Versuch wurde klarer, dass Daten von 20 Patienten nicht genug sind, um etwas so Individuelles, wie den menschlichen Körper, segmentieren zu können. Aspekte wie Körpergröße, Fett- oder

Muskelanteil und Geschlecht spielen eine zu große Rolle, als dass man mit 20 Patienten einen repräsentatives Ergebnis für das Trainieren eines neuronalen Netzes erreichen kann. Auch ein Postprocessing der Ergebnisse, wie das Filtern von zu kleinen Strukturen (also nur noch Beachtung von zusammenhängend-segmentierten Strukturen im Körper) zeigten nur leichte Verbesserungen. Einige Strukturen wurden schlicht nicht erkannt, dies könnte viele Gründe haben, sicher hat auch die Qualität der Datensätze, die besonders bei den MRT-Daten zu wünschen übrig ließ, eine nicht unwichtige Rolle gespielt.

5.3.4 Fazit

Das Trainieren eines neuronalen Netzes ist eine Herausforderung, die großes Fachwissen benötigt, um nicht einfach zufällige Parameter zu verändern, bis etwas passiert. Je nach Aufgabe wird eine große Menge an qualitativ hochwertigen Daten benötigt, sei es die Auflösung oder die Vor-Segmentierung, schließlich soll nur mit *korrekter* Segmentierung trainiert werden, um so die Qualität des Ergebnisses so hoch wie möglich zu halten. Ebenfalls ist eine strukturelle Herangehensweise von großer Wichtigkeit. Ein Training durchzuführen kann sehr lange dauern, umso wichtiger ist es, sich im Vorfeld Gedanken gemacht zu haben, was erreicht werden soll. Natürlich muss gleichzeitig immer der nächste Schritt bedacht werden, um eine systematische Vorgehensweise aufrecht zu halten. Leider war keins der Ergebnisse zufriedenstellend, sicher wegen fehlenden Erfahrung, aber auch aufgrund der nicht so guten Bedingungen bezüglich Qualität und Quantität der Datensätze der CHAOS-Challenge.
Aus persönlicher Sicht war das Bachelorprojekt sehr bereichernd, einerseits durch den Einblick in eine andere Thematik, andererseits war es angenehm zu sehen, wie gut die eigenen Ideen funktionieren (oder auch nicht funktionieren) können.
Die Informationsbeschaffung war auf Grund des großen Wissenskatalogs von Fraunhofer MeVis nicht immer einfach und auch wenn die gewünschte Information gefunden wurde, musste vieles nachgeschlagen werden, da das Fachwissen nicht in dieser Tiefe vorhanden war.

5.4 Chaos

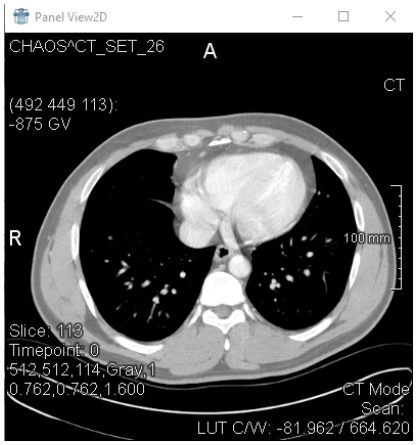
Autor: Thorben Lorenzen, Max Meyer

5.4.1 Einleitung

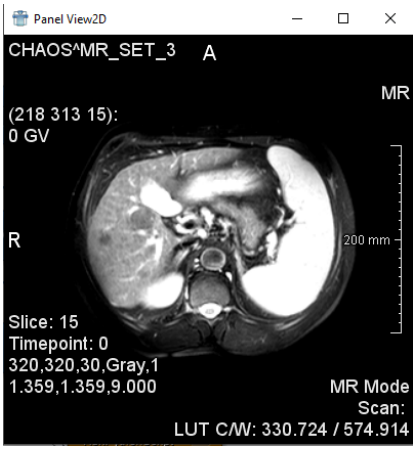
CHAOS ist ein Akronym für Combined Healthy Abdominal Organ Segmentation. Die Challenge befasst sich also mit der Erfassung von abdominalen Organen.
Das Verständnis von diesen ist von großer Wichtigkeit für den Erfolg von medizinischen Eingriffen. Um mehr über diese zu lernen, helfen Werkzeuge zur Visualisierung. Dies Erfordert jedoch die Extraktion der Objekte von DICOM Bildern (Digital Imaging and Communications in Medicine). Eine präzise Segmentierung hat große Wichtigkeit für mehrere klinische Eingriffe, wie zum Beispiel die Vorbewertung der Leber für eine Lebertransplantation.

5.4.2 Daten

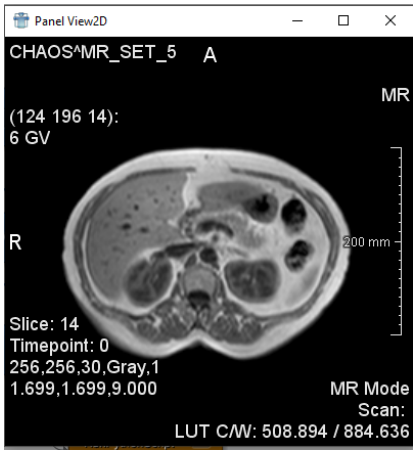
Die Bild-Daten der Chaos-Challenges bestehen aus zwei Datensätzen. Der CT-Datensatz wurde insgesamt von 40 verschiedenen Patienten aufgenommen, wovon 20 mit einer entsprechenden Maske vorliegen. Im
MRT-Datensatz liegen insgesamt 120 aufgenommene Patientendaten vor. Dabei liegen 40 Daten vor, die mit der SPIR (Spectral Pre-Saturation Inversion Recovery) Methode Erklärung oder Glossar und dem T2 Kontrast Mechanismus aufgenommen wurden und jeweils 40 Daten aufgenommen in unterschiedlichen Phasen(In-Phase und Out-Phase) mit dem T1 Kontrast Mechanismus. Insgesamt wurde dieser MRT-Datensatz also mit unterschiedlichen radiofrequentierten Impulsen und unterschiedlichen Kontrastmitteln aufgenommen, welche zur unterschiedlichen Erkennbarkeit der Organe beitragen wie in den Abbildungen zu sehen ist. So werden zum Beispiel bei dem SPIR-Datensatz Fettprotonen unterdrückt, was zur klareren Abgrenzung der Organe vom umliegenden Gewebe führt. Der Inphase Datensatz zeigt Aufnahmen, wenn die Fett- und Wasserprotonen in Phase sind und der Outphase Datensatz, wenn diese Protonen entsprechend außerhalb der



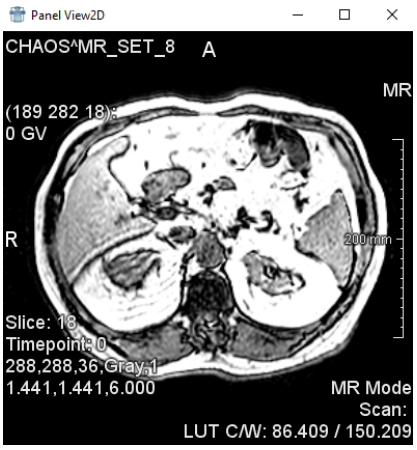
(a) CT Beispiel



(b) T2Spir Beispiel



(c) T1Dual-Inphase Beispiel



(d) T1Dual-Outphase Beispiel

Phase sind. Auch zu erwähnen ist, dass keiner der Patienten kranke Organe besaß.

5.4.3 Datenimport

Da die MRT/CT Chaos-Challenge-Daten im Dateiformat DICOM und die Masken dazu im Dateiformat PNG vorlagen, mussten sie noch in das Dateiformat MLIMAGE umgewandelt werden, das in Mevis-Lab einlesbar ist. Dazu wurden entsprechende Skripte mit MeVisLab programmiert. Wie für den MRT Datenimport in Abbildung 5.7 zu sehen ist.

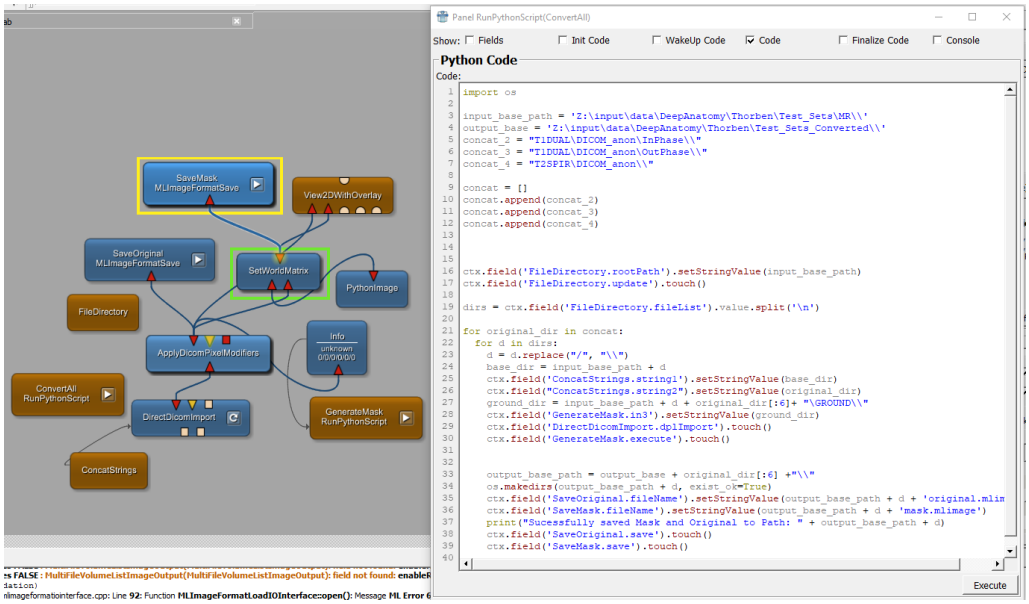


Abbildung 5.7: Mevis-Lab Modul-Netz für die Datenimportierung

5.5 Chaos - Abdominale Organ-Segmentierung in MRI/MRT

Autor: Thorben Lorenzen

5.5.1 Einleitung

Innerhalb des Team Challenges erfolgte eine Aufteilung auf verschiedene Challenges. So hatte sich eine Gruppe von fünf Personen zusammengefunden, die die Challenges von einem Anbieter (Chaos) bearbeiteten. Dies hatte einige Vorteile, so musste zum Beispiel der Datenimport nur einmal vorgenommen werden. Die Chaos-Challenges bestanden aus der Segmentierung von Leber und abdominalen Organen, jeweils auf MRT und CT Daten.

Die Challenge in diesem Kapitel handelt von der abdominalen Organ-Segmentierung auf Basis von MRT Daten. Hier bestand die Aufgabe darin, die Leber, die Nieren und die Milz aus gegebenen MRT Daten zu segmentieren. Es handelt sich hierbei also um ein Multi-Segmentierungs-Problem.

5.5.2 Motivation

Die Challenge, die in diesem Kapitel behandelt wird, gliedert sich in zwei Aufgaben auf und darf mit zwei unterschiedlichen CNNs gelöst werden. Hierbei sei erwähnt, dass sich die Implementierung ausschließlich mit Aufgabe 1 befasst.

1. Aufgabe: Ein CNN basierend auf dem T1Dual-Datensatz (MRT)

2. Aufgabe: Ein CNN basierend auf dem T2Spir-Datensatz (MRT)

Da der T1Dual-Datensatz aus unterschiedlichen Aufnahmen (In- und Outphase) besteht, die zwar dieselbe räumliche Struktur aufweisen, jedoch unterschiedliche Phasen der Fett und Wasser-Protonen zeigen, besteht die Motivation der 1. Aufgabe darin, ein generalisiertes CNN zu bauen, das auf diesen beiden unterschiedlichen Aufnahmen gute Ergebnisse in Form einer guten Erkennung der zu segmentierenden Organe erzielen kann. Diese Art

der Aufgabenstellung gewinnt auch zunehmend an Bedeutung im medizinischen Bereich, da oft unterschiedliche Arten von Aufnahmen vorliegen, die dieselbe räumliche Struktur und somit die gleichen Organe zeigen. Ein großes Problem in Machine Learning und insbesondere in Deep Learning ist, dass sehr große Datenmengen benötigt werden. Dem wird damit entgegen gewirkt, dass dem Datensatz, auf dem ein CNN trainiert, Daten hinzugefügt werden, die zwar das Gleiche zeigen, aber anders in der Darstellung sind. Dadurch stehen insgesamt mehr Daten zu einer gegebenen Fragestellung (beispielsweise der Segmentierung von Organen) zur Verfügung.
Für die 2. Aufgabe der Challenge steht ein Datensatz mit einheitlich aufgenommenen Daten zur Verfügung.
In beiden Aufgaben besteht das Ziel darin, die Parameter für das resultierende CNN zu optimieren, sodass möglichst gut die verschiedenen Organe segmentiert werden. Hier ist auch eine praktische Anwendung in der Medizin gegeben. Unter anderem, um zu evaluieren, ob ein medizinischer Eingriff durchgeführt werden kann, oder nicht.

5.5.3 Implementierung

Für das Trainieren der CNNs, wurde (bis auf kleine Veränderungen) die Standard-Pipeline benutzt Abbildung 5.5
Das `Threshold`-Modul wurde entfernt, da dieses Modul alle Label, die ungleich 0 sind auf 1 setzt. Dies ist aber nur bei binären Segmentierungsproblemen hilfreich, was hier aber nicht vorliegt.
Des Weiteren wurde diese Pipeline zur Bearbeitung der 1. Aufgabe wie in Abbildung 5.8 verändert. Hier wurde das Modul `ConcatenateImages` hinzugefügt. Da der T1Dual-Datensatz zu jeder Aufnahme zwei Bilder mit derselben räumlichen Struktur besitzt, werden diese übereinander gelegt, sodass jeder Patch, mit dem das CNN trainiert wird, zweidimensional ist.

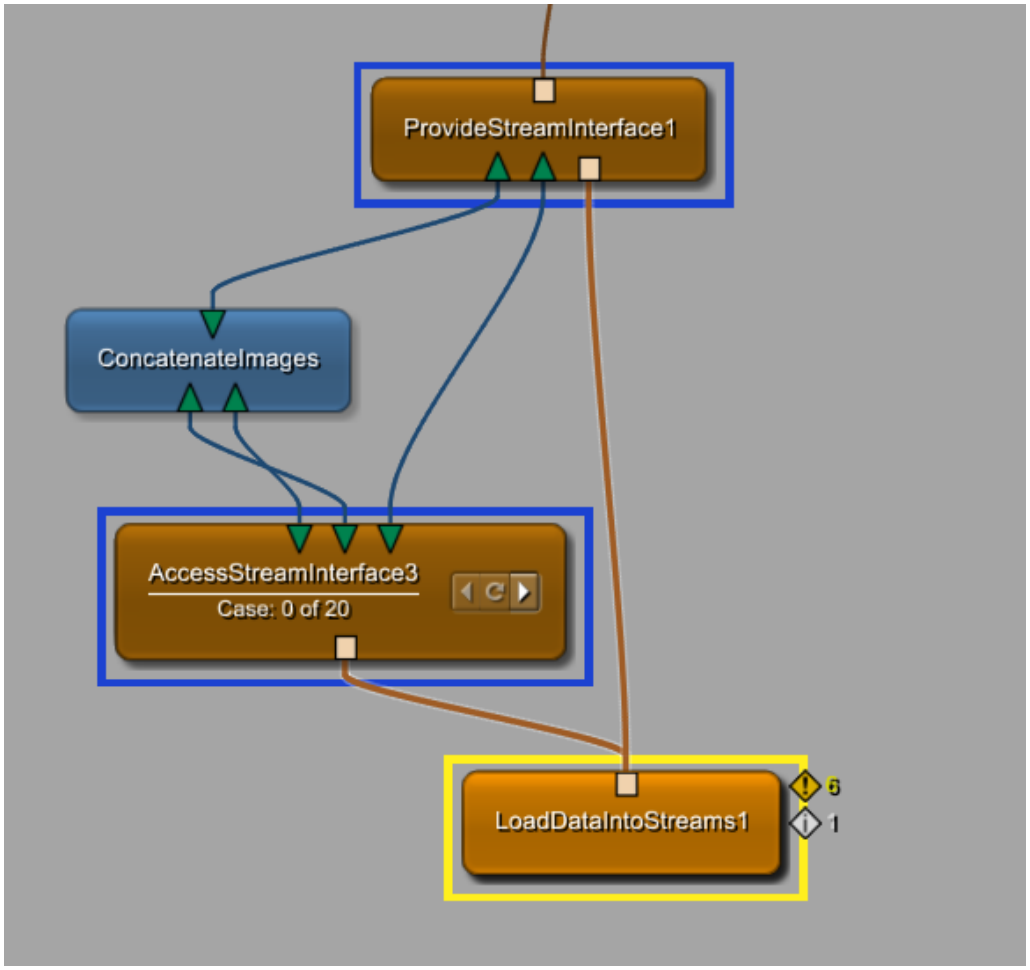


Abbildung 5.8: Pipeline für die 1. Aufgabe der Challenge

Es wurden Level 3 und Level 4 U-Nets für die 1. Aufgabe der Challenge trainiert. Des Weiteren wurden Parameter-Änderungen in der Patch-Size, sowie in der Voxel-Size durchgeführt. Trainiert wurde mit einem Test-Validierungs-Verhältnis von 14:3, wobei die restlichen drei gelabelten Aufnahmen zur Auswertung und zum Testen auf Generalisierbarkeit des trainierten CNNs benutzt wurden. Hier wurde insbesondere das von Team Trainingsschleife entwickelte Modul `ConfusionMatrix` zur Auswertung herangezogen. Im Folgenden werden zwei beispielhafte Trainingsergebnisse diskutiert.

In Abbildung 5.9 ist die Auswertung mithilfe der Confusion-Matrix eines Level 4 U-Nets, das mit einer Patch-Größe von 92x92 und einer Voxel-Size von 2x2 trainiert wurde, zu sehen. Entsprechendes CNN lieferte insgesamt die besten Ergebnisse, gemessen an der addierten, prozentualen Häufigkeit

der richtig erkannten Labels auf einer Aufnahme, mit der nicht trainiert wurde. Die Intention, das CNN mit einer Voxel-Size von 2x2 und mit einer größeren Patch-Größe zu trainieren, lag darin, dass die Leber bei kleinerer Patch-Größe und einer Voxel-Size von 1x1 nicht gut segmentiert wurde. Dies kann damit erklärt werden, dass die Leber auf einigen Slices relativ viel Raum einnimmt und so durch Wählen einer kleineren Patch-Größe auch kleine Gebiete, die der Leber ähneln, aber als Hintergrund gelabelt sind, als Leber erkannt werden.
Den Unterschied kann man deutlich auf Abbildung 5.10 erkennen. Hier ist die Auswertung eines Level 3 U-Nets, das mit einer kleineren Patch-Größe von 44x44 und einer kleineren Voxel-Size von 1x1 trainiert wurde, zu sehen. Die Leber wird deutlich schlechter segmentiert, als bei dem Level 4 U-Net. Außerdem wird auch relativ viel Hintergrund als Leber segmentiert, wie auch im mittleren Bild auf Abbildung 5.10 zu erkennen ist. Die Leber wird außerdem zu 33.3 Prozent als Milz gelabelt, wie in der Konfusion-Matrix zu erkennen ist.

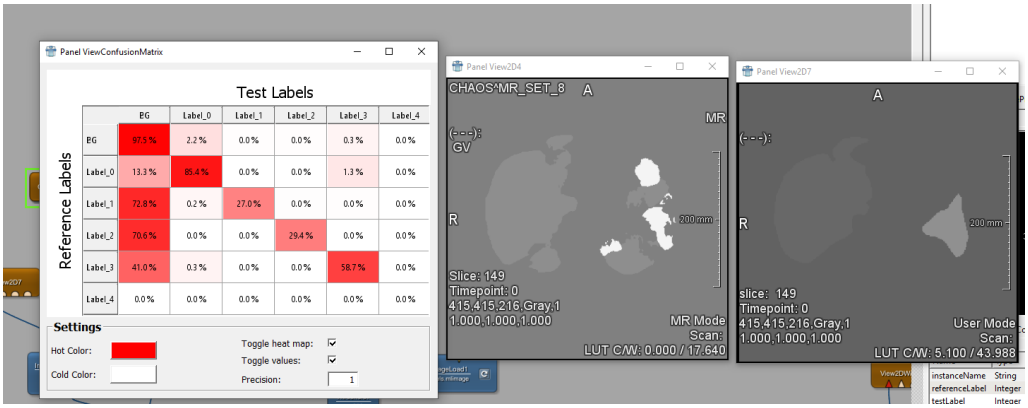


Abbildung 5.9: lvl 4 U-Net 92x92, 2x2 Voxel Size Ergebnisse, dunkelgrau=Leber, hell=Milz, v.l.n.r.: Confusion-Matrix, beispielhaftes Slice der Segmentierung des CNNs, die Maske

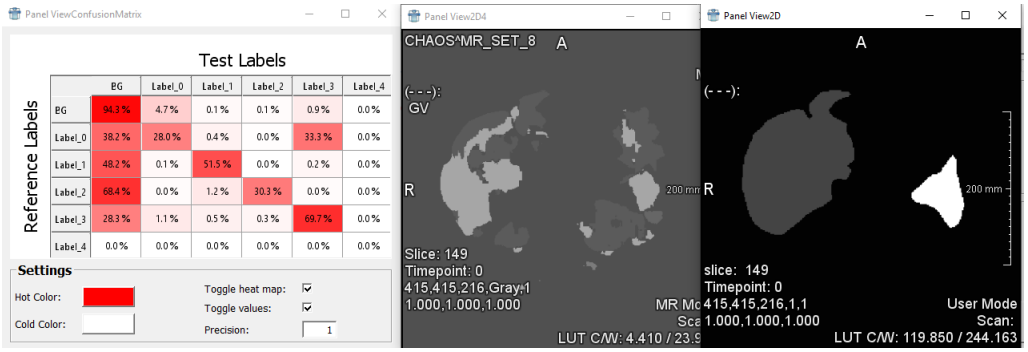


Abbildung 5.10: lvl 3 U-Net 44x44, 1x1 Voxel Size Ergebnisse, dunkelgrau=Leber, hell=Milz, v.l.n.r.: Confusion-Matrix, beispielhaftes Slice der Segmentierung des CNNs, die Maske

Auch in der Tabelle 5.1 ist zu erkennen, dass die Leber einen deutlich höheren Jaccard Wert bei dem Level 4 U-Net aufweist, als bei dem Level 3 U-Net. Interessant ist auch, dass die Jaccard-Werte auf den Validierungsdaten des CNNs deutlich höhere Werte aufweisen, als die Diagonale in der Konfusion-Matrix. Interessant ist es deshalb, weil sie dieselben Werte repräsentieren. Dies ist damit zu begründen, dass ein sehr geringes Datenvolumen zum Trainieren des CNNs zur Verfügung stand und das CNN deswegen nicht gleich gut auf neuen Daten funktioniert.

Name	Accuracy	J* Leber	J* linke Niere	J* rechte Niere	J* Milz
Lvl-3	0.965	0.557	0.606	0.431	0.219
Lvl-4	0.984	0.753	0.691	0.74	0.681

Tabelle 5.1: J* Jaccard

Das implementierte Modul `ConfusionMatrix` von Team Trainingsschleife ist gerade bei Multi-Segmentierungs-Problemen sehr nützlich. Auch, da es nicht nur die reine Accuracy eines Labels berücksichtigt, sondern zudem aufzeigt, welches Label mit welchem verwechselt wurde. Dies bietet neben dem manuellen Durchschauen einzelner Slices objektive Werte zur Beurteilung eines CNNs auf neuen Bilddaten. Die Labels müssen allerdings starr dem Schema One-Hot Encoding entsprechen. Die Implementierung des Auswertungsnetzes in Mevis-Lab hat deswegen erheblichen Aufwand in Anspruch genommen. So mussten durch das Hinzufügen einiger `Threshold`-Module, die Masken jeweils auf One-Hot Encoding gebracht werden, wie auch in Abbildung 5.11 zu sehen ist.

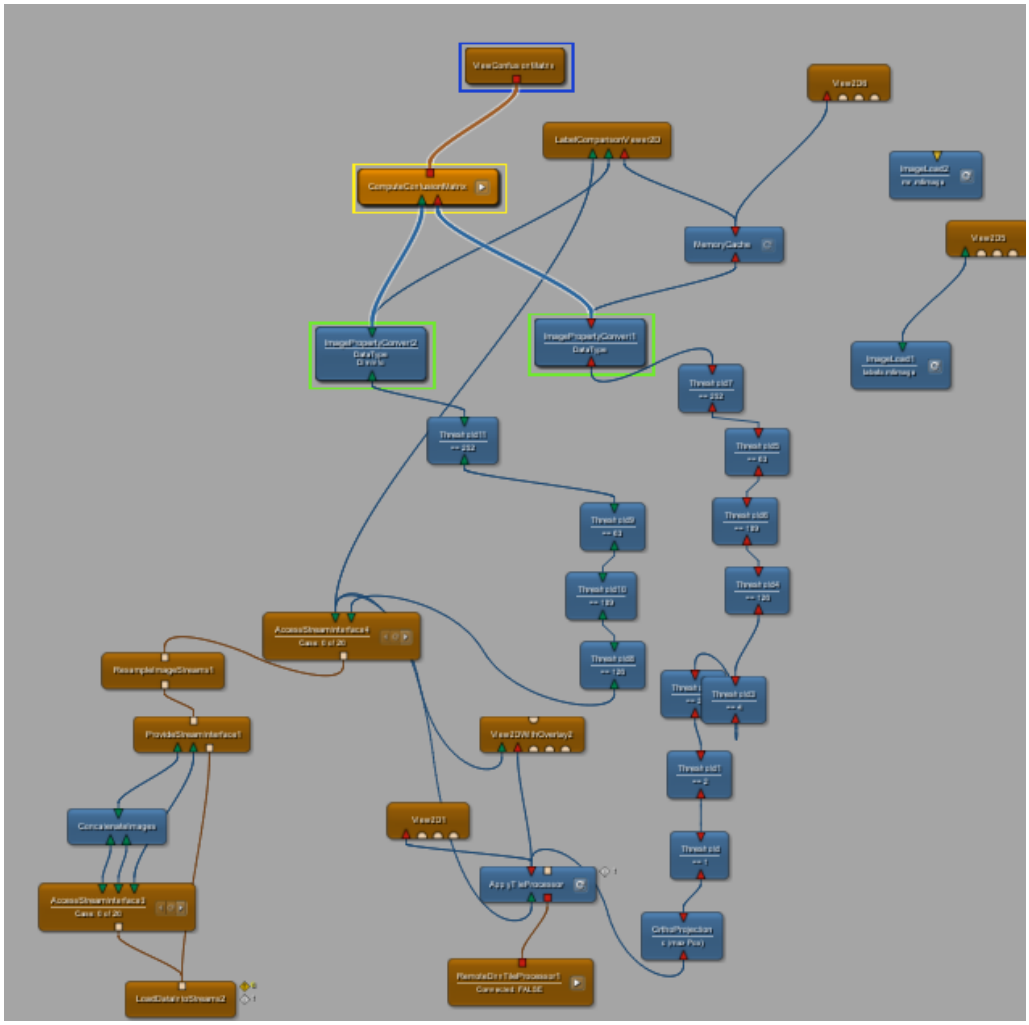


Abbildung 5.11: Mevis-Lab Modulnetz für die Confusions-Matrix

5.5.4 Fazit

17 Patientendaten zum Trainieren eines CNNs, das vier unterschiedliche Organe segmentieren soll, scheint eine zu geringe Datenmenge zu sein, um ein gutes Ergebnis zu erzielen. Zwar wurde die Leber zu circa 85 Prozent richtig erkannt, die Nieren aber jeweils nur zu circa 30 Prozent und die Milz zu circa 60 Prozent. Diese Ergebnisse dienen nicht zur Unterstützung des Arztes und können nicht zur Evaluierung hinzugezogen werden, ob ein medizinischer Eingriff erfolgen darf.

5.6 StructSeg

Autor: Daniel Bröcker, Tom Köhler

5.6.1 Einleitung

Die *StructSeg2019-Challenge* gliedert sich in vier Teilaufgaben. Die Challenge ist Teil der MICCAI 2019 Konferenz gewesen.

Die vier Teilaufgaben sind die Folgenden:

Aufgabe 1: Erkennung von Organen in Kopf-Schulter CT-Scans.

Aufgabe 2: Erkennung von Tumoren im Nasenrachen.

Aufgabe 3: Erkennung von Organen in Brust CT-Scans.

Aufgabe 4: Erkennung von Tumoren in der Lunge.

Zuerst wurde Aufgabe vier, die Erkennung von Tumoren in der Lunge, bearbeitet. Als dort zufriedenstellende Ergebnisse erreicht wurden, hat das Team SructSeg die Konfiguration zusätzlich auf die Aufgabe zwei übertragen. Anschließend wurde die gleiche Trainingspipeline auf Aufgabe zwei und vier gleichzeitig angewandt. Die Datensätze von Aufgabe zwei und vier sind gleich strukturiert, beziehen sich allerdings auf unterschiedliche Organe (Tumoren im Nasenrachen und in der Lunge).

Evaluationskriterum seitens der Challengeherausgeber ist u.a. der Dice Similarity Coefficient (DSC). Der Dice Coefficient gibt die Ähnlichkeit zwischen zwei Datensätzen an. Es ist das am weitesten verbreitete und meist genutzte Werkzeug zur Validierung und zum Vergleich von Bildsegmentierungsalgorithmen.

$$\frac{2 \cdot |X \cap Y|}{|X| + |Y|} \tag{5.1}$$

Das Ergebnis der Formel beschreibt die Ähnlichkeit zweier Mengen. Wenn der Dice 1.0 beträgt, sind die Mengen identisch. Ist der Dice hingegen 0.0,

haben die Mengen kein Element gemeinsam. Bei allen Werten zwischen 0.0 und 1.0 stimmen entsprechend viele Teile überein.

Die Datensätze für Aufgabe zwei und vier bestehen jeweils aus 50 annotierten Serien. Der gesamte Datensatz wurde von einem professionellen Onkologen annotiert und von einem Weiteren verifiziert. Eine Besonderheit dieser Challenge ist, dass die Tumore sehr klein sind, was bedeutet, dass ein verhältnismäßig hoher Anteil der Pixel kein Tumor ist und die Tumore tatsächlich nur einen sehr kleinen Bereich ausmachen.

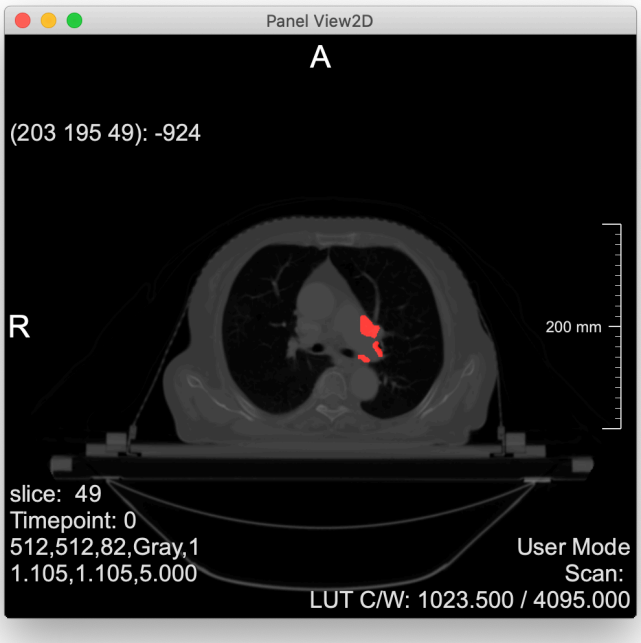


Abbildung 5.12: Eine Schicht eines CT Scans einer Lunge. Tumor in rot.

Aus Abbildung 5.12 wird deutlich, wie viel mehr Pixel zur Lunge gehören als zu dem Tumor. Der Algorithmus muss also jede Schicht des CT Scans nach Tumoren untersuchen und diese entsprechend klassifizieren.

5.6.2 Hintergrund

Die *StructSeg2019-Challenge* fokussiert sich auf die automatische Segmentierung von organs-at-risk (OAR) und grosstarget volume (GTV) von Tu-

moren zur Verbesserung der Strahlentherapie. Die Onkologen kostet es viel Zeit die Strahlentherapie zu planen, denn jedes Organ, was zu schaden kommen könnte und jeder Tumor müssen per Hand segmentiert werden. Dies ist sehr zeitaufwändig und erhöht somit die Kosten der Therapie. Beispiele für Organe, auf die der Onkologe achten muss, wären die optischen Nerven in Augennähe. Das Ziel der Strahlentherapie ist es, die Krebszellen mit genug Strahlung zu treffen und dabei die umliegenden Organe nicht zu sehr zu beschädigen. Deshalb ist die automatische Segmentierung eine wichtige und lebensrettende Aufgabe.

5.6.3 Vorgehen

Diese Challenge wurde zu zweit bearbeitet. Hierdurch konnte relativ effektiv vorgegangen werden. Es wurde wöchentlich zu zweit besprochen, welche Experimente mit welchen Parametern durchgeführt wurden und welche Ergebnisse dies erzielte, sowie Probleme zusammen gelöst. Daraus wurden gemeinsam Rückschlüsse auf die Wirkung der jeweiligen Parameter gezogen und Vermutungen besprochen, welche Parametereinstellungen zielführend sein könnten. Hierbei wurden auch Erfahrungen der restlichen Projektmitglieder mit anderen Challenges besprochen. Anhand dieser Besprechungen wurden grobe Trainingsparameter für die folgende Woche geplant und anschließend durchgeführt und analysiert. Die Analyse wurde größtenteils auf den, oben erwähnten, Dice bezogen, aber auch z.B. auf die Specificity. Da der Algorithmus im Idealfall eine Unterstützung ärztlichen Personals sein sollte, wurde versucht, möglichst selten „Fehlalarme“ auszulösen.

5.6.4 Experimente

Eine Auswahl der durchgeführten Experimente:

Besonderheiten der einzelnen Experimente:

Das Team StructSeg hat zunächst mit der „Standard“-Pipeline begonnen,mit der die meisten Projektmitglieder die Challenges begonnen haben. Da die

Name	Accuracy	Loss	Specificity	Sensitivity	Jaccard	Dice
strct-Lung-02c	0.96	0.113	0.981	0.687	0.55	0.7097
strct-Lung-04	0.945	0.159	0.977	0.618	0.502	-
strct-Lung-04b	0.926	0.196	0.983	0.468	0.412	-
strct-Lung-06b	0.967	0.254	0.984	0.739	0.613	0.76
strct-Lung-07	0.94	0.263	0.957	0.791	0.579	-
strct-Lung+Naso-01	0.965	0.24	0.983	0.758	0.629	0.7722

Tabelle 5.2: Ausschnitt der durchgeführten Trainings

Tumore, wie oben bereits erwähnt, nur einen sehr kleinen Teil der Bildausschnitte ausmachen, wurde zunächst nur auf Bildausschnitten trainiert, welche einen Tumor enthalten (bzw. zumindest Teile eines Tumors), damit das Netzwerk genau lernt, was ein Tumor ist (02c). Als nächstes wurde getestet, welchen Einfluss es hat, die Patchsize zu verringern und verschiedene resampling Strategien anzuwenden(04). Danach wurde die Levelanzahl, die das U-Net hat, von drei auf fünf erhöht und entsprechend das Padding erhöht, damit die Patchsize korrekt ist(04b). In Training 06b wurden die bis hier besten Strategien verwendet und das Resampling auf transversal gestellt. Bei Training 07 wurden die Merkmale der Trainings 06b und 04b kombiniert, dies erzielte jedoch keine weitere Verbesserung.

Da die Experimente gute Ergebnisse erzielten, wurde die Architektur nie grundlegend verändert, sondern lediglich die Konfiguration verändert. So wurde durchweg mit einem Tensorflow U-Net trainiert, aber es wurden verschiedene Levelanzahlen getestet.

5.6.5 Fazit

Im Vergleich zu den offiziellen Teilnehmern der Challenge erzielte das Team StructSeg deutlich bessere Ergebnisse. So erzielten es bei der Teilaufgabe 4 z.B. einen Dice von 0.76, während das bestplatzierte Team der offiziellen Challenge einen Wert von nur 0.54 erzielte. Dies ist aber vermutlich auch darauf zurückzuführen, dass die offizellen Ergebnisse an einem gesonderten Validierungsdatensatz gemessen wurden, welcher vermutlich schwerere Fälle

enthielt. Wir konnten unseren Dice im Laufe des Projektes verbessern und haben einen, nach den üblichen Maßstäben, annehmbaren Wert erzielt.

Die Erfahrungen, welche im Rahmen der Bearbeitung dieser Challenge gesammelt wurden, wurden anschließend teilweise verwendet, um ein MeVisLab-Modul für Standartfälle der Segmentierung (5.9) zu erstellen.

5.7 Challenge xVertSeg

Autor: Joana Becker

5.7.1 Einleitung

Innerhalb des Team Challenge hat sich die Gruppe in mehrere einzelne Teilgruppen aufgesplittet und verschiedene Challenges bearbeitet. Eine dieser Challenges trägt die Bezeichnung xVertSeg und wird vom Laboratory of Imaging Technologies (University of Ljubljana, Faculty of Electrical Engineering, Slovenia) veröffentlicht. Die Challenge besteht aus zwei Teilen. Der eine Teil befasst sich mit der Segmentierung der Lendenwirbel L1-L5, der andere Teil mit der Klassifikation von Wirbelfrakturen an den segmentierten Wirbeln. Da für diese zwei Aufgabenteile verschiedene Arten von Netzen benötigt werden, eins zur Segmentierung und eins zur Klassifikation, wurde der zweite Aufgabenteil weggelassen. Dadurch wurde der Arbeitsumfang reduziert und die Übernahme weiterer Aufgaben ermöglicht. Das Thema der Challenge wurde ausgewählt, weil im persönlichen Umfeld die Wirbelsäule ein zentrales Thema war und daher bereits Interesse bestand.

5.7.2 Datensatz

Um die Aufgabe bearbeiten zu können, stellt die Website der Challenge einen Datensatz zum Download zur Verfügung. Die darin enthaltenen Daten enthalten Trainingsdaten für beide Aufgabenteile. Für die Segmentierung werden Bilder und dazu passende Wirbelsegmentierungen benötigt. Die Bilder der Challenge sind CT Aufnahmen von einer Wirbelsäule, die dazugehörigen Masken enthalten Label für alle 5 Lendenwirbel. Insgesamt ist die Größe des Datensatzes ziemlich gering. Es werden nur 15 Trainingsbilder mit Masken geliefert und nur weitere 10 Bilder ohne Masken.

Die Bilder werden durch eine Header-Datei und eine RAW-Datei repräsentiert, die beim Einbinden in das MeVisLab-Netz mit den dazugehörigen Masken zusammengefügt werden.

5.7.3 Netze

Als Einstieg für ein Netz wurde ein bereits bestehendes Tensorflow U-Net für die Segmentierung mehrerer Strukturen von MeVis verwendet. Es enthielt Module zum Laden von annotierten Bildern, randomisiertem Ändern der Perspektive, Resampling von Streams in ein Koordinatensystem, Caching der Dateien ins mlimage-Format, Patch-Extraktion, einem Modul zum Senden der Daten in einen Deep Learning Prozess, sowie dem Anzeigen von Bildern und dazugehörigen Labels. Dieses Netz war bereits lauffähig und musste nur angepasst und erweitert werden. Beispielsweise wurde es um jeweils einen `StratifiedPatchSampler` für Trainingsdaten und Validierungsdaten erweitert, die die Sampling-Strategie passend zu den zugewiesenen Patch-Klassen änderten. Den Patchsamplern wurden Sample-Frequenzen für die einzelnen Layer von Hintergund und Labeln übergeben. Mit den daraus resultierenden Werten wurde das Netz trainiert.

Um die Funktionalität des trainierten Netzes zu testen, wurde ein zweites Netz erstellt, wieder aus einem bereits bestehenden Test-Netz. Dieses Test-Netz bekommt Testdaten und die berechneten Ergebnisse des aktiven trainierten Netzes. Das Test-Netz wurde so bearbeitet, dass es beispielsweise nur zusammenhängende Bereiche anzeigt, um das Anzeigen fehlerhaft berechneter Bereiche möglichst zu vermeiden.

5.7.4 Training

Das Netz wurde hauptsächlich mit 4 Leveln und in der Sagittalebene trainiert. Je nach Ergebnis der Trainings wurden Werte wie Padding und Patch Size verändert und neue Trainings gestartet. Dabei sind anfangs häufig Probleme durch Unwissenheit aufgetreten. Beispielsweise war unklar, wie genau die Eingabe der Werte bei den PatchSamplern auszusehen hat oder einzelne Eingaben wurden schlichtweg in der Menge der Werte übersehen. Dies stellte sich dann in den trainierten Netzen heraus, indem die Netze auf Testdaten nichts Sinnvolles anzeigten. Nachdem solche Probleme behoben waren, konnten die variablen Werte mit den folgenden Trainings laufend verändert und je nach Ergebnis in eine bestimmte Richtung angepasst werden.

5.7.5 Ergebnisse und Hippocampus-Challenge

Mit fortschreitenden Trainings stellte sich heraus, dass zwar gewünschte Lendenwirbel segmentiert wurden, aber bei Weitem nicht so erfolgreich, wie es wünschenswert gewesen wäre. Hier machte sich bemerkbar, dass sich mit gerade mal 15 Trainingsdaten nicht wirklich erfolgreich trainieren lässt. Es wurde daraufhin ein Versuch gestartet, mit Satori weitere Daten zu annotieren, der wegen verschiedener technischer Probleme abgebrochen wurde. Daraufhin wurde entschieden, dass eine Weiterarbeit an dieser Challenge keinen wirklichen Mehrwert bringt.

Die erlernten Fähigkeiten wurden daraufhin an einer weiteren Challenge angewendet. Diese Challenge, namens Hippocampus, segmentiert die Hippocampi im menschlichen Gehirn. Mit 260 zur Verfügung gestellten Trainingsdaten wurden hier zufriedenstellende Ergebnisse nach kurzer Zeit erzielt, weshalb diese Challenge nicht lange bearbeitet und zum Team Website für das Erlernen weiterer Fähigkeiten gewechselt wurde.

5.8 Chaos-Lebersegmentierung in MRI/MRT

Autor: Johann Keuneke

5.8.1 Einleitung

Da sich die Mitglieder des Team-Challenges in mehreren Einzelgruppen verteilt haben, lag es nah, dass verschiedene Challenges bearbeitet werden. Da es ein große Interesse an der Chaos-Challenge (https://chaos.grand-challenge.org/) gab, wurde diese Challenge von 4 Personne bearbeitet. Die Chaos-Challenge selber hatte 4 verschiedene Teilaufgaben, weswegen jeder der Chaos-Teammitglieder eine unterschiedliche Aufgabe zu bewältigen hatte.

Die Aufgabe war es, ein „Convolutional Neural Network“ zu trainieren, welches auf verschiedenen MRI/MRT Aufnahmen die Leber des Patienten segmentieren sollte. Dafür waren von Chaos selber vorbereitete DICOM Datensätze erhältlich. Diese enthielten 60 Fälle zum Trainieren und 60 Fälle zum Testen. Diese 120 Fälle wurden jeweils in verschiedenen Phasen aufgenommen, somit ergaben sich 40 T1-DUAL inphase, outphase und 40 T2-SPIRE Datensätze, wovon jeweils die Hälfte bereits annotiert war.

5.8.2 Implementierung

Die ersten Schritte im Projekt waren die Annotierung eines Knies, wobei für je 3 Patienten in einem Knie CT, mit der Hilfe des Programms Satori, die Tibia, Femur und Partella annotiert wurden. Dabei wurde gelernt, wie die Masken für Dateien erstellt werden. Im Folgenden wurde ein Training für die erstellten Datensätze und damit das eigentlich erste Training, welches wir zunächst auf unserem eigenen Rechner zum Laufen gebracht haben, gestartet. Dafür wurde zunächst eine MeVisLab-Datei erstellt, in der die erstellten Datensätze aufgeteilt wurden in Trainings- und Testfälle, die Bilddaten in Patches zerschnitten und dann vom Netz zum Training benutzt wurden.

Die Vorarbeit mit MeVisLab führt zu verschiedenen Qualitäten der Netze und war im Verlauf unseres Projekts die Hauptarbeitsquelle. Diese soge-

nannte Pipeline wurde immer weiter verändert, um das optimale Netz zu trainieren.

Das zweite Training, mit den gleichen Datensätzen, das wir gestartet haben, lief dann über das Rechennetz des Fraunhofer Instituts. Das Training konnten wir mit der Hilfe von Quantmed einfach Online starten und laufen lassen. Quantmed ist ein webbasiertes Interface, welches es ermöglicht die Pfade für das benötigte Training anzugeben und diese dann auf den externen Rechennetzen zu starten. Dies hat den Vorteil, dass die Netze wesentlich schneller ihr Training durchführen können.

Nachdem es eine Einführung in die Programme und Hilfsmodule gab, wurde sich daran gesetzt, unsere Challenge zu bearbeiten. Das Erste, was getan wurde, war der Download unserer Datensätze. Diese bestanden aus DICOM Bildern und dazugehörigen Masken, welche im PNG Format abgespeichert waren. Um diese Bilddaten in MeVisLab zu verarbeiten, mussten zunächst die PNG Dateien in DICOM umgewandelt und auf unsere Rohdaten projiziert werden. Dafür benutzten wir ein von unseren Betreuern, in MeVisLab, erstelltest Programm.

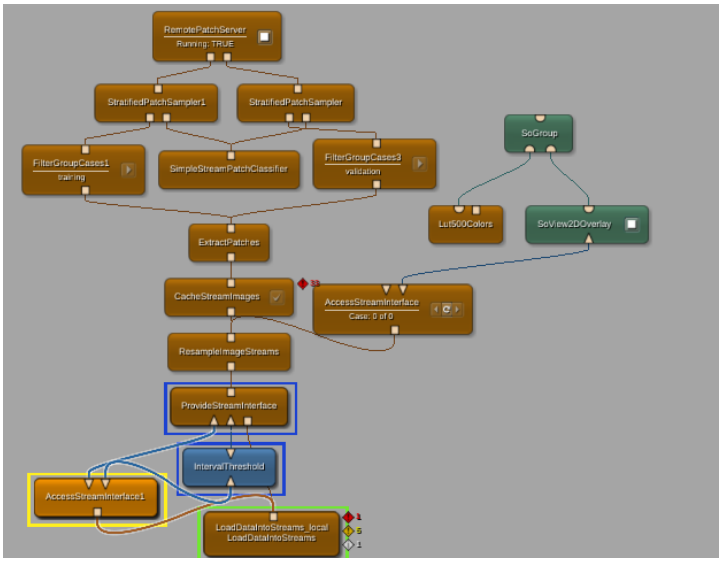


Abbildung 5.13: MeVisLab Pipeline

Nachdem die Dateisätze brauchbar waren, wurden diese auf etwaige Fehler überprüft und korrigiert.

Nun wurden die ersten Trainingsschleifen gestartet für die einzelnen Gruppen. Das erste Training wurde mit der Pipeline gestartet, welche mit der Hilfe der Betreuer erstellt wurde. Diese hatte die Standardeinstellung für ein Level 3 U-Net sowie die Minimum Patchsize. Die einzige Veränderung, die vorgenommen wurde war, dass das Modul `IntervallThreshold` verändert wurde. Die Masken der Chaos Datensätze, beinhalteten alle Organe, welche in der Challenge vorkommen. Die verschiedenen Organe hatten unterschiedliche Grauwerte auf der Maske. Somit wurde durch das Modul `IntervallThreshold` der Grauwert so gefiltert, dass nur die Leber in der Maske enthalten war.

Leider sind die Resultate dieser Trainingsschleifen überschrieben worden, wodurch sie nicht hier aufgezeigt werden können.

In den nächsten Trainingsschleifen wurde nur mit der *Patchsize* experimentiert, um zu sehen was für eine Auswirkung dieses auf die Netze hatte. Zunächst verdoppelten wir die Patchsize von 44x44 auf 88x88

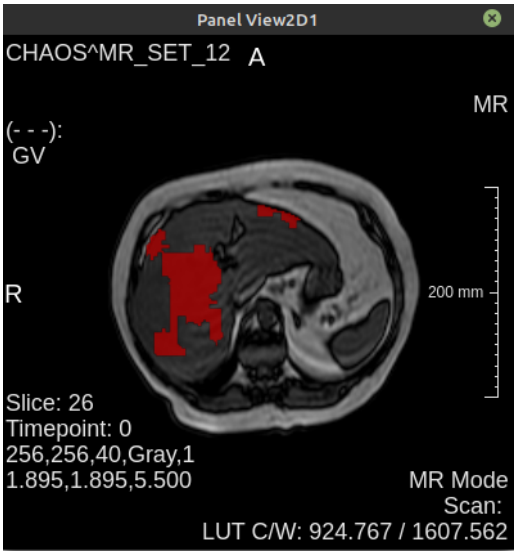


Abbildung 5.14: Resultat des Trainings mit Standardeinstellungen

Wie auf Abb.5.14 zusehen ist, wird nicht die ganze Leber erkannt. Kaum etwas der Ränder des Organs wird markiert, nur Bereiche aus dem Zentrum werden erkannt. Im Verlauf werden Bilder für die Resultate von dem gleichen Patienten genommen. Dabei wird versucht ein Slice zu nehmen, welches die beste Abdeckung hat.

Danach wurde mit dem Validierungsintervall experimentiert. Die MeVisLab Datei blieb unverändert, jedoch wurde der Abstand der Validierung des Netzes auf alle 500 Iterationen angehoben. Der Wert 500 wurde gewählt, da nach Absprache mit einigen Teammitgliedern, welche an einer anderen Teilaufgabe der Chaos-Challenge arbeiteten, bereits verschiedene Intervalle getestet wurden und die besten Resultate zwischen einem Validierungsintervall von 400 und 700 lagen.

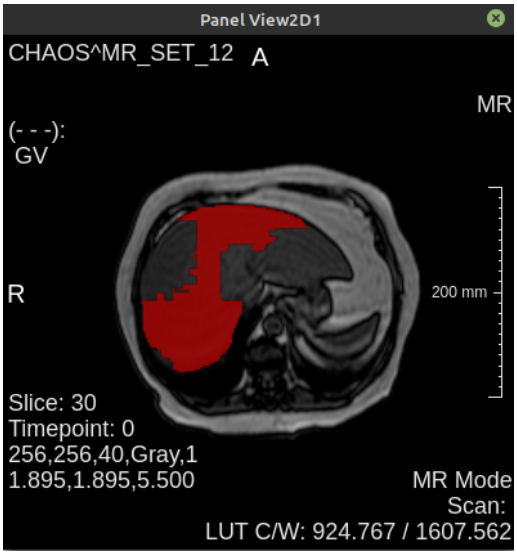


Abbildung 5.15: Resultat des Trainings mit erhöhtem Validierungsintervall

Wie man im Vergleich von Abb.5.15 und Abb.5.14 sehen kann, ist eine deutliche Verbesserung zu erkennen. Die Ränder werden wesentlich besser erfasst und ein Großteil des Organzentrums ebenfalls. Des Weiteren sieht es so aus, als ob rechteckige Flächen aus der Maske geschnitten wurden.

Das nächste und letzte Training, das wir gemacht haben war, ein höheres U-Net Level zu probieren, da wir bis dahin nur mit einem U-Net Level 3 gearbeitet haben. Nun wurde ein Netz mit einem U-Net Level 4, der Patchgröße 92x92 und einem Validierungsintervall von 500 gestartet und für 48h trainiert.

Wie man auf Abb.5.16 sehen kann, wurde fast das ganze Organ erkannt. Die Ränder der Maske sind klar segmentiert und haben wenig bis keine Einbrüche. Des Weiteren bleibt das Problem, dass gelegentlich Teile des Organs nicht erkannt werden, wie beispielsweise die rechte Spitze des Organs.

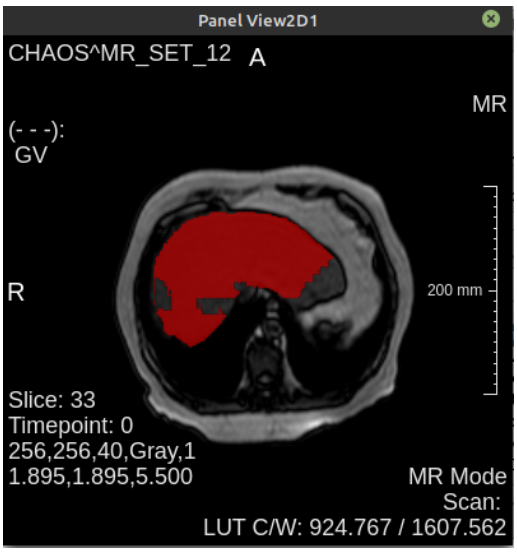


Abbildung 5.16: Resultat des Trainings mit höherem U-Net level

Um die verschiedenen Netze zu vergleichen oder zu bewerten, gab es wenige Möglichkeiten. Hauptsächlich wurden 10 Fälle stichprobenartig betrachtet und mit den gleichen Fällen vom vorherigen Netz verglichen. Im Verlauf des Bachelorprojekts wurde dem Team Challengr vorgestellt. Eine Applikation, welche automatisch alle Testfälle mit einem gegebenen Netz ausprobiert und Abdeckungswerte zurückgibt. Dieses Programm war zu dem Zeitpunkt aber leider nicht auf dem Rechennetzwerk nutzbar, sondern nur auf unseren eigenen Computern. Dafür fehlte die Rechenleistung, weswegen diese Option der Validierung wegfiel. Gegen Ende des Bachelorprojekts kam die Idee auf, dass es möglich wäre zwei Netze in Echtzeit vergleichen zu könnte, indem man die Masken beider Netze gleichzeitig auf eine Bilddatei projiziert und deren Alphawert so anpasst, dass die Unterschied der beiden Masken erkennbar werden.

5.8.3 Fazit

Im Großen und Ganzen war die Arbeit mit CNNs sehr interessant, aber auch sehr deprimierend, da wir hauptsächlich nur verschiedene Einstellungen ausprobiert haben und danach verglichen, welcher dieser Einstellungen zu besseren Resultaten geführt hatte. Dies ist wohl darauf zurückzuführen, dass das Team keinerlei Vorerfahrung im Bereich von CNN oder dem Trai-

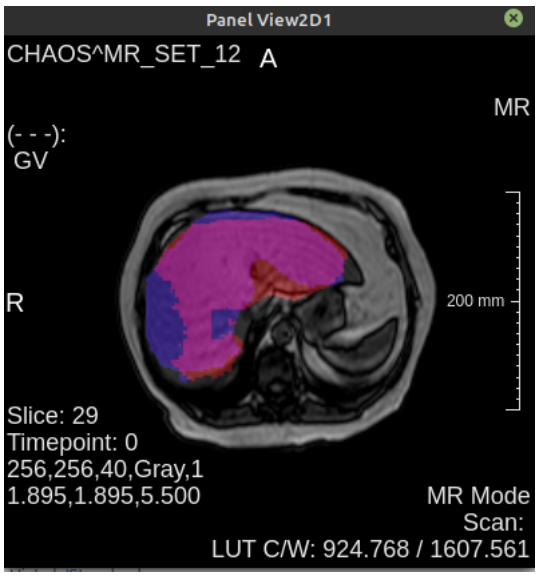


Abbildung 5.17: Vergleich der Resultate von zwei trainierten Netzen mit einander.

nieren dieser hatte, um vorher zu erkennen, was für Auswirkungen etwaige Veränderungen mit sich ziehen. Trotz dessen war die Freude umso größer, wenn eine Verbesserung zum vorherigen Ergebnis erzielt wurde.

Die Menge an Informationen und Ressourcen, die wir durch die Betreuer erhalten haben, war ebenfalls eine große Hilfe besonders beim Trainieren der Netze. Hätten die Netze auf unseren eigenen Rechnern trainiert, wäre wohl nur einen Bruchteil der Arbeit geschafft worden, welche wir durch die Rechennetze vollbracht haben.

Ein weiterer Punkt, welcher oft versucht, aber selten geschafft wurde, war die Kommunikation zwischen den Teams. Da zum Großteil in Einzelgruppen und dazu noch an verschiedenen Aufgaben gearbeitet wurde, gab es sehr wenig über das sich die Teams austauschen konnten. Es wurde sich hauptsächlich über Probleme ausgetauscht, auf die fast nie eine Lösung gefunden wurde, da wir zu wenig mit dem Framework vertraut waren. In den meisten Fällen wurden die Probleme daraufhin von den Betreuern gelöst.

5.9 Standatisiertes Segmentierungs Modul

Autor: Daniel Bröcker

5.9.1 Einleitung

Die Anzahl an Parametern und Möglichkeiten zur Anwendung eines neuronalen Netzes zur Segmentierung von Bildern ist immens. Ziel bei der Entwicklung dieses Moduls ist es, einen Schritt in Richtung einer vollständigen Lösung für binäre Segmentierungsprobleme zu gehen. Dies soll einerseits helfen, binäre Segmentierungsprobleme mit weniger Arbeitsaufwand lösen zu können und anderseits Einsteigern durch die vereinfachte Oberfläche als Unterstützung dienen. Dieses Modul bietet ein geschlossenes Paket, welches alles Nötige für eine binäre Segmentierung in einem einzelnen Modul bindet. Durch die Architektur MeVisLabs kann es dennoch erweitert und für andere Zwecke verändert werden. Es wurde eine übersichtlichere Oberfläche für diesen Zweck entwickelt. Sie zeichnet sich dadurch aus, dass nur die wichtigsten Optionsmöglichkeiten angeboten werden. Durch die Vereinfachung ist der Anwendungszweck der Pipeline auf binäre Segmentierungen beschränkt. Außerdem wird der Nutzer unterstützt, indem falsche Eingabewerte automatisch in korrekte Werte abgewandelt werden.

Die Entscheidungen, welche Optionen als „nötig für die Oberfläche“ zählen (siehe Grafik unten 5.18), wurden anhand der Erfahrungen und Berichte des Challenge Teams sowie der Empfehlungen der Projektleiter Felix Thielke und Hans Meine festgelegt.

5.9.2 Implementierung

Als Basis für dieses Modul dient dasselbe MeVisLab-Netzwerk, welches auch von den meisten aus dem Challenge Team verwendet wurde, um ihre jeweilige Challenge anzufangen.

Ziel bei der Erstellung der Oberfläche war es, wie oben bereits erwähnt, eine möglichst verständliche und übersichtliche Oberfläche zu erreichen. Um dies zu unterstützen, wurde nur eine Auswahl an Optionen in die Oberfläche

übernommen. Teilweise wurden die Optionen eins zu eins von den internen Modulen an die Oberfläche des neuen Moduls weitergeleitet. Teilweise wurden Eingabedialoge vereinfacht, zusammen gruppiert oder umgestaltet, sodass sie verständlicher sind.

Abbildung 5.18: Panel des Segmentierungsmoduls

Die Angabe der Patchgröße wurde vereinfacht, indem nur noch jeweils ein Wert für die Patchgröße und für die Anzahl der Level des U-Nets angegeben wird. Die Größe des Paddings wird nun durch das Modul selbst berechnet und automatisch verwendet. Außerdem werden Patchgrößen, welche nicht mit der angegebenen Levelanzahl kompatibel sind, automatisch zu dem nächsten passenden Wert korrigiert und angewandt. Die Oberfläche des Moduls zeigt außerdem das automatisch gewählte Padding und die letztendliche gesamte Patchgröße an, sowie einen Indikator, ob die Patchgröße im 2D oder 3D Modus eingestellt ist. Über die angebotenen Schaltflächen kann auch automatisch zum nächsten größeren oder kleineren gültigen Wert gesprungen werden.

Die Einstellung des Verhältnisses zwischen den beiden Klassen beim Training und beim Validieren kann nun auch über eine vereinfachte Oberfläche angegeben werden. Statt der ursprünglichen (sehr flexiblen) Eingabe eines regulären Ausdrucks wird nun durch eine Prozentangabe der Anteil der Daten mit Vordergrund jeweils angegeben. Außerdem kann die Stratifizierung des Trainings und der Validierung nun jeweils mit einem einzelnen Klick aktiviert bzw. deaktiviert werden.

Die Einstellmöglichkeiten des internen Moduls, welches für die Ausgabe verantwortlich ist, sind eins zu eins in die Oberfläche des neuen Moduls

weitergeleitet worden, da hier eine Vereinfachung keinen Sinn ergibt. Weil diese alle nötig sind und auch für diesen Zweck bereits ideal angeordnet sind.

Allerdings wird der 2D/3D Wechselschalter des Moduls jetzt auch verwendet, um die Patchgröße automatisch an die jeweils gewählten Dimensionen anzupassen.

5.9.3 Fazit

Die Architektur von MeVisLab ermöglicht es, dieses Modul in andere Module einzubinden. So könnte es beispielsweise erweitert werden, um auch nicht-binäre Segmentierungsprobleme lösen zu können oder in eine Weboberfläche eingebunden zu werden.

6 Team Trainingloop

6.1 Einleitung

Autor: Lukas Schäfer

Das Team Trainingloop beschäftigt sich mit der Verbesserung der Trainingsschleife für eine Segmentierung, die aus Preprocessing, Training des neuronalen Netzes und dem Postprocessing besteht. In diesem Kapitel wird zunächst ein Überblick über die Ziele der gesamten Gruppe in Abschnitt 6.2 gegeben. Anschließend werden in Abschnitt 6.3 und Abschnitt 6.4 die Entwicklungen der beiden erstellten Module beschrieben. In Abschnitt 6.5 wird ein Konzept zur Optimierung einer Segmentation Pipeline vorgestellt und ein Ausblick auf die mögliche Umsetzung für Fraunhofer MEVIS gegeben. Im abschließenden Abschnitt 6.6 wird das Fazit der gesamten Gruppe gezogen.

6.2 Motivation/ Ziele

Autor: Lukas Schäfer

Das Hauptziel des Teams war es, Module und Konzepte zu entwickeln, die es anderen Personen ermöglichen sollen, neuronale Netze effektiver und effizienter zu trainieren und analysieren. Außerdem soll die Nutzung von einer Pipeline für verschiedene Segmentierungsprobleme ermöglicht werden.

Innerhalb der Gruppe werden drei unterschiedliche Module und Konzepte entwickelt.

Die erste Untergruppe beschäftigt sich mit der Entwicklung eines MeVisLab-Moduls, das ein bereits trainiertes Netz analysieren und die Ergebnisse übersichtlich darstellen soll. Im Laufe der Einarbeitung in MeVisLab wurde die Confusionmatrix als Darstellungsform gewählt, die als Modul umgesetzt werden soll. Mithilfe der Ergebnisse dieser Confusionmatrix kann das Training des analysierten Netzes angepasst werden.
Das MeVisLab-Modul, welches von der zweiten Untergruppe entwickelt wird, konvertiert die unterschiedlichen Kodierungen, in denen Datensätze vorliegen können. In dem Modul werden die Kodierungen in einem One- bzw. Multi-Hot-Vector, als Bitmaske und als Integer Label berücksichtigt. Dadurch soll das Preprocessing innerhalb der Segmentierungspipeline erleichtert werden.
Die letzte Untergruppe entwirft ein Konzept zur Nutzung eines nnU-Net zur Optimierung einer Segmentierungspipeline für unterschiedliche Segmentierungsprobleme. Bisher musste die Pipeline für die Nutzung in unterschiedlichen Aufgabenstellungen manuell angepasst werden, dieser Vorgang soll automatisiert werden. Außerdem wird ein Ausblick auf eine mögliche Umsetzung des Konzepts für das Fraunhofer MEVIS gegeben.

6.3 Confusionsmatrix

6.3.1 Einleitung

Autor: Lukas Schäfer

In diesem Kapitel wird die Entwicklung des Confusionmatrix-Moduls beschrieben. In Unterabschnitt 6.3.2 wird dargestellt, was das Team zur Entwicklung des Moduls angeregt hat und welche Ziele mit dem Modul erreicht werden sollen. Im anschließenden Unterabschnitt 6.3.3 wird das Konzept der Confusionmatrix erläutert. DerUnterabschnitt 6.3.4 beschreibt das Vorgehen während der Entwicklung. Das Berechnen der Confusionmatrix, sowie weitere Funktionen im Zusammenhang mit der Darstellung und weiteren Verarbeitung, werden in sechs MeVisLab-Modulen implementiert. Die Implementierung der Module wird in Unterabschnitt 6.3.5 dargestellt. Im

folgenden Unterabschnitt 6.3.6 wird das Ergebnis der Implementierung präsentiert. Die Integration der Confusionmatrix in Challengr wird in Unterabschnitt 6.3.7 beschrieben. Abschließend wird in Unterabschnitt 6.3.8 ein Fazit gezogen und in Unterabschnitt 6.3.9 ein Ausblick auf eine mögliche Zukunft des Moduls gegeben.

6.3.2 Motivation/Ziele

Autor: Lukas Schäfer

Als erste Idee wurde ein Tool vorgeschlagen, das als Hilfe dafür dienen soll, ein trainiertes neuronales Netz auszuwerten. Die Informationen, die durch die Analyse des Netzes gewonnen werden, sollen anschließend eine Verbesserung des Trainings ermöglichen. Das zu entwickelnde Modul soll schnell und übersichtlich anzeigen, wie gut ein Netz trainiert ist. Hierfür soll eine Confusionmatrix erstellt werden, welche die Annotationen eines MRT-Bildes, die durch einen Experten und durch ein neuronales Netz erstellt wurden, miteinander vergleicht. Dabei sollen verschiedene Label der Annotation einzeln verglichen werden, um eine genaue Vorstellung davon vermitteln zu können, in welchen Bereichen ein trainiertes Netz Schwierigkeiten aufweist. Das Modul soll ebenfalls die Möglichkeit bieten, die Bereiche, in denen es gehäuft zu Fehlern in der Annotation durch das neuronale Netz kommt, visuell darzustellen.

6.3.3 Was ist eine Confusionmatrix?

Autor: Markus Rink

Eine Confusionmatrix oder Error Matrix vergleicht den Soll- und Ist-Zustand bei einem Klassifikationsverfahren. Die Klassen werden in Reihe und Spalte gleichmäßig aufgelistet und spannen so eine Matrix auf. Die Reihen repräsentieren dabei die Ground Truth Klassen und die Spalte die vom Netz zugeordnete Klasse oder vice versa. Ein Feld der Matrix zählt die Anzahl an Zuordnungen, für die das Paar Ground Truth/Prädiktion zutrifft.
Werden alle Klassen getestet und klassifiziert das Netz alle Eingaben korrekt, entsteht in der Matrix eine Diagonale mit Werten > 0. In diesem

Szenario haben alle anderen Felder den Wert 0. Zur Visualisierung wird häufig eine Farbkodierung verwendet, sodass eine Heatmap entsteht.
Für Segmentierung als spezielle Klassifikation wird jeder Voxel je nach Label in der Matrix zugeordnet. Zum besseren Verständnis wurden die absoluten Werte in Prozentzahlen umgerechnet.

6.3.4 Vorgehen

Autor: Nina Unterberg und Jan-Gerrit Göbel

Da keiner im Team vor dem Projekt bereits mit MeVisLab gearbeitet hatte, wurden die ersten Wochen dafür verwendet sich mit MeVisLab und den Modulen auseinanderzusetzen. Alle Mitglieder die vorher noch nicht mit Python gearbeitet hatten, haben sich zudem in diese Programmiersprache eingearbeitet. Während dieser Zeit wurde auch das Ziel dieser Untergruppe spezifiziert und schriftlich festgehalten.
In den folgenden Wochen wurde zunächst ein MeVisLab-Modul erstellt welches alle Funktionen in Bezug auf das Ziel umsetzt, von der Berechnung der Confusion Matrix bis hin zur Darstellung der Matrix, sowie das Anzeigen von Unterschieden in Annotationen. Auf Anregung durch die Projektleiter wurde das Modul in mehrere Module mit spezialisierten Funktionen aufgeteilt. So gibt es für jede angedachte Funktion ein eigenständiges Modul. Der größte daraus entstandene Vorteil ist die beliebige Kombination von benötigten Funktionen und das Weglassen von nicht benötigten Komponenten. Die entstandenen Sechs Module und ihre Funktion werden in den folgenden Kapiteln näher erläutert.
Zudem wurde eine Integration in Challengr vorgenommen, um direkt von den entworfenen Modulen zu profitieren und sie schneller nutzbar zu machen.

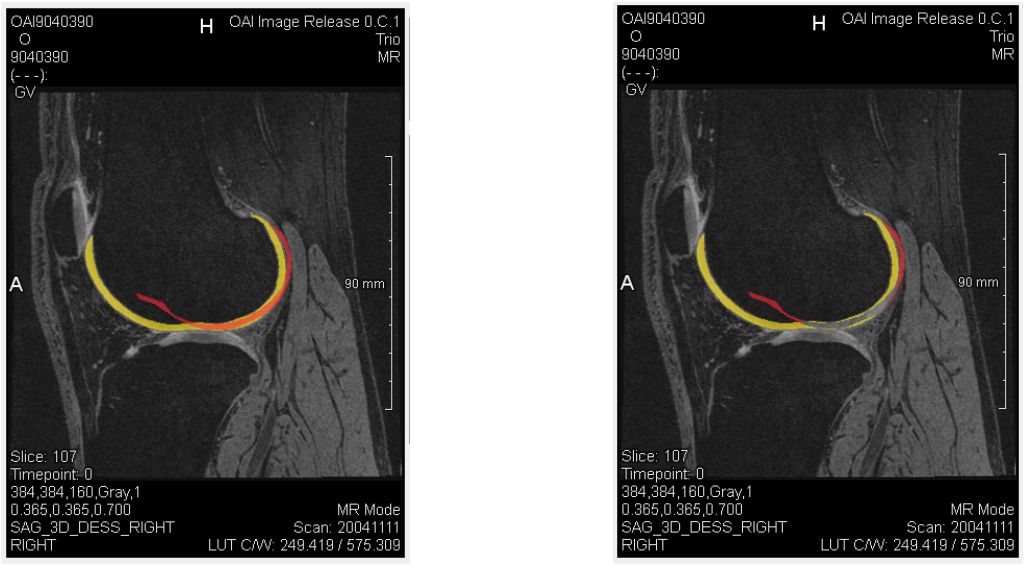
6.3.5 Implementierung

6.3.5.1 LabelComparisonViewer2D

Autor: Nina Unterberg

Das `LabelComparisonViewer2D` Modul soll den Anwender dabei unterstützen Unterschiede zwischen zwei Annotationen zu identifizieren.
Das Modul hat drei Eingabefelder: das Bild, die Testannotationen und die Referenzannotationen. Über die Benutzeroberfläche des Moduls können verschiedene Einstellungen vorgenommen werden:

- Festlegung der Anzahl der Test- und Referenzlabels
- Der Modus für die Visualisierung. Es stehen 5 Modi zur Auswahl: Union, Intersection, Symmetric Difference, Reference Difference und Test Difference. Vier dieser Modi sind in Abbildung 6.1 und Abbildung 6.2 dargestellt. Das Referenzlabel ist in diesem Beispiel gelb und das Testlabel rot eingefärbt.
- Auswahl der Farbe und des Alphawertes des dargestellten Referenz- und Testlabels



(a) (b)

Abbildung 6.1: Verschiedene Modi von dem `LabelComparisonViewer2D` Modul: (a) Union; (b) Symmetric Difference;

(a) (b)

Abbildung 6.2: Verschiedene Modi von dem `LabelComparisonViewer2D` Modul: (a) Intersection; (b) Reference Difference;

Die meiste Logik des Moduls wird über das interne MeVisLab Netzwerk realisiert. Über ein `ExtractLabels` Modul wird jeweils das, durch die GUI, ausgewählte Referenz-/ Testlabel extrahiert. Diese werden über zwei `Arithmetic2` Module, je nach eingestelltem Modi, kombiniert. Durch ein `SoViewOverlay` Modul wird, entsprechend der Einstellungen, die gewünschte Darstellung des Labels erzeugt. Ein `SoGroup` Modul kombiniert die eingehenden Annotationen. Des Weiteren wird das Hintergrundbild in Kombination mit den Annotationen von einem Viewer geladen und dargestellt. In dem Python Skript des Netzwerks werden vor allem die, über die GUI gesetzten, Werte in die entsprechenden Module des Netzwerkes gesetzt und upgedatet.

6.3.5.2 RelativeConfuisionMatrix

Autor: Karol Baginski

Das Modul dient der Umwandlung einer Confusion Matrix von absoluten Voxelzahlen in relative Prozentzahlen. Dies dient vor allem der komfortableren Darstellung durch das `ViewConfusionMatrix`-Modul und in Challengr, kann aber auch als Prozessierung für einen späteren Export durch das `ExportConfusionMatrix`-Modul genutzt werden. Das Module erwartet

als Input eine Confusion Matrix, wie sie programmatisch im `ComputeConfusionMatrix`-Modul definiert wurde, also als Tupel aus Zeilenreferenzen und der eigentlichen Matrix. Die einzelnen Werte der Matrix werden in Prozentwerte umgerechnet, wobei die Referenz für 100% jeweils der Referenzwert der Zeile des Eintrags darstellt. Daher ist es möglich, dass bei nicht One-Hot-enkodierten Daten Zeilensummen von über 100% auftreten können. Die Zeilenreferenzen der berechneten Matrix werden aufgrund der Angabe als Prozent auf 100 gesetzt. Die Berechnung wird durch den Button in der Moduldarstellung gestartet. Die so berechnete Confusion Matrix in relativen Werten kann daraufhin am Ausgang des Moduls verwendet werden. Sie entspricht derselben programmatischen Definition wie der Output des `ComputeConfusionMatrix`-Moduls.

6.3.5.3 ViewConfusionMatrix

Autor: Philipp Haker

Dieses Modul dient der Visualisierung einer Confusionmatrix. Die Matrix wird durch den Input eingegeben und in einer Tabelle dargestellt. Hierbei gibt es einige Darstellungsoptionen:

- Ein Haken bei „input is relativ“ kann gesetzt werden, um Matrizen zu verwenden, deren Werte relativ berechnet wurden. Ist dies der Fall, kann zusätzlich konfigurieren werden, wie viele Nachkommastellen in der Tabelle zu sehen sind und ob %-Zeichen hinter den Werten stehen sollen. Zusätzlich wird das Maximum der Werte für die Heat-Map auf 100 gesetzt, um die erwartete Funktion der Heat-Map zu gewährleisten.

- „Toggle heat-map“ schaltet die Darstellung einer Heat-Map ein bzw. aus. Die `Hot-` und `Cold`-Farben können auf der GUI eingestellt werden. Für ein natürlich wirkendes Verhältnis dieser Farben wird die `stevenssche Potenzfunktion` verwendet. Das Verhältnis ist das des Maximums der Werte in der Tabelle, im Falle einer Matrix mit Relativen Werten also mindestens 100) und dem aktuellen Zellen-Wert. Ein Verhältnis von 0 entspricht der `Cold`-Farbe, 1 der `Hot`-Farbe.

- Weiterhin kann die Darstellung der Werte in der Matrix ganz ein und aus geschaltet werden.

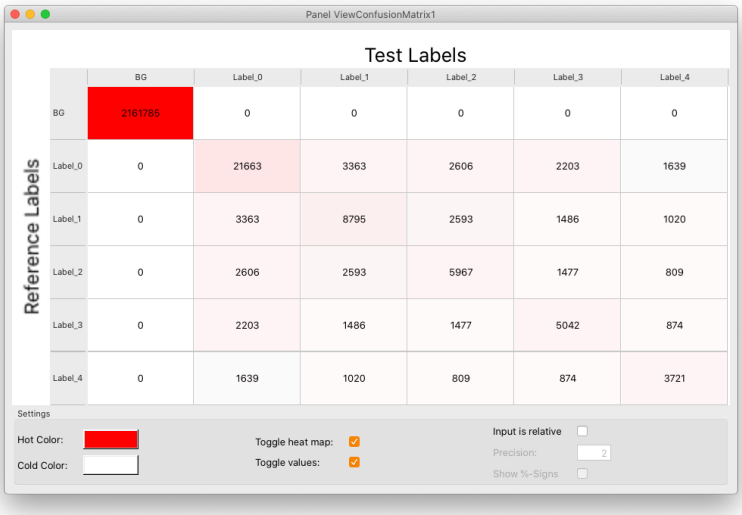


Abbildung 6.3: View ConfusionMatrix Modul, welches eine Matrix mit absoluten Werten darstellt

Abbildung 6.4: ViewConfusionMatrix Modul, welches eine Matrix mit relativen Werten darstellt

6.3.5.4 ExportConfusionMatrix

Autor: Philipp Haker

Das `ExportConfusionMatrix`-Modul dient dem Export einer Confusion Matrix als `.csv` Datei. Sind die Export-Buttons nicht anklickbar, so hilft bei der Problembehebung ein Feld oben im Modul in dem der Input-Typ angezeigt. So ist schneller zu erkennen, warum die Export-Buttons nicht klick-bar sind. Hier stehen einige Optionen zur Verfügung, die schon im `ViewConfusionMatrix`-Modul vorkommen:

- Die Angabe, ob die Eingabe relative Werte enthält. Wenn ja, so ist es möglich Prozentzeichen anzuhängen und die Dezimal-Präzision einstellen.
- Zusätzlich besteht die Option, einen benutzerdefinierten Separator zu verwenden. Der default Separator ist das Komma.

Weiterhin kann als Input für das Modul auch der Output des `ViewConfusionMatrix`-Moduls genutzt werden. In diesem Fall kann, solange das GUI Fenster des Viewers offen ist, ein View als `.png` exportiert werden. Zusätzlich können die Daten auch direkt aus der Tabelle extrahiert und als `.csv` gespeichert werden.

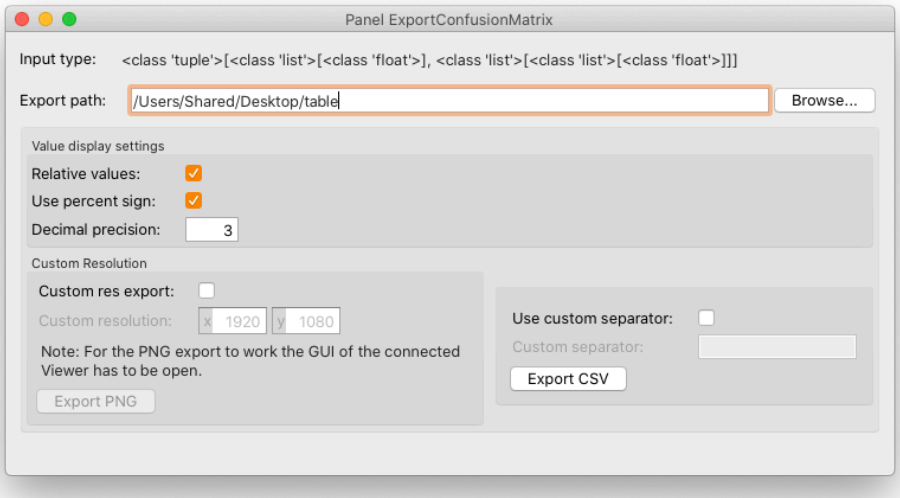


Abbildung 6.5: ExportConfusionMatrix Modul

6.3.5.5 ComputeConfusionMatrix

Autor: Jan-Gerrit Göbel

Das `ComputeConfusionMatrix` Modul berechnet eine Confusionmatrix aus zwei eingegebenen, segmentierten Bildern. Meist haben beide Bilder eine gleiche Anzahl von Labeln. Es ist aber auch möglich aus zwei Bildern mit einer unterschiedlichen Anzahl an Labeln eine Matrix zu berechnen. Die Matrix, welche vom Modul berechnet wird, liefert die Angaben in der Matrix als absolute Voxelanzahl.

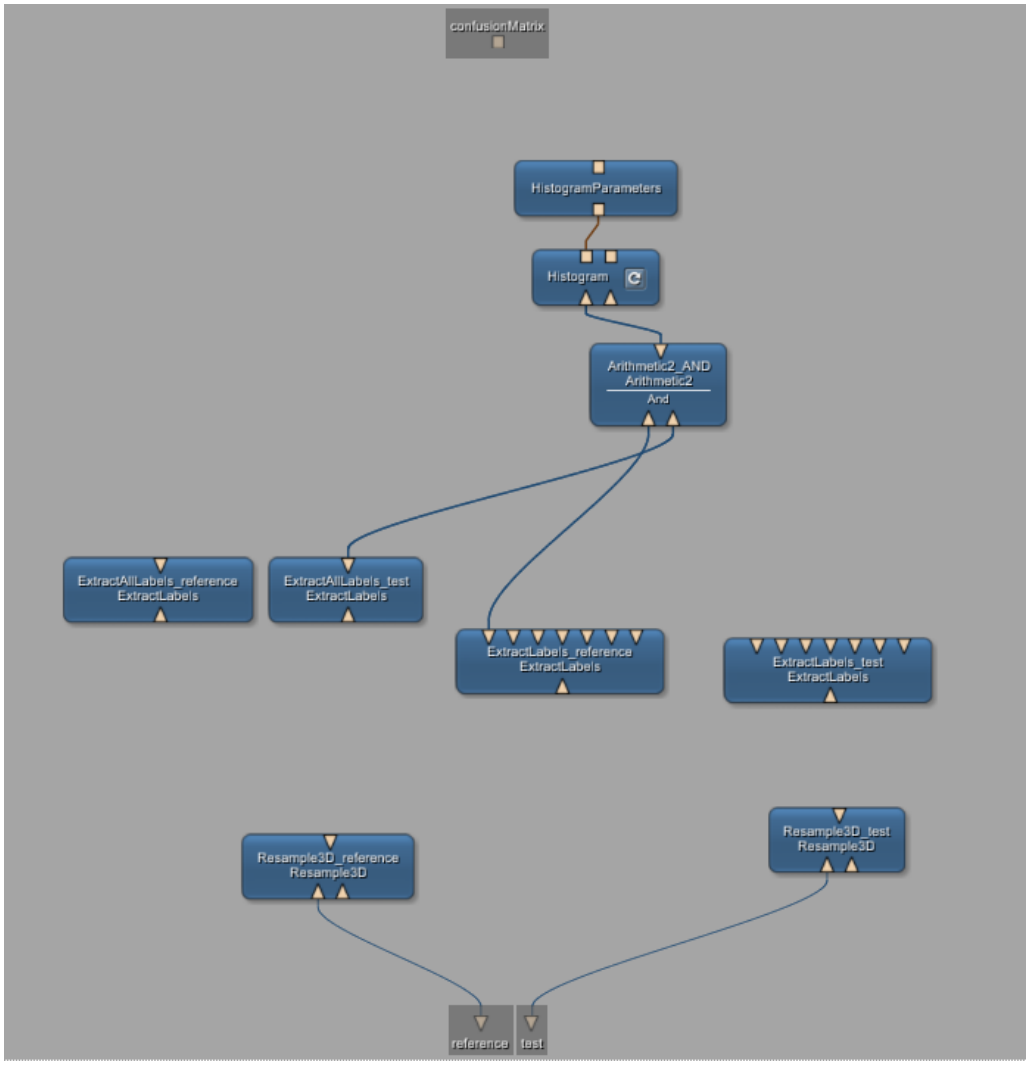


Abbildung 6.6: Internes Netzwerk des ComputeConfusionMatrix Moduls

Damit beide eingegebenen Bilder die selbe Voxelgröße haben, werden diese durch das `Resample3D` Modul auf die selbe Größe gebracht. Mit dem

`ExtractLabels` Modul werden alle Label extrahiert, mit dem `ExtractAllLabels` Modul werden alle Voxel welche kein Label besitzen maskiert, also der Hintergrund des Bildes erfasst. Somit hat `ExtractLabels` nutzbare Ausgänge in Höhe der verfügbaren Labels. Dies wird technisch begrenzt durch das `ExtractLabels` Modul auf 7 Label. `ExtractAllLabels` besitzt nur einen Ausgang auf welchem der Hintergrund maskiert ist.

Über das `Histogram` Modul in Kombination mit dem `HistogramParameters` Modul, kann von einem Label die Anzahl der dazugehörigen Voxel bestimmen.

Zuerst wird die Anzahl der Voxel im Hintergrund der beiden Bilder berechnet. Danach wird das `ExtractAllLabels_reference` Modul direkt am `Histogram`-Modul angebunden. Der Wert in intervalSum des `HistogramParameters` Modul enthält nun die Anzahl der Voxel, welche als Hintergrund klassifiziert wurden. Das gleiche wird für das `ExtractAllLabels_test` Modul wiederholt.

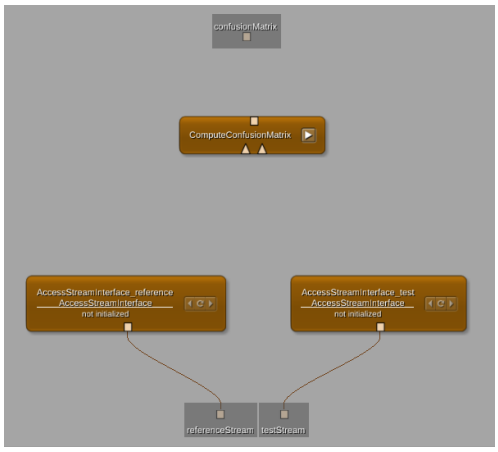
Nun kann durch eine logische Konjunktion der Schnitt zwischen zwei Masken berechnet werden. Wenn der Schnitt zwischen der Maske des Labels x des Referenzbildes und der Maske für das Label y des Testbildes errechnet wird, kommt in Kombination mit dem Histogram Modul die Anzahl der Voxel, welche eigentlich als Label x klassifiziert sein sollten, im Testbild aber als Label y klassifiziert wurden heraus. Die Anzahl der Label der beiden Bilder ist bekannt, da sie im Modul eingestellt werden muss. Nun kann über die Kombination aller Label iterieren und den Schnitt, sowie die Anzahl der Voxel in diesem berechnen, und für die Matrix speichern.

6.3.5.6 StreamComputeConfusionMatrix

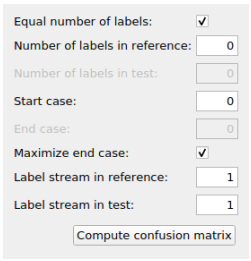
Autor: Karol Baginski

Das Modul dient der Berechnung der Confusion Matrix eines gesamten Streams. Hierfür werden die Matrizen der Fälle summiert und als neue Confusion Matrix ausgegeben. Das Modul erwartet zwei Streams als Input, jeweils für Referenz- und Testdaten. Neben den aus den Modul `ComputeConfusionMatrix` übernommenen Optionen muss noch jeweils der Stream, der die Labels enthält angegeben werden, sowie die Range der Fälle,

für die die Matrix berechnet werden soll (s. Abbildung 6.7b). Das Modul iteriert über alle gewünschten Fälle mittels der internen AccessStreamInterface-Module und nutzt das interne ComputeConfusionMatrix-Modul um die Confusion Matrix eines Falls zu berechnen (s. Abbildung 6.7a). Anschließend wird die Matrix auf die bisher berechnete Confusion Matrix summiert. Da dieser Prozess in Abhängigkeit der Dimensionen der Fälle und ihrer Anzahl merkbare Zeit benötigt, wird der Fortschritt durch einen Balken visualisiert.
Nach Abschluss der Berechnung kann die Confusion Matrix des gesamten Streams am Output des Moduls verwendet werden. Sie entspricht derselben programmatischen Definition wie der Output des `ComputeConfusionMatrix`-Moduls.



(a) Internes MeVisLabNetzwerk des StreamCompute-ConfusionMatrixModuls.



(b) Optionen des StreamCompute-ConfusionMatrix-Moduls.

6.3.6 Ergebnis

Autor: Nina Unterberg

Entstanden sind sechs MeVisLab Module, welche dazu beitragen sollen ein besseres Verständnis für die Qualität und für die Unterschiede zwischen den Annotation der Referenz- und Trainingsbildern zu entwickeln. Die Module können hintereinander in ein MeVisLabNetzwerk geschaltete werden, um den Benutzer möglichst viele Informationen bereitzustellen. Ein möglicher Anwendungsfall ist in Abbildung 6.8 dargestellt. Über die Module `ComputeConfusionMatrix` und `RelativeConfusionMatrix` wird die Confusionmatrix berechnet und durch das Modul `ViewConfusionMatrix` visualisiert. Um die in der Matrix dargestellten Zahlen nochmals besser zu visualisieren, können die entsprechenden Labels einzeln durch das Modul `LabelComparisonViewer2D` betrachtet werden.

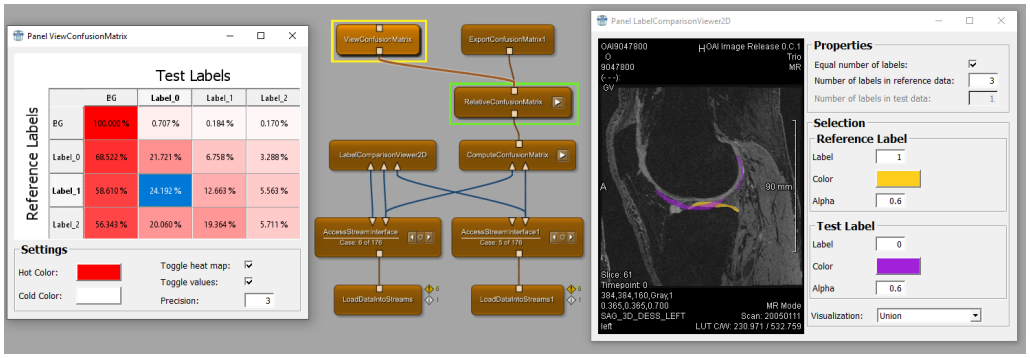


Abbildung 6.8: Ein MeVisLabNetzwerk mit einer Vielzahl von Confusionmatrix-Modulen

6.3.7 Integration in Challengr

Autor: Nina Unterberg

Challengr ist ein Tool, welches dem Benutzer dabei unterstützt den Validierungsprozess von Trainingsalgorithmen zu managen. Die Algorithmusvalidierung besteht aus folgenden Schritten:

- Aufbereitung der Eingabedaten des Algorithmus
- Aufbereitung von Referenzdaten
- Ausführung des Algorithmus auf den Eingabedaten

- Evaluierung der Ergebnisse des Algorithmus, durch Erhebung von Metriken und Kennzahlen auf den Referenzdaten
- Visualisierung der Ergebnisse und Vergleich von Evaluierungsergebnissen

Zusammenfassend kann CHALLENGR als eine Struktur betrachtet werden in der Algorithmen gespeichert und Evaluierungsergebnisse durchsucht und visualisiert werden können.

6.3.7.1 Aufbau

CHALLENGR bringt ein eigenes Command Line Interface mit. Über dieses kann beispielsweise eine Challenge erstellt oder eine Challenge im CHALLENGR Frontend angezeigt oder ausgeführt werden. Folgende Information über den Aufbau von CHALLENGR sind der CHALLENGR Hilfe-Seite [1] entnommen.
Eine angelegte Challenge enthält eine Reihe von Dateien über die diese konfiguriert werden kann. Die wichtigsten sind:

- `config.json`: Konfiguration der Challenge, beispielsweise können Trainingssätze und Referenzdaten definiert werden.
- `model.py`: Definiert eine `AlgorithmInput`, `AlgorithmResult`, `CaseMetaData`, `EvaluationResult` und `ReferenceData` Klasse.
- `AlgorithmExecution.mlab`: Ein MeVisLabNetzwerk für eine remote gesteuerte Ausführung des Algorithmus.
- `AlgorithmEvaluation.mlab`: Ein MeVisLabNetzwerk für eine remote gesteuerte Algorithmus Evaluation.
- `CaseVisualization.mlab`: Ein MeVisLabNetzwerk zum Visualisieren von Fällen, welche im CHALLENGR Frontend ausgewählt werden können.

Die Daten, welche in CHALLENGR verfügbar gemacht werden, sind in logischen Einheiten definiert (s. `model.py`). Alle diese logischen Gruppen erben von der Klasse `FieldContainer`, welche eine Menge von `Field`- Objekten halten kann. Ein `Field` definiert einen Datentyp und stellt die Logik bereit

Eigenschaften, wie z.B. den zulässigen Bereich oder den Standartwert, zu halten.

In Challengr können auch mehrere Sitzungen angelegt werden. Eine Sitzung beschreibt eine Ausführung des Algorithmus mit einer spezifischen Parametrisierung, Auswertung der Ergebnisse mit verschiedenen Referenzdaten und Ausführungen. Diese können im Frontend gegenübergestellt und verglichen werden.

Über die Challengr CLI kann ein Server gestartet werden, welcher das Challengr vue-Frontend ausführt und anzeigt. Der Server enthält Rest-API Endpunkte zum Abfragen der Challengr Daten, also beispielsweise der Sitzungen und Evaluierungsergebnisse.

6.3.7.2 Integration

Das Confusionmatrix-Modul kann an verschiedenen Stellen in Challengr integriert werden. Eine Möglichkeit ist es, das Modul zu dem MeVisLab-Netzwerk `CaseVisualization.mlab` hinzuzufügen. Wird ein Fall über das Challengr Frontend ausgewählt, wird dieser in das Netzwerk geladen. Die geladenen Test- und Referenzbilder können an das Modul *CalculateConfusionMatrix* weitergeleitet werden. Dieses berechnet die Confusionmatrix und leitet die Ausgabe an das Modul *ViewConfusionMatrix* weiter, welches die Matrix visualisiert.

Die zweite Möglichkeit besteht darin eine Confusionmatrix direkt im Challengr Frontend anzuzeigen. Bereits existierende Datentypen in Challengr können einzelne Werte und Datensätze enthalten. Allerdings ist die Visualisierung dieser Werte im Frontend auf einzelne Felder beschränkt. Eine Matrix könnte auf diese Weise nur schwer dargestellt werden. Aus diesem Grund ist ein neuer Abschnitt in dem Frontend notwendig, welcher eine Matrix darstellen kann. Um die Darstellung so generisch wie möglich zu halten, wurde die Visualisierung der Confusionmatrix auf die Darstellung einer Tabelle vereinfacht, welche alle Eigenschaften einer Confusionmatrix anzeigen kann. Zusätzlich muss beachtet werden, dass sowohl die Confusionmatrix der *current*, sowie der *compare* Sitzung gegenüber gestellt werden

können. Die Tabelle muss außerdem in der Lage sein, eine unterschiedliche Anzahl von Reihen und Spalten darzustellen.

In Challengr wurde ein neuer Datentyp `Table` erstellt, welcher folgende Parameter besitzt:

- `label: string`- Die Überschrift der Tabelle
- `defaultRowLabel: string`- Die default Bezeichnung einer Reihe
- `defaultColumnLabel: string`- Die default Bezeichnung einer Spalte
- `rowLabels: [string]`- Spezifische Beschriftungen für die Reihen
- `columnLabels: [string]`- Spezifische Beschriftungen für die Spalten
- `colorCells: bool`- Falls `true` werden die Zellen abhängig von ihrem enthalten Wert eingefärbt. Die minimal Färbung ist weiß, die maximale Färbung ist rot. Die minimal/ maximal Werte sind abhängig von dem übergebenen `range` und `optimum`.
- `range: (number, number)`- der Zahlenbereich der übergebenen Werte
- `optimum: number`- Das Optimum der Werte

Die Klasse `Table` erbt von `Field` und kann somit durch die Klasse `EvaluationResult`, welches von `FieldContainer` erbt, gehalten werden. Dies ist in Abbildung 6.9 dargestellt.

```
class Lung_GTVEvaluationResult(EvaluationResult, CaseIndex):

    per_case_confusionMatrix = Table(
        range=[0, 100],
        optimum=100,
        label='ConfusionMatrix',
        default_row_label="Label",
        default_column_label="Label",
        row_labels=["Background"],
        column_labels=["Background"],
        color_cells=True)
    per_case_table = Table(
        range=[0, 100],
        optimum=100,
        label='Tabelle',
        row_labels=["CustomRow1", "CustomRow2", "CustomRow3"],
        column_labels=["CustomColumn1", "CustomColumn2", "CustomColumn3"],
        color_cells=False)
```

Abbildung 6.9: EvaluationResults in der Datei `model.py`

In das MeVisLabNetzwerk `AlgorithEvaluation.mlab` wurde anschließend das Modul `ComputeConfusionMatrix`, sowie das Modul `RelativeConfusionMatrix` eingebaut. Die `Table`-Objekte können über das Python-Skript der `AlgorithEvaluation.mlab` mit den errechneten Werten befüllt werden.

Um die `Table`-Objekte im Frontend anzeigen zu können, muss ein API-Aufruf hinzugefügt werden, welcher die `Table`-Objekte liefert.

Abbildung 6.10 zeigt die fertige Visualisierung der in Abbildung 6.9 gezeigten `Table`-Objekte.

ConfusionMatrix	Background	Label 1	Label 2
Background	97	2	1
Label 1	18	22	60
Label 2	30	20	50

(Current)

ConfusionMatrix	Background	Label 1	Label 2
Background	97	2	1
Label 1	18	22	60
Label 2	30	20	50

(Compare)

Tabelle	CustomColumn1	CustomColumn2	CustomColumn3
CustomRow1	11	12	13
CustomRow2	21	22	23
CustomRow3	31	32	33

(Current)

Tabelle	CustomColumn1	CustomColumn2	CustomColumn3
CustomRow1	11	12	13
CustomRow2	21	22	23
CustomRow3	31	32	33

(Compare)

Abbildung 6.10: Challengr Frontend mit Tabellen

6.3.8 Fazit

Autor: Lukas Schäfer

Die in Unterabschnitt 6.3.2 beschriebenen Ziele wurden vollständig umgesetzt. Statt eines einzelnen Moduls wurden sechs kleine Module entwickelt, die einzelne Funktionen enthalten und so auch in anderen Anwendungsfällen genutzt werden können. Das in Unterabschnitt 6.3.6 beschriebene Zusammenspiel der kleinen Module entspricht den gesetzten Zielen. In Unterabschnitt 6.3.7 ist die zusätzliche Integration des Moduls in Challengr beschrieben, die über das angestrebte Ziel eines Confusionmatrix-Moduls in MeVisLabhinausgeht. Mit diesem Ergebnis wurden die ursprünglichen Ziele des Teams erfüllt und sogar übertroffen.

6.3.9 Ausblick

Autor: Lukas Schäfer, Markus Rink

Es existieren noch einige Möglichkeiten, die entwickelten Module sinnvoll zu erweitern oder zu verbessern. Beispiele dafür wären grafische Änderungen, wie das Einfügen einer anderen Heat-Map Farbe für die Hauptdiagonale. Eine Verbesserung kann ebenfalls bei der Laufzeit erreicht werden, die die Module benötigen, um eine oder mehrere Confusionmatrizen zu berechnen. Aus den Werten in der Confusionmatrix lassen sich False Positve, True Positive, False Negative und True Negative ablesen. Daraus werden viele weitere Maße, wie Accuracy oder F-Score berechnet. Werden diese in Challengr ergänzt, bieten sich mehr Möglichkeiten des Vergleichens von Ergebnissen.
Die fertigen Module sollten durch Mitarbeiter der Arbeitsgruppe geprüft und anschließend für alle Nutzer eingefügt werden. Zusätzlich zu diesem Schritt ist eine Vorstellung der Module und seiner Erweiterungen durch ein nachfolgendes Projekt denkbar.

6.4 One Hot Konvertierung

Autor: Markus Rink

6.4.1 Motivation

In neuronalen Netz für Klassifikationsproblemen werden die Ergebnisse auf verschiedene Arten kodiert. Die beiden betrachteten Kodierungen sind One-, bzw. Multi-Hot-Vektoren. Die Vektoren können verschieden repräsentiert werden, wodurch beim Vergleich von Ausgabewerten von verschiedenen Netzen eine Konvertierung nötig sein kann.
Dieses Modul bietet eine vereinheitlichte Lösung für die Konvertierung. Ein für diesen Fall spezifisches Modul, kapselt den nicht visuellen Programmieranteil, weil das erzeugte Modul die Skripte indirekt aufruft und sich so dem visuellen MeVisLab Stil anpasst.

6.4.2 Einführung

Bei einem neuronalen Netz für Klassifikationsprobleme, korrespondiert die letzte Schicht direkt mit den abzubildenden Klassen und ergibt so ein Array mit Werten zwischen 0 und 1, wobei die Werte für die Konvertierung auf oder abgerundet werden müssen. Darf das Netz dabei nur eindeutig Klassifizieren, d.h. das Maximum des Arrays wird zur 1 alle anderen 0, wird von einem One-Hot-Vektor und andernfalls von Multi-Hot-Vektor gesprochen.
Der One-Hot-Vektor lässt sich allein durch den Index der 1 eindeutig repräsentieren. Für Multi-Hot-Vektoren werden die 0/1 Werte des Arrays als Binärzahlen interpretiert, welche somit im Speicher kompakter repräsentiert werden können.
Die Abbildung 6.11 zeigt diesen Zusammenhang visuell und gibt die Limitationen des `unsigned interger 64` Datentypen an. Mit diesem ist auch das Modul auf 64 Klassen begrenzt.

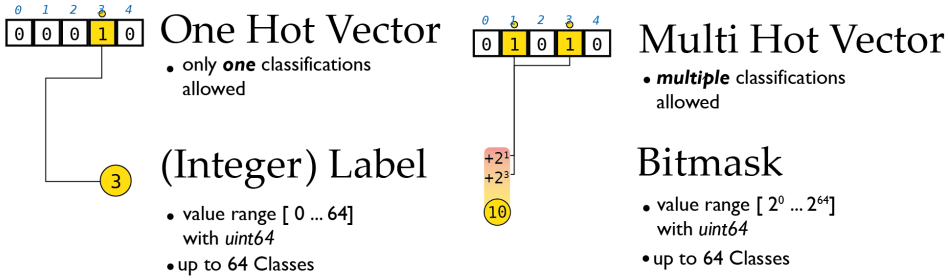


Abbildung 6.11: Konverter Modul: Konvertierungsrichtungen

6.4.3 Implementierung

Das Eingabearray `inImage` ist ein numpy array mit 6 Dimensionen, welche die MeVisLabBilddimensionen in umgekehrter Reihenfolge widerspiegeln. Ihre Form, `shape` in Numpy, ist (u, t, c, z, y, x), wobei c die Standard Dimension für Labels ist. u steht hier für *user* und soll als weitere Option als Dimension für die Konvertierung eingesetzt werden können.
Da es keine feste Bitorder gibt, wurde hier von der Bitorder Big Endian ausgegangen, da diese mit dem Array Zugriff übereinstimmt.
Die `numpy` Version von MeVisLab ist 1.15, welche die Funktion `packbits` nur in Bitorder Little Endian unterstützt, weswegen die Dimension der Label erst gedreht werden muss. Außerdem geht der Algorithmus nur bis zu 8 Labels, da nur `uint8` als Typ unterstützt wird.
Daher wurde stattdessen mit Slicing und Masken gearbeitet. Die Abbildung 6.12 fasst die möglichen Konvertierungen zusammen. Auf die einzelnen Codeabschnitte wird im Folgenden genauer eingegangen.

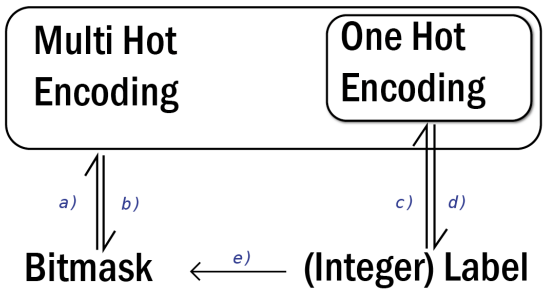


Abbildung 6.12: Konverter Modul: Konvertierungsrichtungen

a) Um aus der Bitmaske einen Vektor zu bekommen, wird zunächst eine Maske aus 2er Potenzen gebaut. Die daraus entstandene Maske hat für jeden passenden Index einen Teilvektoren mit einer 1 an dieser Stelle. Diese Maske kann durch Numpy's Broadcasting mit einer Zusatzdimension des Eingabe Bildes (`inImage`) bitweise verundet werden. Die Ergebnisse sind allerdings noch die verundeten Werte und daher in Integer Form auch größer als 1. Um den Wertebereich auf $\{0, 1\}$ zu bringen, wurde hier zu Boolean und dann zurück zu Integer konvertiert.

Das `shape` von `inImage` und `outImage` verändert sich durch die Konvertierungen und wird in `calculateOutputImageProperties()` und `calculateInputSubImageBox()` geändert.

```
1 result = []
2 try:
3     result = ((inImage[..., np.newaxis] & (1 << np.
          arange(ctx.field("numberOfLabels").value))) >
          0) * 1
4 except TypeError:
5     result = ((np.uint64(inImage[..., np.newaxis]) &
          (1 << np.arange(ctx.field("numberOfLabels").
          value, dtype=np.uint64))) > 0) * 1
6 outImage[:] = np.moveaxis(np.squeeze(result, dim), 5,
      dim)
```

Listing 6.1: Bitmaske zu Multi-Hot-Vektor

b) Da die Bitmaske im Grunde eine Konvertierung von Binärer in Dezimaler Darstellung ist, kann ein Multi-Hot-Vektor als Maske für ein Array aus 2er Potenzen genutzt werden, welches dann aufsummiert wird. Praktisch wird hier mit verschieden großen Numpy Arrays gearbeitet.
`labels` erzeugt diese 2er Potenzen und bringt es in die Form von `inImage`. Das konvertieren des Multi-Hot-Vektors in Boolean Werte, erzeugt eine Maske. Das inverse dieser Maske setzt in Zeile 3 alle Werte auf 0, welche nicht klassifiziert wurden. So werden sie von der

Summe ausgeschlossen. Übrig bleiben die klassifizierten 2er Potenzen, welche in der Summe das skalare Ergebnis bilden.

```
labels = (np.moveaxis(np.broadcast_to((2**np.arange(
    ctx.field("numberOfLabels").value)), (inImage.
    shape[:dim] + inImage.shape[dim + 1:] + (ctx.field
    ("numberOfLabels").value,) ) ), 5, dim)).copy()
mask = inImage.astype(bool)
labels[~mask] = 0
outImage[:] = np.sum(labels, axis=dim)
```

Listing 6.2: Multi-Hot-Vektor zu Bitmaske

c) Um die (Integer) Labels in One-Hot-Vektoren umzuwandeln, werden sie als Index Maske auf einer Eye Matrix verwendet. Ein Eye bezeichnet hier eine quadratische Matrix aus Nullen, mit jeweils einer Eins an den Stellen (1,1), (2,2), (3,3) usw. Die erzeugte Dimension wird hinten angehängt, weswegen sie durch `moveaxis` an die richtige Stelle geschoben wird.

```
outImage[:] =  np.moveaxis(np.squeeze(np.eye(ctx.
    field("numberOfLabels").value)[inImage], axis=dim)
    , 5, dim)
```

Listing 6.3: Label zu One-Hot

d) Der Befehl `argmax` gibt den Index des größten Elements aus einer Liste an. Hier wird nur das Maximum aus der gewählten Dimension gesucht, sodass das One-Hot Encoding in den dazugehörigen Wert übertragen wird.

```
outImage[:] = np.argmax(inImage, axis=dim)
```

Listing 6.4: One-Hot zu Label

e) Für die Konvertierung von Label zu Bitmaske werden alle Werte in ihre 2er Potenzen umgerechnet.

```
outImage[:] = np.exp2(inImage)
```

Listing 6.5: Label zu Bitmaske

6.4.4 Fazit

Das Makromodul konnte fertiggestellt und getestet werden. Im Hinblick auf das große Ziel die Trainingsschleife vereinfachen oder beschleunigen zu können ist damit ein praktisches kleines Werkzeug entstanden. Letztendlich ist es kein Teil des `ConfusionMatrix` Moduls geworden, kann aber durch seine Kompaktheit für andere Makro Module und MeVisLab Netze genutzt werden. Numpy hat durch die Maskierungs und Slicing Methoden sehr effiziente Umrechnungen ermöglicht, welche außerdem noch recht kompakt sind.

6.5 Segmentation Pipeline Optimizer

Kapitel Autor: Karol Baginski

6.5.1 Einleitung

Ein großes Problem von Segmentierungsalgorithmen stellt die Tatsache dar, dass diese, trotz ihrer guten Ergebnisse, oft hochspezifisch sind. Ein Algorithmus, der darauf optimiert wurde die Leber in bestimmten MRI-Scans zu segmentieren, kann bei CT-Scans, abweichender Auflösung und ähnlichem drastisch an Performanz verlieren. Das stellt vor allem ein Problem für den Einsatz solcher Systeme im klinischen Alltag dar, da aufwendige, manuelle Optimierungen der Algorithmen dort nicht praktikabel sind [2].

Der *Medical Segmentation Decathlon* [3] machte sich zur Aufgabe einen soliden Benchmark für die Bewertung und den Vergleich von Algorithmen zur medizinischen Bildsegmentierung zu liefern. Dabei wurde die angesprochene Problematik zum Kern der Challenge. Anstatt, wie in vielen Challenges üblich, die Algorithmen an einer spezifischen Aufgabe zu messen, werden sie an vielen, stark unterschiedlichen Aufgaben gemessen, was die Challenge besonders schwer macht. Die zehn zur Verfügung gestellten Datensätze variieren in der zu segmentierenden anatomischen Struktur, sowie ihrer Anzahl, Größe und Struktur, der Modalität der Daten, der Auflösung, der Anzahl der verfügbaren Daten, etc. Dabei ist keine manuelle Anpassung eines Algorithmus auf die aktuell zu verarbeitenden Daten erlaubt. Somit

zwingt die Challenge die Teilnehmer, das Problem der hohen Spezifität der Segmentierungsalgorithmen aktiv anzugehen.

Einen vielversprechenden Ansatz verfolgt das 2018 vorgestellte nnU-Net, das bis jetzt (Mai 2020) die Rangliste des Medical Segmentation Decathlon [4] anführt. Statt einen Algorithmus auf Basis des üblichen Ansatzes einer statischen Segmentation Pipeline aus Preprocessing, U-Net und Postprocessing zu verwenden, adaptiert der Algorithmus die Pipeline dynamisch anhand der aktuellen Daten, noch bevor das Netzwerk auf diesen trainiert wird. Dies widerspricht nicht den Anforderungen der Challenge, da diese Adaption vollautomatisch vollzogen wird. Die Wahl des Pre- und Postprocessings, sowie der Netzwerkarchitektur geschieht anhand der spezifischen Eigenschaften der aktuellen Daten, um so eine bestmögliche Grundlage für das eigentliche Training, und damit der Performanz sicher zu stellen. Dieser Ansatz ermöglicht eine große Bandbreite verschiedenster Daten zuzulassen und gleichzeitig die hohe Performanz von spezifisch optimierten Segmentation Pipelines auszunutzen [2].

Der nnU-Net Algorithmus ist nur eine Möglichkeit eines solchen Algorithmus, der auf der Prämisse basiet die Segmentation Pipeline vor dem Training anhand der Eigenschaften der Daten zu optimieren. Um die Möglichkeiten dieses Ansatzes und verschiedener dieser Algorithmen im Setting einer bestehende Infrastruktur für neuronale Netzwerke zu erforschen, bietet sich der Aufbau und die Einbindung eines Systems an, das Management und Anwendung dieser Algorithmen ermöglicht. In diesem Abschnitt soll daher das Konzept eines solchen *Segmentation Pipeline Optimizers* (SPO) dargestellt werden. Das Konzept arbeitet mit wenigen Voraussetzungen, um als Basis für viele konkrete Anwendungsszenarien dienen zu können. Die Schnittstellen zur vorhandenen Infrastruktur sind minimal gehalten und wenig spezifiziert. Des Weiteren ist es flexibel gestaltet, sodass Anpassungen in Form von zusätzlichen Funktionen und dem Abwandeln oder Entfernen vorhandener Funktionen leicht möglich sind. Die Spezifizierung und Anpassung des Basiskonzepts an eine vorhandene Infrastruktur wird beispielhaft am *Fraunhofer MEVIS* aufgezeigt. Es werden dabei keine Implementierungen vorgenommen, lediglich die möglichen Vorgehensweisen für diese skizziert.

Der Begriff *Optimierung* wird in diesem Abschnitt im Bezug auf die Op-

timierung der Segmentation Pipeline verwendet, also die Optimierung von Preprocessing, Postprocessing und Netzwerkarchitektur. Das Training eines Netzwerks, das auch eine Optimierung darstellt, wird hier exkludiert. Der Begriff Architektur bezeichnet wie üblich die Topologie des neuronalen Netzwerks, während mit Softwarearchitektur die Programmstruktur des vorgestellten Systems gemeint ist.

6.5.2 Softwarearchitektur des SPO

6.5.2.1 Konzept

Die Ausführung einer Segmentation Pipeline (kurz: Pipeline) Optimierung kann in drei unabhängige Prozesse unterteilt werden. Erstens müssen die Daten analysiert werden. Der Optimierungsalgorithmus verwendet bestimmte Eigenschaften der Daten, um abhängig von ihnen die Pipeline anzupassen. Diese Eigenschaften, der *Data Report*, müssen zunächst berechnet werden, um dem Algorithmus als Input dienen zu können. Zweitens muss der Data Report dem eigentlichen Optimierungsalgorithmus zur Verfügung gestellt werden und dieser berechnet die optimierten Parameter für die Pipeline. Drittens müssen die berechneten Parameter auf die Pipeline angewendet werden. Da das System nicht nur einen Algorithmus zur Pipeline Optimierung zur Verfügung stellen, sondern auch zum Test und Vergleich neuer Algorithmen dienen soll, muss ein Management der Algorithmen als vierter Prozess eingeplant werden, der als zentrale Kontrolleinheit dient.

Die Trennung dieser Prozesse in separate Komponenten hat den Vorteil, die Teilprozesse in gleich mehreren Dimensionen separieren zu können. Zunächst wird eine zeitliche Entkopplung möglich. Dies ist vor allem deswegen zu präferieren, da die Teilprozesse signifikant unterschiedliche Zeiten benötigen. Vor allem die Analyse kann unter Umständen sehr lange in Anspruch nehmen. Insbesondere die Analyse großen Datensätzen bis auf Voxelebene hat erheblichen Zeitbedarf. Die zweite mögliche Art der Trennung betrifft die Anwendungen die verwendet werden. Es sollte nicht davon ausgegangen werden, dass die Tools zur Analyse der Daten die selben sind, die zum Aufbau der Pipeline genutzt werden. Somit wird dem Problem vorgebeugt, dass eine Veränderung eines Tools mit der Notwendigkeit den gesamten SPO neu

zu implementieren verbunden ist. Die Minimierung der Abhängigkeit von vorhandenen Tools verhindert, dass das System ein blockierender Faktor für Veränderungen wird. Die letzte mögliche Trennung ist die infrastrukturelle. Die Komponenten müssen nicht auf dem selben Computersystem ausgeführt werden. Dies ermöglicht es Prozesse auf Serverstrukturen zu verlagern, was vor allem bei zeit- und rechenintensiven Prozessen von Vorteil ist. Es ermöglicht zusätzlich Parallelisierung und die Verwendung von spezieller Hardware zur Beschleunigung, zum Beispiel GPUs, ohne die Praktikabilität einzuschränken, sie tendenziell sogar stark zu erhöhen. Insgesamt bewirkt die strikte Trennung der Prozesse in separate Komponenten größtmögliche Flexibilität bei der Implementierung und ermöglicht so für eine Vielzahl von Infrastrukturen geeignet zu sein.

Verschiedene Optimierungsalgorithmen verwenden zu können hat zur Folge, dass die Analyse der Trainingsdaten nicht mehr statisch ist. Die Optimierungsalgorithmen haben unterschiedliche Ansprüche an den Data Report, bezüglich der konkreten Eigenschaften der Daten die für die Optimierung benötigt werden. Aufgrund des hohen Ressourcenverbrauchs der Analyse ist es ineffektiv jedem Optimierungsalgorithmus standardisiert alle möglichen Eigenschaften der Daten zur Verfügung zu stellen. Daher muss der Optimierungsalgorithmus ein *Report Scheme* zur Verfügung stellen, in dem spezifiziert wird welche Eigenschaften der Trainingsdaten für die Ausführung der Optimierung benötigt werden.

Damit ergeben sich vier Komponenten des SPO:

1. **Data Analyzer:** Dient der Analyse der Eingabedaten und Erstellung des Data Reports

2. **Optimization Manager:** Dient dem Management der Optimierungsalgorithmen, dem Verteilen verwendeter Daten und als User Interface

3. **Optimization Algorithm:** Dient der Berechnung optimierter Pipelineparameter anhand der Eigenschaften von Trainingsdaten

4. **Optimization Operator:** Dient der Anwendung optimierter Pipelineparameter auf eine Pipeline

Die Teilung des Systems in potenziell infrastrukturell getrennte Komponenten wirft die Frage auf, wie Daten zwischen ihnen ausgetauscht werden. Simple Daten, wie Zahlen oder kurze Strings, können direkt in der verwendeten Programmiersprache bzw. als Datenpakete, z.B. TCP bei infrastrukturell verteilter Implementierung, ausgetauscht werden. Für komplexere Austauschdaten, namentlich der Data Report, das Report Scheme und die Optimization, bietet sich die Verwendung von separaten Dateien an. Die Komplexität der Daten ist nur ein Faktor, der dies vorteilhaft erscheinen lässt. Eine zeitliche Trennung der Prozesse ist nur dann möglich, wenn Zwischenergebnisse sicher vorgehalten werden können. Dies erhöht zusätzlich die Ausfallsicherheit, da bereits beendete Berechnungen bei Teilausfällen des Gesamtsystems nicht wiederholt werden müssen. Ein weiterer Vorteil ist die Möglichkeit den Optimierungsalgorithmus zu bearbeiten, ohne die notwendige Analyse für einen Trainigsdatensatz wiederholen zu müssen, vorausgesetzt, dass sich das Report Scheme nicht verändert hat.

Da die Möglichkeit zur Entwicklung der Optimierungsalgorithmen ein Hauptziel des Systems ist, sollte hier die größtmögliche Praktikabilität angestrebt werden. Dasselbe gilt für die Optimization, die damit von der Pipeline auf die sie angewendet wird, entkoppelt werden kann. Für das Report Scheme ebenfalls Dateien zu verwenden, ist zum einen dem Fakt geschuldet, dass viele potentielle Formate für den Data Report, wie XML oder JSON, bereits Formate für Schemata bereit stellen. Damit steht ein größeres Softwareökosystem für die Implementierung zur Verfügung. Zum anderen trennt es so Optimierungsalgorithmus und sein Schema. Dies ermöglicht größere Flexibilität bei der Auswahl der verwendeten Sprachen. Insbesondere bieten für Algorithmen verwendete Programmiersprachen oft nicht die Möglichkeit einer übersichtlichen Definition eines Schemas. Hier sind Skriptsprachen, wie Python zwar im Vorteil, dennoch bietet sich auch hier aus Gründen der Übersichtlichkeit eine Trennung von Algorithmus und assoziiertem Report Scheme in zwei Dateien an.

Damit ergeben sich drei Dateiformate, die dem Austausch komplexer Daten zwischen den Komponenten dienen:

1. **Report Scheme:** Definiert die für die Berechnung des assoziierten Optimierungsalgorithmus benötigten Eigenschaften der Trainingsda-

ten

2. **Data Report:** Enthält bestimmte Eigenschaften eines Trainingsdatensatzes

3. **Optimization:** Enthält die optimierten Parameter der Segmentation Pipeline

Das UML Komponenten Diagramm (Abbildung 6.13) fasst die Konzeption des Systems auf Ebene der Komponenten zusammen.

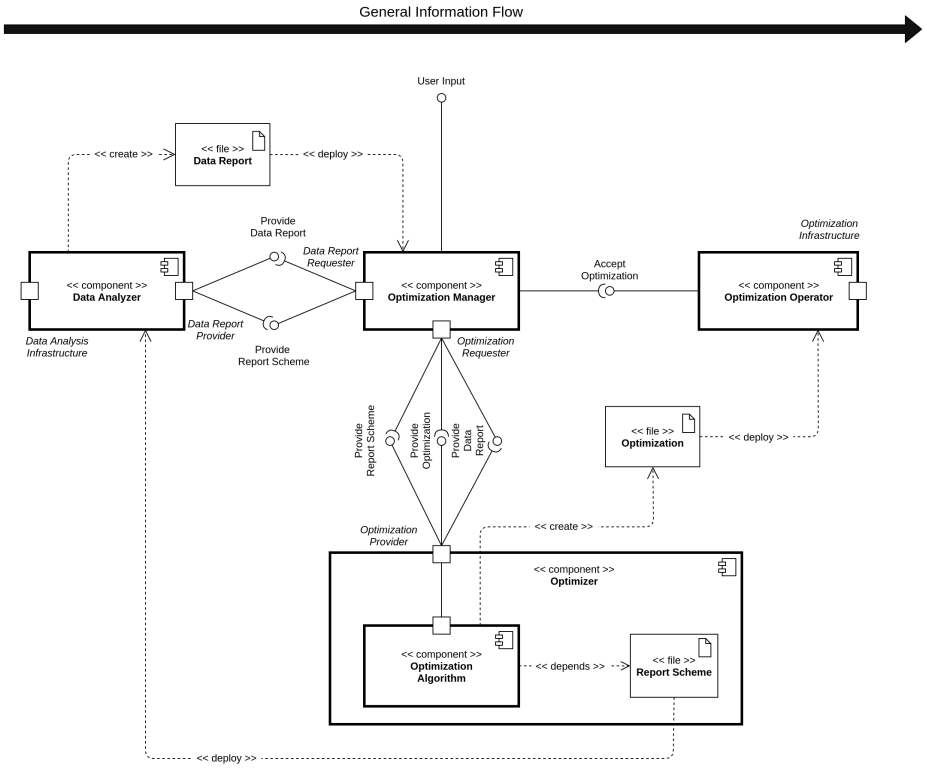


Abbildung 6.13: Komponenten Diagramm des Segmentation Pipeline Optimizers.

Die Formate der Dateien müssen einige Voraussetzungen erfüllen, um im System einsetzbar zu sein. Zunächst muss eine endliche Menge I von möglichen Analysewerten, den *Report Items* (kurz: Items), festgelegt werden. Diese repräsentieren einzelne Eigenschaften der Trainingsdaten, wie Anzahl der Voxel, Median der Voxelwerte, ob die Trainingsdaten anisotrop sind, etc. Dabei sollte die Aufnahme eines Items zu I davon abhängen, wie hoch der Informationsgehalt ist, ob er sinnvoll in Optimierungsalgorithmen eingesetzt werden kann, wie oft er verwendet wird und ob er sich nur um-

ständlich bzw. gar nicht durch Kombination anderer Items berechnen lässt. Jedem Item wird eine Domäne zugewiesen aus der seine Werte stammen. Die Items sollten des Weiteren parametrisierbar sein, da Items wie Quantile nur mit Parametern sinnvoll und übersichtlich verwendet werden können. Ein Report Scheme stellt dann eine Teilmenge aller Items I und eine Zuweisung der Items zu konkreten Parameterwerten dar. Ein Data Report stellt ein Report Scheme dar, in dem jedem Item nicht nur passende Parameterwerte, sondern auch jedem Item selbst ein passender Wert zugewiesen wird. Formal lässt sich dies so definieren[1]:

Grundmengen:

$\mathcal{I}$ (endliche Menge der Report Items)
$\mathcal{P}$ (endliche Menge der Parameter)
$\mathbb{D} = \{Integer, Boolean, \dots\}$ (endliche Menge der Wertebereiche)

Zuweisung von Wertebereichen ($\mathcal{O}$ s.u.):

$\delta : \ (\mathcal{I} \cup \mathcal{P} \cup \mathcal{O}) \to \mathbb{D}$

Zuweisung von Parametern zu Items:

$\phi : \ \mathcal{I} \to \ \bigcup_{i \in \mathbb{N}_0} \{f \mid f : \ [n] \to \mathcal{P}\}$

Definition eines Report Schemes S:

$S \ := \ (I, w_P)$,
$I \subseteq \mathcal{I}$,
$w_P : \ I \to \ \bigcup_{n \in \mathbb{N}_0} \{f \mid f : \ [n] \to \bigcup_{D \in \mathbb{D}} D\}$,
$\forall i \in \mathcal{I}, p \in \mathcal{P} : \ w_P(i)(p) \in \delta(\phi(i)(p))$,
$\forall i \in \mathcal{I} : \ dom(w_P(i)) = dom(\phi(i))$

Definition eines Data Reports R:

$R \ := \ (I, w_P, w_I)$,
(I, w_P) ist ein Report Scheme,
$w_I : \ I \to \bigcup_{D \in \mathbb{D}} D$,
$\forall i \in \mathcal{I} : \ w_I(i) \in \delta(i)$

[1] $[i] \ := \ \{n \in \mathbb{N} \mid n \leq i\}$
$dom(f) = A, \ img(f) = B \mid f : \ A \to B$

Bei Überladung der Items wird ϕ zu einer Multifunktion.

Zudem kann noch die Eigenschaft von Data Reports und Schemes definiert werden in einer Teilbeziehung $\leq$ zu stehen, da diese Eigenschaft dafür genutzt wird die Verwendbarkeit unspezifischer Data Reports für einen anderen Algorithmus zu prüfen. Eine Report oder Scheme ist ein Unterreport eines anderen Schemes oder Reports, wenn jedes Report Item und seine Parameterwerte auch im anderen vorhanden sind. Seien $R_n = (I_n, w_{P,n}, w_{I,n}),\ n \in \{1,2\}$ Data Reports und dementsprechend $S_n = (I_n, w_{P,n})$ Data Schemes. Dann ist X_1 Teilschema von $X_2,\ X \in \{S, R\}$ geschrieben $X_1 \leq X_2$, wenn[2]:

$$\sigma(X) := \begin{cases} (I, w_P) & \mid X = (I, w_p, w_I) \\ (I, w_p) & \mid X = (I, w_p) \end{cases} \qquad \text{(Schemafunktion)}$$

$$X_1 \leq X_2 \ :\Leftrightarrow\ \sigma(X_1) \leq \sigma(X_2)$$

$$(I_1, w_{P,1}) \leq (I_2, w_{P,2}) \ :\Leftrightarrow\ I_1 \subseteq I_2 \ \wedge\ w_{P,1} = w_{P,2}|_{I_1}$$

Ein ähnliche Definition muss für die Optimization festgelegt werden. Es müssen Optimization Items mit Wertebereich festgelegt werden. Zusätzlich müssen den Items Standardwerte zugewiesen werden, um Teiloptimierungen der Parameter leichter vornehmen zu können. Eine erzeugte Optimization entspricht dann einer Zuweisung einer Teilmenge der Optimization Items zu geeigneten Werten. Die eigentliche Optimierungsfunktion, entspricht dann der Kombination der berechneten Parameter mit der Standardfunktion, wobei explizite Werte in der Optimization, implizite Werte aus der Standardfunktion überschreiben. Dies lässt sich formalisieren:

$\mathcal{O}$ (Menge der Optimization Items)

Definition der Standardfunktion $w_{default}$:

$w_{default} :\ \mathcal{O} \to \bigcup_{D \in \mathbb{D}} D,$
$\forall i \in \mathcal{O} :\ w_{default}(i) \in \delta(i)$

Definition einer Optimization O:

[2] $f|_A = A \to img(f),\ x \mapsto f(x) \mid A \subseteq dom(f)$

$O := (O, w_O)$,
$O \subset \mathcal{O}$,
$w_O : O \to \bigcup_{D \in \mathbb{D}} D$
$\forall i \in \mathcal{O} : \; w_O(i) \in \delta(i)$

Definition der Optimierungsfunktion Ω ($W_{default}$ Menge der Standardfunktionen, W_O Menge der Optimizationfunktionen):

$$\Omega : W_{default} \times W_O \to \{f \mid f : \mathcal{O} \to \bigcup_{D \in \mathbb{D}} D\}$$
$$\Omega(w_{default}, w_O)(i) \mapsto \begin{cases} w_O(i) & \mid i \in dom(w_O) \\ w_{default}(i) & \mid i \notin dom(w_O) \end{cases}$$

Trotz der Vagheit der Definition, die auch ihre Flexibilität garantiert, sollten bestimmte Items immer zu Data Report und Optimization gehören, da diese der Nachvollziehbarkeit und dem Management der Dateien dienen:

- ID zur eindeutigen Identifikation
- Zeitstempel (kann als ID verwendet werden)
- Name des Users
- Textuelle Beschreibung
- Optimierungsalgorithmus der Anfrage

Aus den bisher etablierten Eigenschaften des SPO lässt sich bereits der Ablauf einer typischen Pipeline Optimierung ableiten. Der User wählt einen Optimierungsalgorithmus aus und optional einen bereits vorhandenen Data Report. Sollte kein Data Report angegeben werden, muss dieser berechnet werden. Bei expliziter Angabe eines Reports, muss überprüft werden, ob dieser spezifisch für den ausgewählten Algorithmus erstellt wurde. Falls nicht ist es weiterhin möglich, dass der vorhandene Bericht ein Oberreport des benötigten Reports darstellt. Zur Prüfung wird das Report Scheme vom gewählten Algorithmus angefordert. Sollte der angegebene Report dennoch nicht geeignet sein, muss ebenfalls ein passender Data Report erstellt werden. Sollte ein neuer Data Report angefertigt werden, können aufgrund der Möglichkeit der Parallelisierung weiter Optimierungsaufträge aufgegeben werden. Die Analyseanfrage erzeugt einen spezifischen Data Report

in Abhängigkeit vom Report Scheme. Auf Basis dieses Reports wird anschließend eine Optimierung vom gewählten Optimierungsalgorithmus angefertigt, die dann auf eine Pipeline angewendet werden kann. Das UML Aktivitätendiagramm (Abbildung 6.14) fasst diesen Ablauf zusammen.

Zusammen mit den bereits etablierten Komponenten des SPO und ihrer Funktion, den Dateien, die zum Austausch komplexer Daten verwendet werden und dem Ablauf einer typischen Optimierung lässt sich letztere detaillierter Ausarbeiten. Bezüge zu den implementierten Funktionen werden in Klammern dargestellt. Der User stellt eine Anfrage an den Optimization Manager über eine Optimierung (`optimize`) einer Pipeline (`operator`), mittels eines bestimmten Algorithmus (`algorithm`) und auf Basis der Eingabedaten(`data`). Aufgrund der Notwendigkeit einen Data Report anzufertigen, fordert Optimization Manager vom gewählten Optimierungsalgorithmus das benötigte Report Scheme an (`requiredReportScheme: reportScheme`). Daraufhin stellt der Manager eine Anfrage an einen Data Analyzer über die Anfertigung des benötigten Data Reports (`requestDataReport`). Dieser benötigt neben dem zuvor erhaltenen Report Scheme (`reportScheme`) und den zu analysierenden Daten (`data`), eine Identifikationsmöglichkeit des anfragenden Optimization Managers (`requester`) zur Notifikation und eine ID zur Identifikation des zu liefernden Data Reports. Die Berechnung des Reports verwendet die Inputdaten, das Report Scheme und die zur Berechnung der einzelnen Items verwendete Analyseinfrastruktur. Nach Anfertigung des Reports (`dataReport`) wird dieser an den anfragenden Manager ausgeliefert (`notifyDataReportReady`). Die Anfrage und Auslieferung eines Reports findet asynchron statt, um die nötige zeitliche und infrastrukturelle Trennung gewährleisten zu können. Der erhaltene Data Report wird nun vom Optimization Manager dem gewählten Optimierungsalgorithmus zur Verfügung gestellt und eine Optimierung auf dessen Basis angefordert (`requestOptimization`). Nach der Anfertigung wird der Manager darüber informiert (`notifyOptimizationReady`). Auch dies findet aus genannten Gründen asynchron statt. Die erzeugte Optimierung kann daraufhin dem anfangs gewählten Optimization Operator übergeben (`acceptOptimization`) und mittels der Optimierungsinfrastruktur auf die Pipeline angewendet werden. Das UML Sequenzdiagramm (Abbildung 6.15) stellt den detaillierten Ablauf zusammenfassend dar.

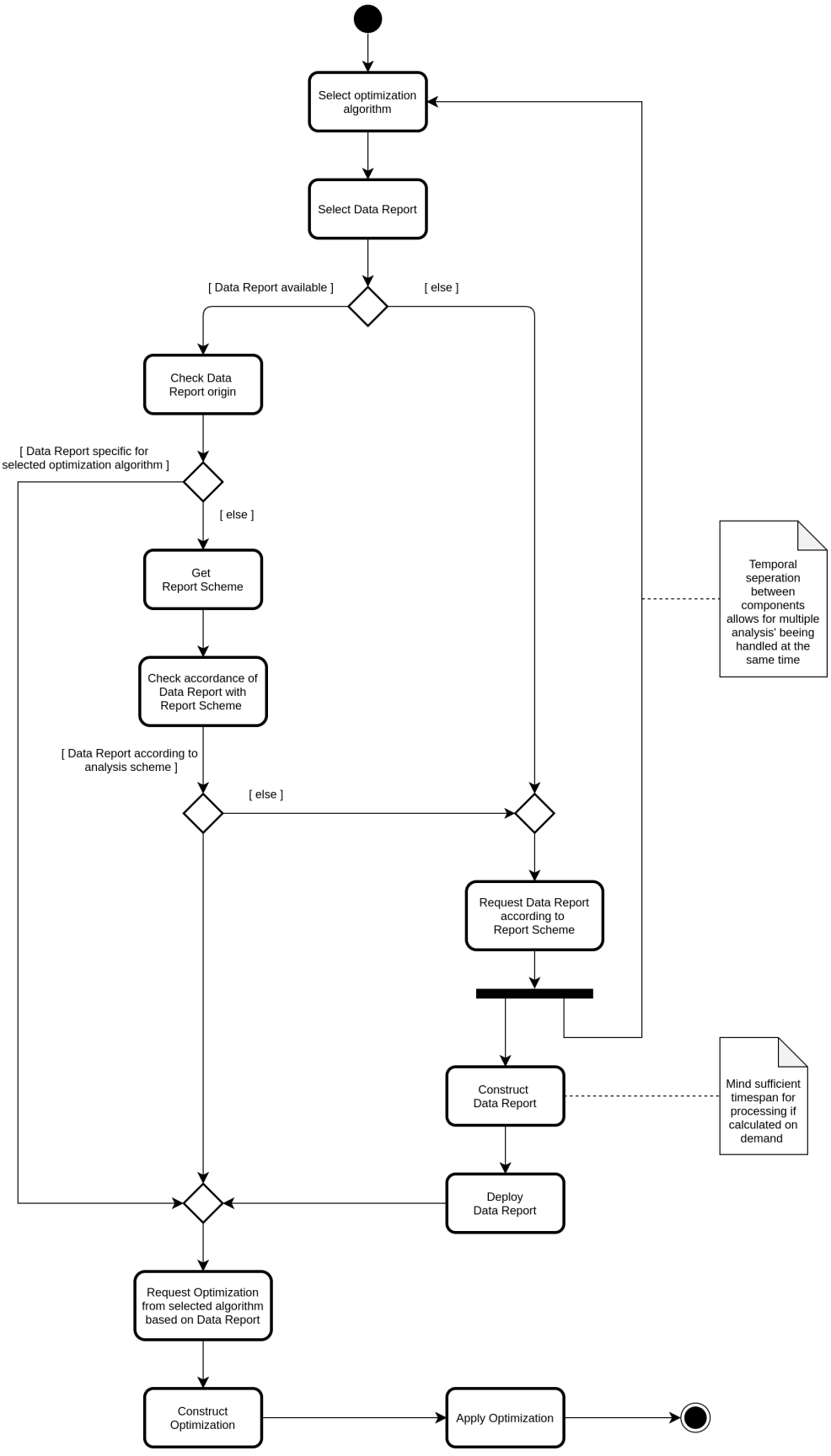


Abbildung 6.14: UML Aktivitätendiagramm einer Segmentation Pipeline Optimierung.

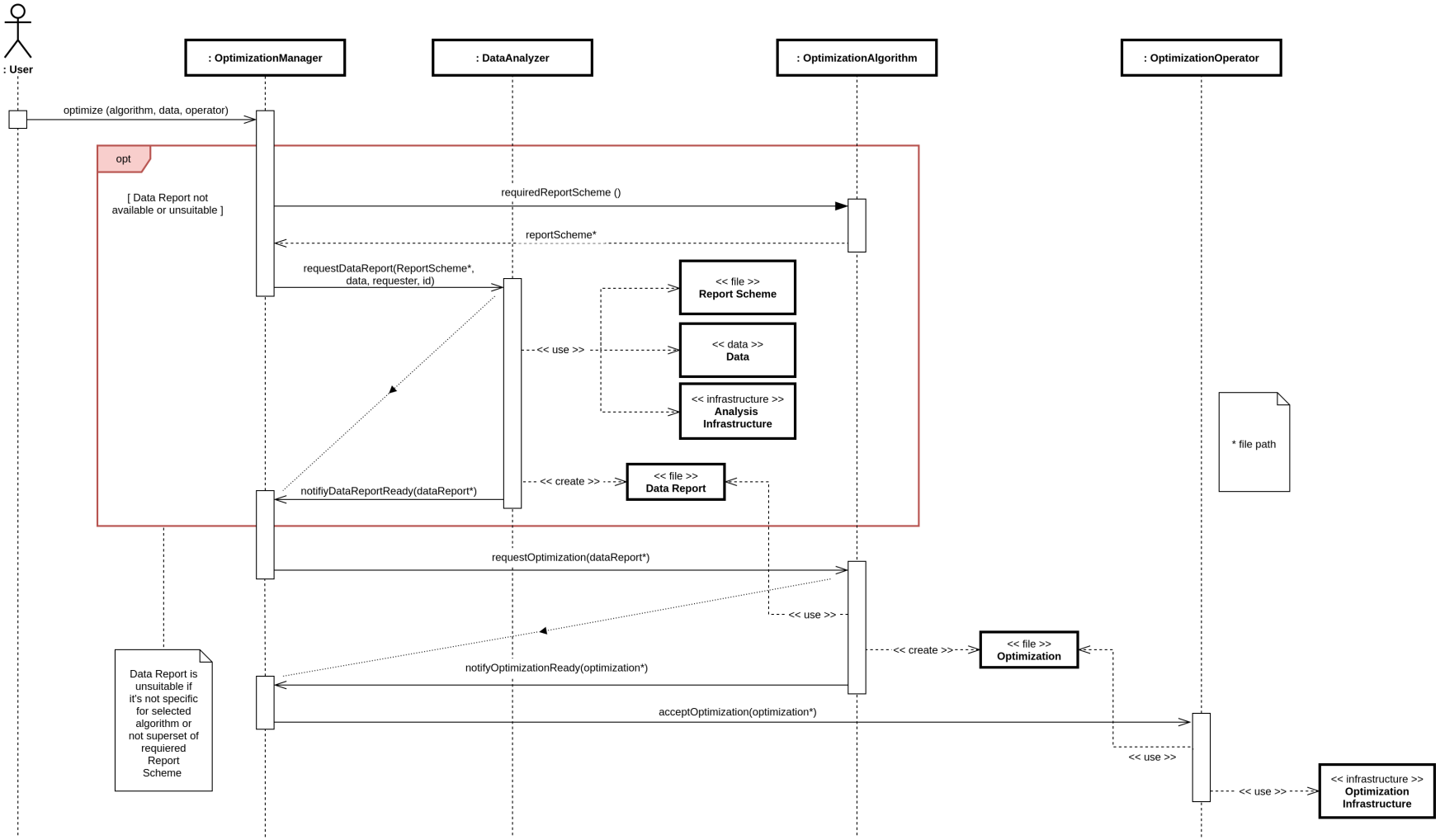


Abbildung 6.15: UML Sequenzdiagramm einer Segmentation Pipeline Optimierung.

Auf Basis der gesammelten Konzeptionen lässt sich das UML Klassendiagramm (Abbildung 6.16) anfertigen. Dieses zeigt bereits ein großes Maß an Details und ist daher mit einem gewissen Abstand zu betrachten. Das Diagramm stellt die Komponenten als abstrakte Klassen dar. Die genaue Bedeutung kann, je nach Implementierungssprache, variieren. So könnten die Definitionen ebenfalls auch als Interfaces angesehen werden. Ebenfalls kann sich die konkrete Bedeutung bei den Komponenten unterschieden, da diese nicht notwendigerweise in derselben Sprache implementiert werden müssen. Bereits hier zeigen sich, je nach vorhandener Infrastruktur und gewünschter Implementierung, große Unterschiede. Besonders bei einer infrastrukturellen Trennung der Komponenten sind die Kommunikationsbeziehungen nicht als direkte Verbindung zwischen Klassen eines einheitlichen Programms anzusehen. Daher wird auf die detaillierte Darstellung der einzelnen Funktionen verzichtet. Optimization Operator und Data Analyzer sind prinzipiell in variabler Anzahl angelegt und müssen beim Manager an- und abgemeldet werden. Dementsprechend werden sie in geeigneten Datenstrukturen, wie Arrays oder Listen, gespeichert. Die Optimierungsfunktion ist überladen, um verschiedene mögliche Anwendungsfälle abzudecken, z.B. die Optimierung mit und ohne bereits vorhandenem Data Report. So werden auch die öffentlichen Funktionen zur Prüfung von Teilbeziehungen zwischen Reports und Schemes und der Spezifität von Reports zur Verfügung gestellt. Optimization Algorithms sind gezwungen ein Report Scheme zur Verfügung zu stellen.

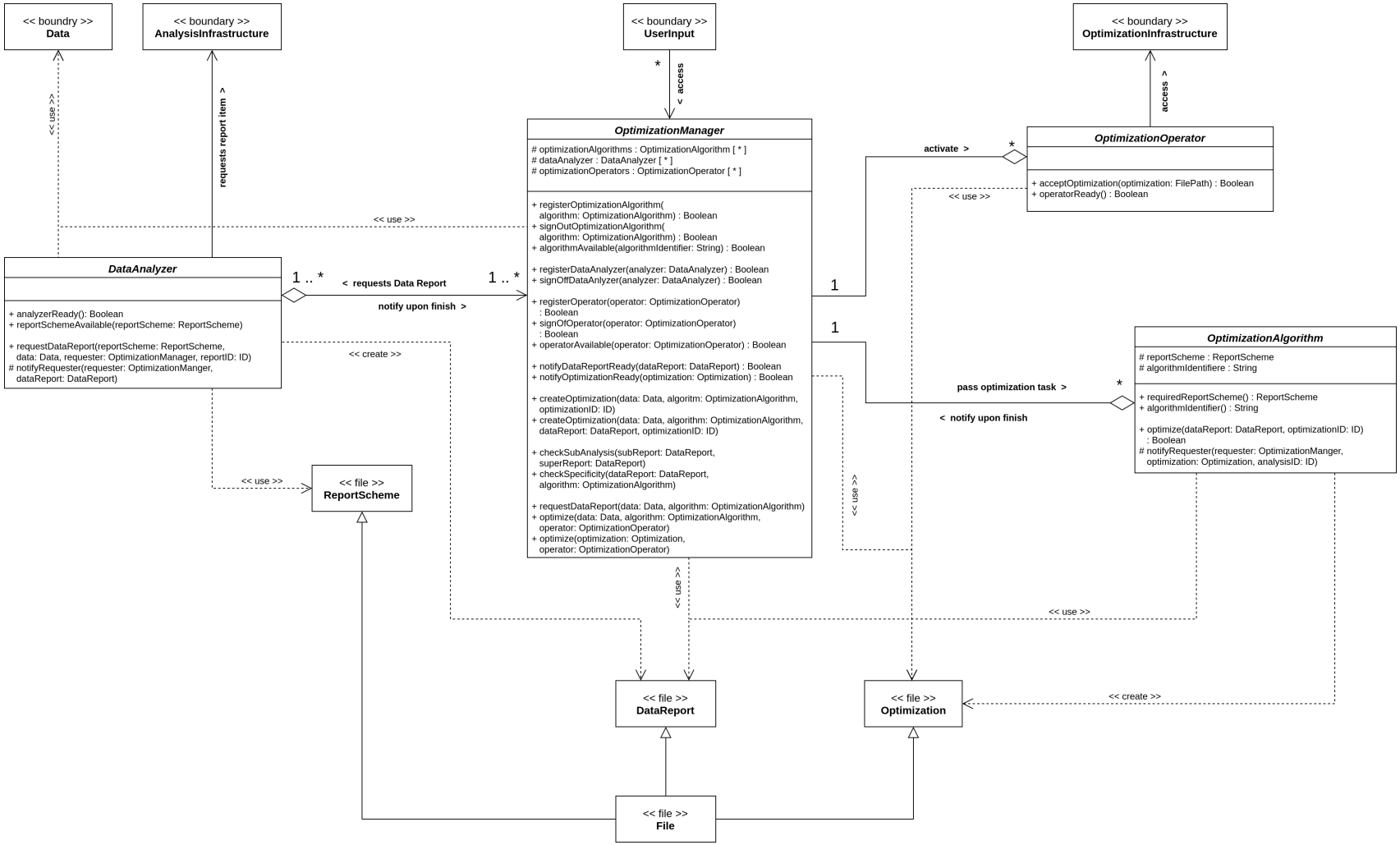


Abbildung 6.16: UML Klassendiagramm des Segmentation Pipeline Optimizers.

6.5.2.2 Umsetzung bei MEVIS

Die für eine Umsetzung verwendete Infrastruktur beim Fraunhofer MEVIS basiert zu ein großen Teil auf der Anwendung *MeVisLab* [5]. Diese stellt eine graphische Umgebung zum Laden und Darstellen vornehmlich medizinischer Bilddaten und der Erstellung von Prozessierungspipelines zur Verfügung. Die Verarbeitung erfolgt in *Modulen* mit definierten Input und Output, die zu komplexeren Verarbeitungsstrukturen verbunden werden können. Obwohl MeVisLabauch als Standalone-Anwendung zum Einsatz kommt, wird eine speziell für die Bildverarbeitung und das Training Neuronaler Netzwerke ausgestattete Serverstruktur zur Verfügung gestellt. Aufgrund der Effizienz dieser, gegenüber der lokalen Verwendung von MeVisLab, sollte diese ebenfalls für die Implementierung des SPO zum Einsatz kommen. Dabei ist durch die hohe Flexibilität von MeVisLabbezüglich des Hostsystems der lokale Einsatz nicht ausgeschlossen, lediglich als ineffizient anzusehen.

Das Starten eines Trainings auf dem Server benötigt, neben den Trainingsdaten, drei Inputs. Erstens ein MeVisLab-Netzwerk, das den Rahmen der Pipeline bildet. Zweitens eine Architekturdatei zur Definition der Parameter des Neuronalen Netzwerks. Drittens eine Konfigurationsdatei als zusätzliche Möglichkeit zur Manipulation von Netzwerkparametern. Eine naheliegende Möglichkeit der Implementierung der Komponenten ist, sie als MeVisLab-Module zu konzipieren. Diese müssen noch vor dem Training gestartet werden und können so das Trainingsnetzwerk und die assoziierten Dateien anpassen. Ein weitere Möglichkeit wäre die Implementierung als separater Dienst der Serverstruktur. Dies hätte den Vorteil, die Analyse parallelisieren zu können, indem die Report Items als separate Tasks verteilt werden. Der SPO könnte durch eine Erweiterung des bestehenden QuantMed-Systems angesprochen werden, das momentan zum Starten von Traingjobs auf dem Rechencluster genutzt wird. Dies hat den Vorteil kein neues UI etablieren zu müssen. Die Pipelineoptimierung wäre lediglich eine neue Option und würde so bessere Useability garantieren. In diesem Falle sollte es leicht möglich sein, Optimierungsalgorithmen zur Auswahl des SPO hinzuzufügen, z.B. durch explizite Angabe eines Pfads, Hochladen neuer Algorithmen oder Zugriff auf einen standardisierten Ordner zur Ablage der Algorithmen und der assoziierten Dateien. Eine Implementierung

in der beim Fraunhofer MEVIS weitläufig genutzten Skriptsprache Python kann den flexiblen Umgang mit den Optimierungsalgorithmen erleichtern. Die Berechnung der Report Items sollte aufgrund der allgemein besseren Performanz weiterhin in Instanzen von MeVisLaberfolgen.

Die Anpassung der Pipeline Parameter erfordert eine Pipeline, die einer geeigneten Struktur entspricht und das Anwenden der Optimierungsfunktion Ω ermöglicht. Da beim Fraunhofer MEVIS nicht mit statischen Pipelines gearbeitet wird, sondern gerade die große Variabilität, die mittels MeVisLaberreicht werden kann einen großen Vorteil darstellt, muss diese Variabilität in den SPO aufgenommen werden. Zwei Varianten sind hier als sinnvoll zu erachten. Die Erste besteht in einer zusätzlichen Teilkomponente des Optimizers (vgl. 6.13). Ähnlich dem Report Scheme wird jedem Optimierungsalgorithmus ein *Pipeline Scheme* zugeordnet, das eine Spezifikation der für den Algorithmus zulässigen Pipelines darstellt. Diese Variante ermöglicht die hohe Variabilität der Pipeline beizubehalten und gleichzeitig die Möglichkeiten der Trainingsdaten basierten Optimierung auszunutzen. Beim Test, der Entwicklung und vor allem dem Vergleich dieser Optimierungsalgorithmen kann diese hohe Variabilität der Pipelines allerdings auch die Beurteilung unnötig komplex werden lassen. Daher besteht die zweite Variante in der Implementierung einer wenig spezifischen Standardpipeline, die als Grundlage jedes Optimierungsalgorithmus dient und so zu einer besseren Vergleichbarkeit führt. Das in Abschnitt 5.9 beschriebene Modul kann als Basis für solch eine Standardpipeline dienen. Beide Ansätze schließen sich nicht aus, da eine Standardpipeline auch als von vielen Optimierungsalgorithmen gemeinsam genutztes Pipeline Scheme angesehen werden kann. Beide Ansätze haben ihre Berechtigung, je nachdem ob Beurteilung der Algorithmen oder bestmögliche Performanz der Segmentierung im Fokus stehen.

Für die Implementierung des Data Schemes, des Data Reports und der Optimization bietet sich, aufgrund der bereits vorhandenen Schnittstellen und der ohnehin breiten Verwendung, Python an. Die Möglichkeiten der Kodierung der Daten sind vielfältig. Eine einfache, variable und dennoch für den User leicht verständliche soll hier vorgestellt werden. Items, egal ob Report oder Optimization, werden als Variablen dargestellt. Dies ermöglicht sie übersichtlicher aufzuschreiben, im Gegensatz zur Darstellung als Einträge

Report Scheme:
```
author = None
quantil =  "value": 0.3
anisotropicity = None
...
```

Data Report:
```
author = "Ada Lovelace"
quantil = ( "value": 0.3 , 397)
anisotropicity = (None, True)
...
```

Optimization:
```
networkDimensionality = "2D"
cnnLevels = 4
batchSize = 42
...
```

Abbildung 6.17: Beispiel der Struktur der Austauschdateien des SPO bei Fraunhofer MEVIS.

eines Dictionaries. Da Python zu den interpretierten Sprachen gehört, können die Variablen dennoch problemlos ausgelesen und verwendet werden. Parametrisierten Items wird im Report Scheme ein Dictionary ihrer Parameter als Wert zugewiesen. Ein Tupel würde die Parameter unleserlicher kodieren, da kein Begriff zu ihrer Identifizierung vorhanden ist. Nicht parametrisierten Items wird `None` zugewiesen. Ein Data Report enthält immer sein zugrunde liegendes Scheme (vgl. Unterunterabschnitt 6.5.2.1). Daher wird den Variablen im Report ein 2-Tupel zugewiesen, das als erste Komponente die Parameter enthält und als zweite den Wert des Items. Die Optimization enthält lediglich die Items als Variablen denen ihr Wert zugewiesen ist. Abbildung 6.17 gibt ein Beispiel für die Kodierung.

6.5.3 Diskussion

Die hier vorgestellte Softwarearchitektur des Segmentation Pipeline Optimizers stellt eine allgemeine Basis für die Umsetzung eines solchen Systems dar. Die gezielte Fokussierung auf eine flexible, abstrakte Grundstruktur erwies sich als vorteilhaft. Die wenigen, aber begründeten Grundannahmen auf denen sie basiert ermöglichen der Implementierung sehr viel Spielraum

und maximiert so die Anzahl an Infrastrukturen für die sie in Frage kommt. Sie ermöglicht die zeitliche, infrastrukturelle und prozesshafte Trennung der Komponenten und sogar ihrer Teilaufgaben, z.B. der Berechnung einzelner Report Items im Data Analyzer. Auch die Multiplizitäten sind so gewählt, dass durch die Verwendung mehrerer Data Analyzer oder der internen Verteilung auf mehrfach vorhandene Subkomponenten im Data Analyzer, vor allem bei der Berechnung on-demand ein Hochskalieren der Analyseleistung leicht möglich ist. Hierfür ist keine Änderung des Grundkonzept notwendig. Trotz der Flexibilität ist die Architektur koherent genug, um eine klare Struktur vorzugeben, mit der Schnittstellen für die praktische Arbeit eindeutig spezifiziert werden können.

Die Flexibilität des Systems führt allerdings auch zu mehr Unklarheit bei einer konkreten Implementierung. Dies zeigte sich bereits bei der Ausarbeitung des abstrakt gehaltenen Klassendiagramms und noch viel stärker bei der skizzierten Umsetzung bei Fraunhofer MEVIS. Selbst die Definition der Kodierung des Report Schemes zeigt eine Fülle an Möglichkeiten. Diese müssen bei der Implementierung sehr genau abgewogen werden, um sich bezüglich der Useability, dem Anschluss an vorhandene Systeme und der Portabilität in die bestehende Infrastruktur und die üblichen Abläufe einzufügen.

Eine abschließende Ausarbeitung einer konkreten Umsetzung bei Fraunhofer MEVIS war im Rahmen dieser Ausarbeitung leider nicht möglich. Daher sind die erbrachten Konkretisierungen der Implementierung lediglich als Skizze anzusehen, die nur der beispielhaften Übertragung der abstrakt definierten Grundlagen auf die tatsächliche Anwendung dienen. Hier sind viele Varianten der Details denkbar, die anhand konkreter Eigenschaften der Infrastruktur und des Workflows definiert werden sollten.

6.5.4 Fazit

Der vom nnU-Net geführte Beweis, dass die automatische Optimierung einer Segmentation Pipeline zu einer signifikanten Performanzsteigerung führen kann [2], ist als wichtiger neuer Ansatz zu betrachten. Um so wichtiger ist die Aufgabe eine Forschungsumgebung für diesen Ansatz zu schaffen, die

den Fokus auf die Optimierungsalgorithmen selbst erlaubt und sich gleichzeitig in eine bereits etablierte Forschungsinfrastruktur einfügt. Hier kann die vorgestellte Softwarestruktur als eine wichtige Basis für die Entwicklung einer solchen Umgebung dienen.

6.6 Fazit

Autor: Lukas Schäfer

Die Ziele, die in Abschnitt 6.2 gesetzt wurden, konnten erreicht werden. Es wurden sieben Module erstellt, die es den Nutzern von MeVisLab erleichtern, ein trainiertes neuronales Netz zu analysieren. Außerdem wurde ein Konzept entworfen, das bei der Einführung des nnU-Net Konzepts zur automatischen Optimierung einer Segmentation Pipeline bei MEVIS helfen soll.

6.7 Ausblick

Autor: Markus Rink

Die Visualisierung der Differenz durch das Confusion Matrix Modul zeigt die Fehlerbereiche Regional Abbildung 6.2b. Berechnet man diese Masken mit probabilistisch Netzen, kann man über den Durchschnitt verschiedener Prädiktionen die Fehlerbereiche als eine Art Unsicherheit des Netzes interpretieren (Uncertainty Sampling).

Aus Sicht des Aktiven Lernens, sollte man diese Maße schon zwischen den Trainingsintervallen als Feedback bekommen. So könnte z.B. mit Hard Sample Mining das nächste Trainingsset aufgrund des Feedbacks ausgewählt werden.

In SATORI werden bereits Label mit Maskierungs Tools gezeichnet. Ein Benutzer könnte damit anstelle von Label Masken, Regionen besonderer Priorität einzeichnen. Die ausgewählten Voxel sind Teil von Patches, die dann Teil des nächsten Trainingsintervalls werden.

7 Team Front- und Backend

Kapitel Autor: Marvin Alexander Schäcke, Clara Odinius

7.1 Gruppenmitglieder

Marvin Alexander Schäcke
Clara Maria Odinius

7.2 Einleitung

Das Team Front- und Backend hat sich mit der Erstellung einer Visualisierungsmöglichkeit für DICOM Tags beschäftigt. Zunächst folgt eine kurze Erläuterung und ein Überblick über die Ziele der Gruppe. Danach wird präziser auf die Motivation, vorherige Zielsetzungen und Arbeiten, die Planung und das Vorgehen und die eigentlich Implementierung eingegangen. Abschließend wird ein Fazit gezogen.

7.3 Erläuterung

DICOM (**D**igital **I**maging and **Co**mmunication in **M**edicine) Tags sind Datenelemente oder Attribute, die aus drei Teilen bestehen. Dem Tag, anhand dessen das Attribut eindeutig identifiziert werden kann, der Value Representation (VR), die den Datentyp und das genaue Format beschreibt und dem Namen des Tags. Jedes Tag hat wiederum einen Value, also einen Wert, der idealerweise Informationen enthält beziehungsweise enthalten muss.

Ein Beispiel hierfür wäre:
(0010,0010)
PN
Patient's Name
Peter Patient

Diese Daten sind Tag, VR, Name und Value in genau dieser Reihenfolge. Größere Datensätze beinhalten tausende Patienten und Studien und die dazugehörigen Scans, die alle jeweils DICOM Tags haben.

7.4 Ziele und Anmerkungen

Das Hauptziel des Teams ist es, eine Übersicht dieser Datensätze in Form eines logischen Dashboards zu erstellen, auf das per Weboberfläche zugegriffen werden kann. Wichtig hierbei ist, dass einerseits viele Daten eingebunden und dargestellt werden können und andererseits die Performance subjektiv schnell genug für die Arbeit mit dem Dashboard ist. Dies hat sich auch auf die Wahl der Technologien ausgewirkt. Zusätzlich ist anzumerken, dass dieses Team aus zwei Mitgliedern besteht, weshalb sich der Umfang des internen Projektziels zu denen der anderen unterscheidet.

7.5 DICOM Dashboard

7.5.1 Einleitung

Digital Imaging and Communications in Medicine ist ein offener Standart zur Speicherung und zum Austausch von Informationen im medizinischen Bilddatenmanagement. DICOM Tags beinhalten Informationen zu Patienten, Studien, Serien und vielem mehr. Der Anwender soll unter Anderem auf Anhieb erkennen können, wie viele Patienten an welcher Studie teilgenommen haben, wie umfangreich die jeweilige Studie ist oder um welche Art der Aufnahme von welchem Teil des Körpers oder Organes es sich handelt. Es entsteht ein Gesamtbild der Datensätze. Verbunden mit der Erfahrung

des Nutzers können so einfacher Rückschlüsse darauf gezogen werden, ob z.B. manuell eingetragene Tags richtig sind.

7.5.2 Motivation

Ein Problem der DICOM Tags ist es, dass sie viele Informationen beinhalten, diese aber nicht in übersichtlicher Form darstellen. Insbesondere bei größeren Datensätzen mit vielen Patienten und Studien wird es dadurch unübersichtlich, weshalb eine Visualisierungsmöglichkeit sinnvoll ist. Durch ein Dashboard lassen sich auch mehrere Tags in Kombination darstellen, wodurch sich neue Erkenntnisse für den Nutzer ergeben können. Außerdem variiert die Qualität der Daten, wodurch sie ohne Dashboard manchmal schwer vergleichbar sind.

7.5.3 Vorherige Zielsetzungen und Arbeiten

Vor der eigentlichen Implementierung wurde sich zunächst mit der Auswahl und Aneignung der verschiedenen Technologien beschäftigt. Einige davon haben nach Testläufen doch keine Anwendung gefunden. Unter anderem SQLite, CouchDB, ein lokaler SATORI Server, eine WebApp in JavaScript und Docker fallen in diese Kategorie. Gemeinsam mit den Projektbetreuern wurde sich auf mehrere Technologien geeinigt. Elasticsearch wurde gewählt, da die einfachen Such- und Filterfunktionen und eine hohe Performance, auch bei großen Datensätzen, wichtig sind. Ein zusätzlicher Vorteil ist die Kombination mit Kibana, wodurch sich das Dashboard einfacher erstellen lässt und die vorhandene Verbindung zwischen Suchmaschine und Visualisierung, die bereits implementiert und gut dokumentiert ist. Die Umwandlung der Daten in ein passendes Format und die Einbindung in Elasticsearch wurden durch Module in MeVisLab realisiert. MeVisLab wurde gewählt, da es bei Fraunhofer MEVIS bereits im Einsatz ist und Module, die für die Aufgabenstellung relevant sind, verwendet werden können.

7.5.4 Planung und Vorgehen

Die User Stories, Meilensteine und Zwischenziele wurden gemeinsam mit einem Projektbetreuer und einem Fraunhofer MEVIS Mitarbeiter, der in diesem Bereich tätig ist, erstellt. Die User Stories entsprechen hierbei den Erfahrungen und Vorstellungen des Mitarbeiters. Dadurch wurde die Zielsetzung konkretisiert. Die Meilensteine wurden definiert, um einzugrenzen, was realistisch in der Gruppe erreicht werden kann und wo die Prioritäten liegen. Die Zwischenziele wiederum dienen als Ablaufplan. Anhand dieser kann überprüft werden, ob die Umsetzung noch im Zeitplan ist.

7.5.5 Implementierung

Zunächst musste eine Datenbank oder Suchmaschine gefunden werden, in der die Tags hinterlegt werden können, um mit ihnen anschießend zu arbeiten. Hier bietet sich Elasticsearch in Kombination mit Kibana an. Elasticsearch ist ein in Java geschriebener Suchserver auf Basis von Lucene, der mit Dokumenten im JSON Format arbeitet. Der Client kommuniziert über ein RESTful Webinterface. Kibana baut auf Elasticsearch auf und ist eine browserbasierte Analyseplattform, die sowohl die Suche, als auch die Visualisierung der Dokumente, die in Elasticsearch verfügbar sind, ermöglicht. Module, die mit Elasticsearch und Kibana interagieren, wurden in MeVisLab realisiert.

Im ersten Schritt werden die Hosts und Ports definiert. Findet die Arbeit ausschließlich lokal statt, reichen die Voreinstellung von Elasticsearch (localhost:9200) und Kibana (localhost:5601). Ansonsten wurden hierfür die Module `ElasticsearchConnection` und `KibanaConnection` erstellt. Diese modifizieren die Konfigurationsdateien so, dass beide unter ihren jeweiligen Hosts und Ports erreichbar sind und sie weiterhin untereinander kommunizieren können. Beide können auch einzeln konfiguriert werden. Allerdings muss bei einer Änderung bei Elasticsearch auch immer Kibana angepasst werden, da die Kommunikation sonst nicht mehr stattfindet.

Daher ist es sinnvoll, beide Module zu verwenden und diese vom Output von `ElasticsearchConnection` zum Input von `KibanaConnection` zu verbinden. Im Modulpanel von `ElasticsearchConnection` können der Host und Port von Elasticsearch angepasst werden. Diese werden auch an den Moduloutput weitergegeben. Zwingend notwendig ist die Angabe des Pfades zur Konfigurationsdatei `elasticsearch.yml`, der je nach Betriebssystem, Installationsart und Installationspfad unterschiedlich ist. Im Modul `KibanaConnection` können der Host und Port von Kibana angepasst werden. Des Weiteren muss, bei einer Änderung bei Elasticsearch, die URL inklusive http oder https zu Elasticsearch angegeben werden. Die Daten aus dem Modulinput werden automatisch übernommen. Auch hier ist es zwingend notwendig den Pfad zur Konfigurationsdatei `kibana.yml`, der je nach Betriebssystem, Installationsart und Installationspfad unterschiedlich ist, anzugeben, da Änderungen sonst nicht übernommen und von Elasticsearch und Kibana verarbeitet werden können. In beiden Modulen sorgt der Button „Define connection“ dafür, dass dies geschieht.

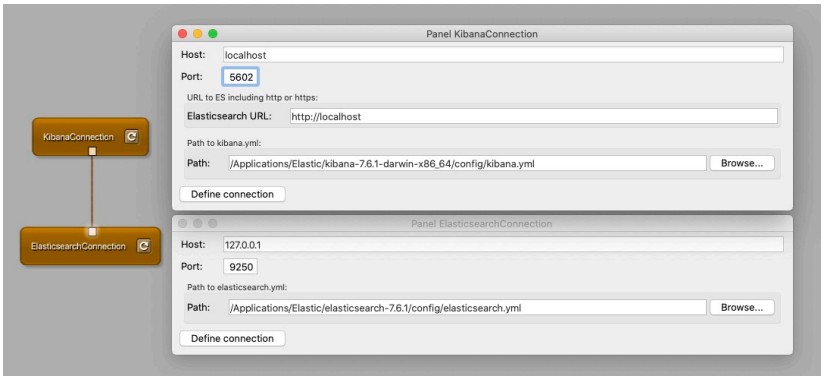


Abbildung 7.1: Beispielhafte Änderungen in MeVisLab

Der nächste Schritt ist es, die DICOM Tags aus den Datensätzen zu extrahieren, da die Aufnahmen selbst nicht verwendet werden. Basierend auf der Wahl der Technologien ergibt sich die Aufgabenstellung, die DICOM Datensätze in JSON umzuwandeln. Das bereits existierende Modul ”DirectDicomImport” von Wolf Spindler wurde in den Prozess eingebunden. Es ermöglicht das Importieren der DICOM Datensätze in MeVisLab und fasst diese in Volumen zusammen. An dessen Output wird das Modul `MultiFileVolumeListIteratorOutput` von Wolf Spindler angeknüpft. Es iteriert über die gefundenen Volumen des `DirectDicomImport`.

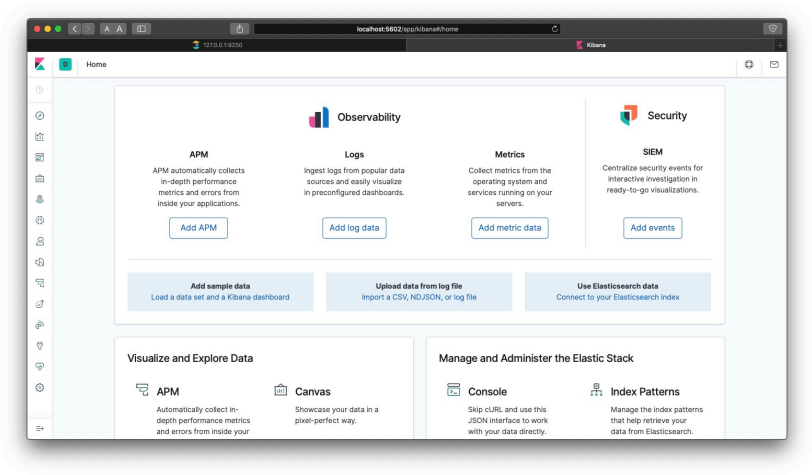


Abbildung 7.2: Änderungen sichtbar im Browser

An den Output von `MultiFileVolumeListIteratorOutput` knüpft das Modul `PushJsonToElasticsearch` an, wodurch jedes einzelne Volumen einfach bearbeitet werden kann.

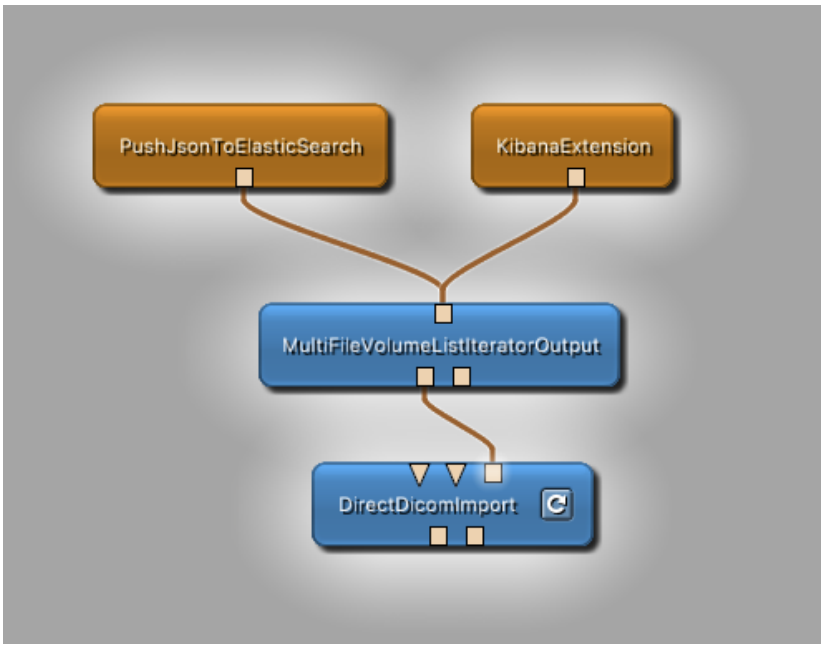


Abbildung 7.3: Funktionsaufbau

Die Datenstruktur in Elasticsearch sieht für jeden Eintrag einen Index und eine ID vor. Der Index wird zuerst erstellt und beschreibt einen Datensatz. Jeder erzeugte Eintrag im Datensatz erhält eine eigene ID.

Wird ein neuer Eintrag mit der gleichen ID erstellt, wird der vorherige

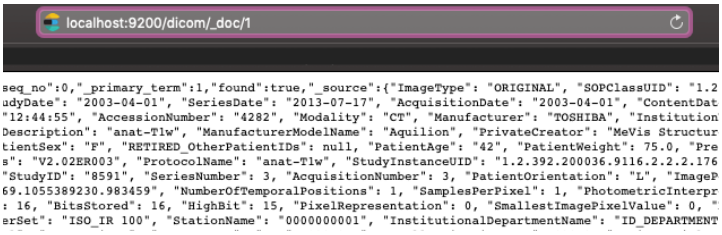


Abbildung 7.4: Daten mit dem Index „dicom“ und der ID „1“

überschrieben. Damit dies nur beabsichtigt geschieht, kann der Wert des Tags (0008,0018) *SOP Instance UID* als ID verwendet werden, da jedes Volumen über eine einmalige SOPInstanceUID verfügt.

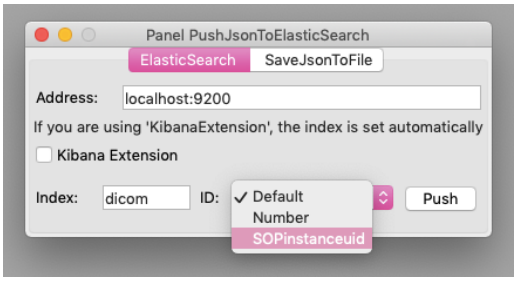


Abbildung 7.5: User Interface von PushJsonToElasticSearch - Elasticsearch

Im Modul „PushJsonToElasticsearch“ gibt der Anwender zunächst die Adresse zu Elasticsearch an. Anschließend bestimmt er eine Bezeichnung für den Index und wählt für die ID zwischen *Default (hexadezimal)*, *Number (chronologisch nummeriert)* und *SOPInstanceUID (wie zuvor erläutert)*. Das Drücken des „Push“-Buttons sorgt dafür, dass die Eingaben gelesen, Elasticsearch instanziiert und der Index erstellt werden. Für jedes Volumen werden die Tags initialisiert, deren *Tag Names* mit ihren *Tag Values* verknüpft und in JSON-Objekte zusammengefasst. Diese Objekte können nun als „body“ zusammen mit dem Index und der ID *gepusht* werden und sind somit in Elasticsearch hinterlegt. Des Weiteren können die Daten unter Angabe eines Pfades zusätzlich lokal gespeichert werden. Die Art der Datei ist auswählbar.

Lokale JSON-Dateien können über das Modul `LocalJsonToElasticsearch` in Elasticsearch eingebunden werden. Falls möglich, werden der Host und Port von Elasticsearch vom Modulinput übernommen. Ansonsten müssen beide über das Modulpanel angegeben werden. Des Weiteren werden ein Indexname und die Art der ID-Zuweisung benötigt. Abschließend können ent-

Abbildung 7.6: User Interface von PushJsonToElasticSearch - Save JSON to file

weder einzelne JSON-Dateien oder ganze Ordner per Pfad angegeben werden. Bei Ordnern werden ausschließlich Dateien verarbeitet, die auf `.json` enden. Die Buttons „Save File To ES“ und „Save Dir To ES“ starten diese Prozesse.

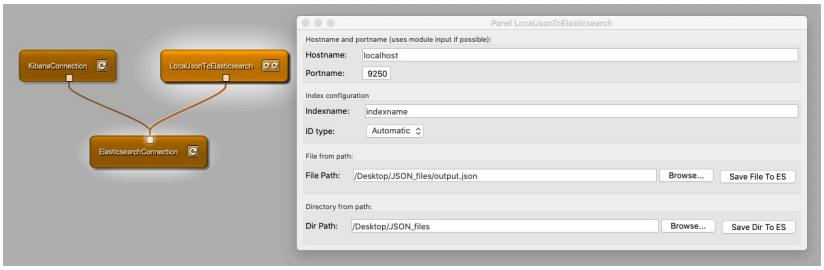


Abbildung 7.7: Beispielhafte Konfiguration in MeVisLab

Kibana bietet die Möglichkeit die Datensätze zu visualisieren. Dafür muss ein Index-Pattern erstellt werden, das die Daten einer Visualisierung zuweist. Im Folgenden wird die Funktionsweise beispielhaft erläutert.

In der Abbildung 7.8 wird veranschaulicht, wie viele Volumen welchen Körperteil beschreiben. Für die Y-Achse wird die Aggregation count angegeben, um die jeweilige Menge der Volumen zu zählen. Für die X-Achse wird das Feld *BodyPartExamined* ausgewählt, um für jeden vorkommenden Wert eine Säule darzustellen. Dadurch, dass bei *Split Series* erneut *BodyPartExamined* in Zusammenhang mit *PatientID* gewählt wird, werden die Säulen farblich unterteilt. Jede Farbe ist einem Patienten zuzuordnen. Darüber hinaus sind diese Visualisierungen dynamisch. Wird auf ein Feld geklickt, werden nur noch die Daten angezeigt, die im Zusammenhang mit der Auswahl stehen. In diesem Fall sind es die Daten des Patienten, dem das Feld zuzuordnen ist.

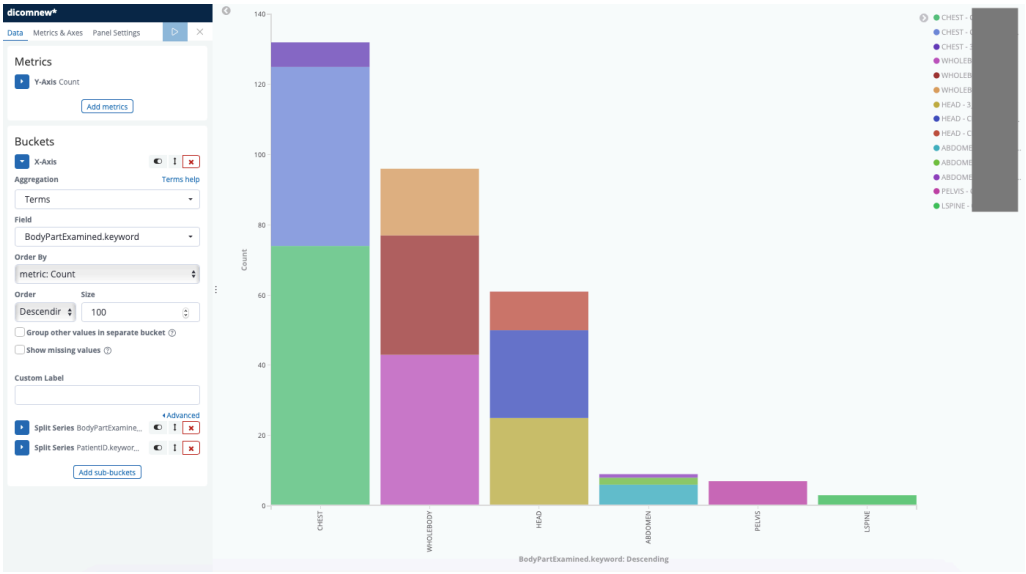


Abbildung 7.8: Beispielhafte Kibana Visualisierung

Wird ein Dashboard erstellt, können die Visualisierungen hinzugefügt und angeordnet werden. Alle Objekte werden in Kibana als *Saved Objects* gespeichert. Die *Saved Objects* Index-Pattern, Visualisierungen, Dashboards und Saved Searches können als `.json` exportiert und importiert werden. Wird ein Dashboard exportiert, so werden auch die darin enthaltenen Objekte exportiert. Ausgenommen sind die dem Index-Pattern hinterlegten Daten.

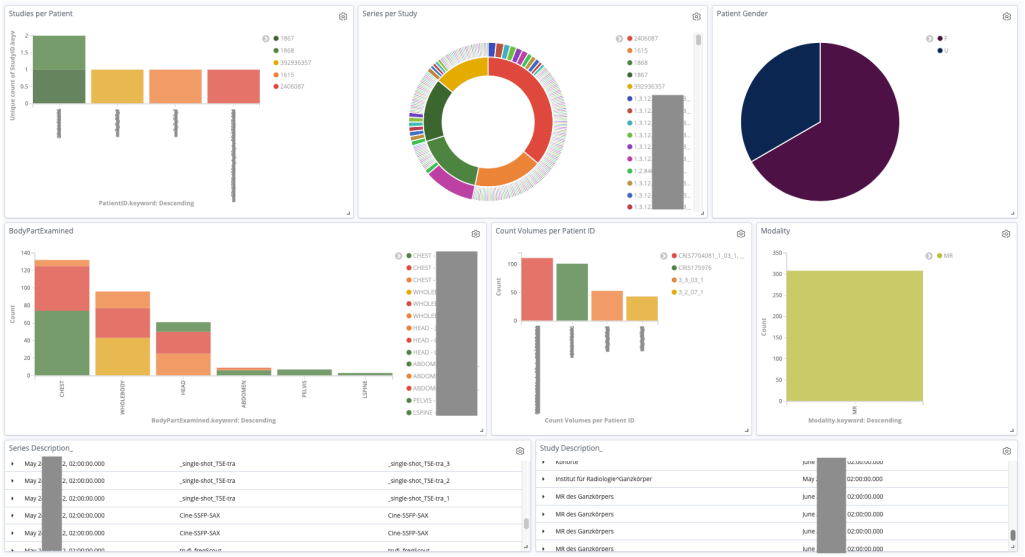


Abbildung 7.9: Dashboard einer Vorstudie mit vier Patienten

Der bisherige Funktionsumfang in MeVisLab kann durch das Modul

Abbildung 7.10: Dashboard mit Gender-Filter

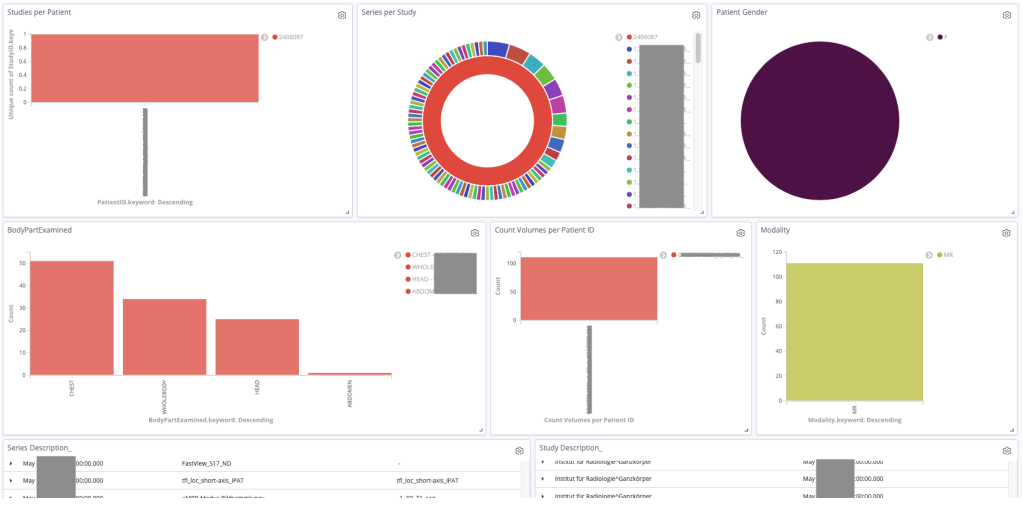


Abbildung 7.11: Dashboard mit Patient-Filter

`KibanaExtension` erweitert werden. Es beinhaltet das im Bachelorprojekt erstellte Dashboard als Script-Datei. In dieser Datei sind die entsprechenden cURL Befehle und die `export.json` hinterlegt. Um sie ausführen zu können, muss der Benutzer den Pfad zu seinen Makromodulen und die Adresse angeben. Per Klick auf „Import Dashboard“ wir das Dashboard importiert.

Da das Index-Pattern des Dashboards fest definiert ist, gibt es in

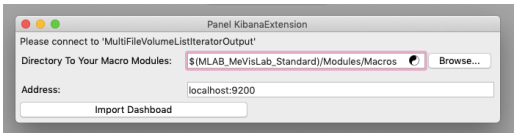


Abbildung 7.12: User Interface von KibanaExtension

`PushJsonToElasticsearch` eine Checkbox (siehe Abbildung 7.5). Wenn Daten, mit der Absicht sie in diesem Dashboard anzeigen zu lassen, gepusht werden, kann diese Checkbox gewählt werden. Der Index wird dann automatisch initialisiert und das Index-Pattern erstellt. Der Anwender hat so die Möglichkeit die DICOM Daten optimal in Kibana zu nutzen.

7.6 Ausblick

Das Dashboard könnte in zukünftigen Projekten oder Bachelorarbeiten um zwei Aspekte erweitert werden. Dies wäre einerseits die Zusammenfassung ähnlicher Werte oder verschiedener Schreibweisen der Werte der DICOM Tags zu einem dargestellten Wert. Dadurch würde das Dashboard bei großen Datensätzen übersichtlicher werden. Andererseits wäre die Möglichkeit zu einem Viewer, der die ausgewählten Scans anzeigt, zu wechseln eine sinnvolle Erweiterung.

7.7 Fazit

Auf das Projektziel hinzuarbeiten war sehr interessant und lehrreich. Es gab Hürden, an denen die Teammitglieder gewachsen sind, aber ebenso auch Erfolgserlebnisse, die zur Weiterarbeit motiviert haben. Es war eine tolle Möglichkeit, viele verschiedene Technologien kennenzulernen und Einblicke in das Fraunhofer MEVIS Institut zu bekommen. Es hat Freude bereitet, an dem Projekt zu arbeiten und die gesetzten Meilensteine zu erreichen. Das Team ist mit der Realisierung zufrieden und kann sich vorstellen, diese im Rahmen einer Bachelorarbeit zu erweitern.

8 Team Website

8.1 Einleitung

Autor: Tom Köhler

Die Website www.deepanatomy.de soll als Anlaufpunkt für Interessenten am Projekt dienen, sei es Studenten auf der Suche nach einem Bachelor- oder Masterprojekt oder anderen Interessenten der Thematik. Da der Projekttag abgesagt wurde, auf dem die Ergebnisse der Projekte jährlich vorgestellt werden, präsentieren wir die Ergebnisse nun auf der Website. Die Website wurde so designed, dass folgende Jahrgänge ihre Ergebnisse dort auch präsentieren können. Die Mitglieder des Team Website sind Joana Becker, Jan-Gerrit Göbel, Tom Köhler und Lukas Schäfer.

8.2 Motivation/ Ziele

Autor: Joana Becker

Das Erstellen der Website folgt mehreren Gründen. Einerseits ist eine Website über die Aufgaben, Teilnehmer und Ergebnisse jedes Jahres ein guter Anlaufpunkt für Interessenten an dem Bachelorprojekt im nächsten Jahr. Des Weiteren bietet es einen formalen und angenehmen Überblick über die erreichten Ergebnisse, der optisch ansprechend dargestellt ist und andererseits stellt es einen Ersatz zu dem fehlenden Projekttag dar, der wegen Covid-19 abgesagt wurde. So können sich auch Firmen, andere Studierende, Dozenten und weitere Interessierte über das Projekt informieren, die sonst den Projekttag besucht hätten.

8.3 Struktur

Autor: Tom Köhler, Joana Becker

Mithilfe des static Website Generators Jekyll wurde die Website aufgebaut. Jekyll erstellt eine statische Website anhand von Layout-Dateien und Inhaltsdateien.

Die Layout-Dateien geben Jekyll an wie es die Inhaltsdateien darstellen soll. Die Ordnerstruktur spiegelt die Aufteilung in Layout-Dateien und Inhaltsdateien wieder. Zu den Layout-Dateien zählen zum Beispiel folgende Ordner: layouts, sass, assets, data und die config. Der Inhalt wird durch die `*.html` Dateien, den Bildern und `.pdf` Dateien dargestellt.

Jekyll kann Internetseiten aus HTML- und Markdown-Dateien erstellen. Für die Vereinheitlichung wurden fast alle Seiten in HTML geschrieben. Die Optik ist hingegen in den `sass`-Dateien in SASS verfasst, einer Erweiterung von CSS. SASS lässt sich einfach schreiben und einbinden. Ebenso ermöglicht SASS die Anpassung von jedem Detail der Oberflächenoptik.

Um die Bearbeitung der Website einfacher und zukunftsgerichtet zu gestalten, wurden die Inhalte in Ordnern angelegt. Da die Verwendung der Website von den folgenden Projektgruppen ebenfalls eingeplant wurde, gibt es auch eine Aufteilung in Jahre. Die Projektdateien dieses Jahres befinden sich daher in einem eigenen Order. Allgemeine Informationen wurden ebenfalls in Ordnern nach Themen oder Inhalten sortiert. Damit ist ein einfacheres Auffinden der Dateien, eine simplere Benennung und eine bessere Übersicht über die Struktur sichergestellt.

Der strukturelle Aufbau der Website selbst folgt einem einfachen Schema. Es gibt einen allgemeinen Header, der auf allen erreichbaren Seiten angezeigt wird und das Logo, sowie die Navigation enthält. Dieser Header bleibt fix am oberen Rand der Website, unabhängig vom Scrollen im Inhalt des Bodys. Am Ende des Inhalts findet sich ein Footer, der eine kurze Beschreibung des Projekts und der Webseite enthält, sowie die Links zur Datenschutzerklärung und zum Impressum.

Die Informationen zum Projekt und die einzelnen Jahre sind jeweils über ein Drop-Down-Menü in der Navigation erreichbar. Nach Auswahl eines

Jahrgangs wird dem Header eine weitere fixe Leiste hinzugefügt, über die die Informationsseiten zu dem jeweiligen ausgewählten Jahr erreichbar sind.

Damit die Nutzer einen schnellen Überblick über ihren Standort auf der Website erhalten, wurden die in Abbildung 8.1 dargestellten Breadcrumbs integriert. Diese stellen die Folge der aufgerufenen Unterseiten dar und ermöglichen dem Nutzer eine vereinfachte Navigation.

Abbildung 8.1: Breadcrumbs

8.4 Inhaltlicher Aufbau

Autor: Lukas Schäfer

Wie im Abschnitt 8.2 erwähnt, soll die Website einen Überblick über das Projekt im Allgemeinen geben, aber auch die Arbeit der einzelnen Projektteams über die Jahre enthalten. Der inhaltliche Aufbau richtet sich an diesen Zielen aus. Wie in Abbildung 8.2 dargestellt, ist der Grundaufbau der Website eingeteilt in die allgemeine Vorstellung des Projekts, die Auswahl eines bestimmten Jahrgangs, die Ansprechpartner des Projekts und Kursempfehlungen zur Vorbereitung auf das Projekt. Die allgemeine Vorstellung des Projekts teilt sich auf in einen Einblick in das Projektumfeld, mit einer kurzen Vorschau zu den Themen „Segmentierung anatomischer Strukturen in medizinischen Bildern“, „Vorarbeiten und Infrastruktur“ und die „konkrete Aufgabenstellung“, sowie der Vorstellung der Arbeitsgruppe und der Geschichte des Projekts.

Abbildung 8.2: Navigationsleiste

Der Bereich, in dem die Ergebnisse eines bestimmten Jahrgangs präsentiert werden, unterteilt sich, wie in Abbildung 8.3 zu sehen, in weitere Abschnitte. Im Abschnitt „Team *Jahrgang*“ ist eine Übersicht mit einem Gruppenfoto und Links zu den im Folgenden beschriebenen Abschnitten zu sehen. Im

Abschnitt „Ziele“ sind die konkreten Ziele dieses Jahrgangs dargestellt, die im Abschnitt „Untergruppen“ Teilen des Projektteams zugeordnet werden. Zusätzlich zu dieser Zuordnung wird im Abschnitt „Untergruppen“ auf die Arbeit an den Zielen eingegangen. Dabei wird die Motivation hinter dem Ziel erläutert, das Vorgehen skizziert und das Endresultat präsentiert. In weiteren Abschnitten werden das Projektteam des Jahrgangs und die Verantwortlichen vorgestellt. Im letzten Abschnitt kann der Projektbericht, der am Ende jedes Durchlaufs erstellt wird, heruntergeladen werden.

Abbildung 8.3: Navigationsleiste Projektteam

8.5 Gitlab CI-Pipeline

Autor: Jan-Gerrit Göbel

Um Fehler schneller zu bemerken, sowie den Deployment-Prozess zu beschleunigen, wurde eine CI-Pipeline definiert.
Die Pipeline soll bei jedem commit auf den master-Branch automatisiert das Projekt bauen und auf Fehler während des Builds hinweisen. Außerdem soll beim Pushen auf einen gesonderten Branch die Website deployt werden. Damit Pipelines ausgeführt werden können wird ein GitLab Runner benötigt. Ein GitLab Runner ist eine Virtuelle Maschine in welcher fest definierte Prozesse ablaufen können. Ähnlich wie bei Docker gibt es eine Konfigurationsdatei in welcher unter anderem festgelegt wird welches Image benötigt wird. Da zum Bauen der Website Jekyll, ein Static-Site Generator, verwendet wird, benötigen wir in der Pipeline ein Ruby Image.

Die Pipeline besteht aus zwei Phasen, der Build-Stage sowie der Deploy-Stage, welche nacheinander ausgeführt werden. Bevor die erste Stage durchlaufen wird, wird Bundler installiert. Bundler ist ein Package Manager für Ruby gems welcher für die Installation von Jekyll benötigt wird.

In der Build-Stage wird Jekyll installiert. Dadurch dass der Installationspfad im cache gehalten wird geschieht die Installation ein einziges mal und wird

auch für die nächsten Durchläufe der Pipeline verwendet, bis der Runner heruntergefahren wird. Dies sorgt für eine erheblich schnellere Durchlaufzeit der Pipeline. Nach der Installation wird die Website gebaut und als Artifact ausgegeben.

In der Deploy-Stage wird das Artifact auf den Server auf dem die Seite gehostet wird deployt. Die Umsetzung dieser Stage geschah mit Unterstützung durch das IT-Team von Mevis, welches auch das Einrichten eines Servers sowie die Registrierung der Domain `deepanatomy.de` übernahm.

8.6 Fazit

Autor: Lukas Schäfer

Die in Abschnitt 8.2 formulierten Anforderungen wurden vollständig umgesetzt. Es ist eine Website entstanden, auf der das diesjährige und die folgenden Projekte ihre Arbeitsergebnisse präsentieren können. Außerdem wurde die durch das Team erstellte Website auf der Projekttagswebsite verlinkt, um den interessierten Besuchern einen Einblick in unsere Arbeit zu gewähren.
Auch das Ziel, den Interessenten der nächsten Jahrgänge eine Anlaufstation zu bieten, wurde erreicht.

8.7 Ausblick

Autor: Lukas Schäfer

Eines der in Abschnitt 8.2 genannten Ziele war es, eine Website zu erschaffen, auf der auch die folgenden Bachelor- und Masterprojekte ihr Arbeitsergebnisse präsentieren können. Für dieses Ziel wurde mit dem inhaltlichen Aufbau der Grundstein gelegt, auf dem die folgenden Projektteams aufbauen können.

Eine Erweiterungsmöglichkeit ist eine zusätzliche Seite, die bei einem Klick auf „Projektteams“ geöffnet wird. Auf dieser Seite werden die unterschiedlichen Jahrgänge mit jeweils einem Bild als Gesamtübersicht dargestellt.

Mit einem Klick auf eines der Bilder, wird der Bereich des entsprechenden Jahrgangs geöffnet.

Eine weitere Erweiterungsidee wäre das Einbinden der Arbeitsergebnisse der Challenge Teams zur interaktiven Ansicht durch die Besucher der Website. Es könnte eine Seite erstellt werden, in der eine 3D Ansicht integriert ist. Es können verschiedene Datensätze und Strukturen ausgewählt werden, die in der 3D Ansicht angezeigt und selbst zu erforschen sind. Im Zuge dieser Erweiterung könnte es auch möglich sein, die eigenen MRT- oder CT-Bilder hochzuladen und die annotierten Bilder angezeigt zu bekommen.

9 Projekttag

Kapitel Autor: Lukas Schäfer

Der Bachelorprojekttag ist eine Veranstaltung die jährlich stattfindet und bei der die aktuellen Bachelorprojektteams ihre Projekte präsentieren. Die Veranstaltung findet traditionell etwa einen Monat vor Ende der Projekte, in der Ebene 1 bzw. Ebene 0 des Mehrzweck Hochhauses, statt. Organisiert wird der Projekttag von einer Organisationsrunde, die sich aus den „Außenministern“ der einzelnen Projekte zusammensetzt. Die Außenminister haben die Aufgabe ihr Projekt gegenüber den anderen Projekten zu vertreten. Für das Projekt DeepAnatomy wurde Lukas Schäfer zum Außenminister ernannt.

Auf dem Bachelorprojekttag haben die Projekte die Chance ihre Arbeitsergebnisse zu präsentieren. Die Präsentation findet wie auf einer Messe in Form eines „Messestandes“ für jedes Projekt statt, wobei die Form der Präsentation variieren kann. Zusätzlich zu den Ständen der Projekte präsentiert jedes Projekt seine Arbeitsergebnisse in Form einer 20 minütigen Präsentation im großen Vorlesungssaal der Ebene 1. Die üblichen Präsentationsarten an den Ständen sind Plakate, Bilder oder Filme, die auf großen Monitoren oder per Beamer gezeigt werden und Live-Demonstrationen, die beispielsweise mit den Robotern eines Projekts durchgeführt werden. Zusätzlich können interessierte Personen mit den Projektmitgliedern am Stand sprechen, Fragen stellen oder sich bestimmte Aspekte des Projekts näher erläutern lassen. Die Besucher des Bachelorprojekttags sind üblicherweise Mitglieder der zugehörigen Arbeitsgruppen, Professoren und andere Mitarbeiter, sowie die Studierenden des Fachbereichs.

Das Datum für den Projekttag wurde durch den Fachbereich auf den 17. April gelegt. Die Planung des Projekttags wurde daraufhin im Januar mit

einem Treffen der Außenminister gestartet, bei dem die grundlegenden Ideen für die Veranstaltung in diesem Jahr zusammengetragen, mögliche Dokumente aus den letzten Jahren organisiert und Kommunikationskanäle angelegt wurden. Im weiteren Verlauf der Planung fand alle zwei Wochen ein Meeting statt, bei denen der aktuelle Stand ausgetauscht und die weitere Vorgehensweise besprochen wurden. Die Hauptaufgaben der Organisation waren es, Räume, Plakatwände, Monitore, LAN-Anschlüsse und Strom für die Messestände bereitzustellen. Weitere Aufgaben waren das Pflegen der Projekttagswebsite und das Erstellen der Präsentationen in den Projektteams. Eine weitere Idee der diesjährigen Organisationsrunde war es auch, Firmen und Schulklassen für den Projekttag einzuladen, sodass ein Teil der Außenminister sich mit Einladungen und Personenlisten beschäftigt hat.

Die Vorbereitungen wurden Anfang März durch die Schließung der Uni aufgrund der Covid-19-Pandemie ausgesetzt und es wurde auf den neuen Termin für den Projekttag gewartet. Da nicht abzusehen war, wann und ob der Projekttag stattfinden wird, wurde im Projekt DeepAnatomy beschlossen, kurzfristig eine Website zu erstellen, auf der das Projekt präsentiert wird. Die Einzelheiten zur Entwicklung der Website sind in Kapitel 8 zu lesen.

Ende April wurde durch die Studienkommission entschieden, den Projekttag durch eine Website zu ersetzen, auf der sich die Projekte präsentieren können. Da zu diesem Zeitpunkt die Entwicklungsphase der Projektwebsite bereits angelaufen war, wurde mit in der Runde der Außenminister beschlossen, dass ein Link auf die Projektwebsite ausreichend ist, um das Projekt auf der Projekttagswebsite darzustellen.

10 Ausblick

Kapitel Autor: Marvin Alexander Schäcke, Markus Rink, Karol Baginski

Das Projekt greift in alle Schritte des Trainieren eines neuronalen Netzes mit der Eingrenzung auf Segmentierung und dem medizinischen Kontext. Grob kann man die Teams in folgende Bereiche einteilen:

1. Daten sammeln und Überblick über Daten → Team DICOM Dashboard (Tool zum Import und zur Visualisierung der DICOM Tags)
2. Training erstellen → Team Challenges (Expertise der praktischen Anwendung)
3. Training durchführen → Team Optimierte Trainingsschleife (Tools für mehr Feedback und reibungslose Benutzung)
4. Vergleichen und Auswerten → Bisher kein spezialisiertes Team, aber bereits unterstützt mit Challengr (z.B. mit Confusion Matrix)

Wie in den Ausblicken der Teams zu sehen, gibt es in den einzelnen Bereichen noch viel zu ergänzen.

Das Modulsystem zur Arbeit mit Confusion-Matrizen in MeVisLab kann weiter ausgebaut werden (Abschnitt 6.7). Aber auch die Weiterentwicklung der vorhandenen Module kann verfolgt werden, um z.B. automatische Labelnamen, für die sogar bereits Ansätze bei Fraunhofer MEVIS existieren. Das System eines Segmentation Pipeline Optimizers wurde theoretisch erstellt und kann verfeinert, sowie teils oder ganz implementiert werden.

Auch Team übergreifend gibt es noch Erweiterungspotential. DICOM Tags enthalten reichhaltige Informationen neben den Bilddaten, welche zur Zeit nur im Dashboard genutzt werden. In zukünftigen Projekten könnte eine

Auswertung der Werte der DICOM Tags erstellt. Hierfür müssten die Values exportiert und in ein sinnvolles Format gebraucht werden, sodass ein neuronales Netz mit diesen arbeiten könnte. Die Ergebnisse wiederum müssen aufbereitet, analysiert und auf Richtigkeit überprüft werden.

Die in den Challenges gesammelte Erfahrung der Teilnehmer kann anderen als Startpunkt für die ersten Schritte bei ihrer Bearbeitung dienen. Dadurch können erste Stolpersteine vermieden und eine schnellere Einarbeitung erreicht werden. Auf dieser Basis können neue Erkenntnisse gesammelt werden, die das Wissen über Challenges ergänzen.

Bei der Bearbeitung sind verschiedene Architekturen entstanden. Diese bestehen aus mehreren Dateien und liegen daher nur durch Ordner sortiert im Filesystem. Eine Plattform für MeVisLab Netzwerke könnte Transparenz, Übersichtlichkeit und Struktur bringen. Transparenz in Versionen und Varianten von Netzen, Übersichtlichkeit über bereits bearbeitete Daten und Struktur in die Speicherung.

11 Fazit

Kapitel Autor: Marvin Alexander Schäcke, Markus Rink, Karol Baginski

Das Projekt bot eine umfangreiche Auswahl an Arbeitsgebieten und ermöglichte so jedem eine zu ihm passende Aufgabe zu finden. Dazu gehören nicht nur die praktische und theoretische Weiterentwicklung von Spezialsoftware, die Auseinandersetzung mit konkreten medizinischen Anwendungsszenarien und das Verständnis von CNNs. Auch die Entwicklung wichtiger Softskills, wie die Themenübergreifende Präsentation zwischen und die Zusammenarbeit von Teams, eine offene und dennoch zielgerichtete Diskussionskultur, sowie die eigenverantwortliche, als auch kooperative Einarbeitung in neue Themen, konnte das Projekt fördern.

Die Kooperation zwischen Gruppen litt vor allem unter der im Allgemeinen erst späten Implementierung und dem langen Fehlen konkreter Ergebnisse, auf die sich andere Gruppen beziehen hätten können. Dies ist auch dem Umstand geschuldet, dass die Konfrontation der Teilnehmer mit unbekannten Technologien eine lange Einarbeitungsphase mit sich bringt. Dadurch war es auch nur schwer möglich ein kohärentes Ziel für das Projekt als solches zu definieren, das allen Gruppen gemeinsam einen Rahmen für ihre Arbeit gegeben hätten. Gemessen an diesen Möglichkeiten und Hürden konnten dennoch zufriedenstellende und dem Umfang angemessene Ergebnisse erzielt werden.

Die Wichtigkeit einer guten Projektorganisation wurde schnell erkannt. So wurden Kommunikationskanäle, Managementsysteme und auch Sanktionen zu beginn des Projektes festgelegt. Dennoch hätte die eigenverantwortliche Beachtung dieser Regeln etwas besser ausfallen können, z.B. bei der regelmäßigen Pflege der Diarys oder der Organisation und Dokumentierung der

Arbeitspakete in Jira. Ein durchgängiges Management hätte dem Auseinanderdriften der Gruppen entgegenwirken können. Allerdings ist fragwürdig, ob dies im Rahmen dieses Projekts realistisch gewesen wäre, da dies von den Teilnehmern zusätzlich zu anderen Aufgaben hätte geleistet werden müssen.

Die Projektbetreuung gestaltete sich hervorragend. Die Zuständigen Mitarbeiter waren jederzeit bei Ideen und Problemen ansprechbar und wiesen auch eigenständig auf Probleme und Möglichkeiten hin. Sie vermittelten aktiv den Kontakt zu geeigneten Ansprechpersonen und halfen unkompliziert bei internen Problemen, wie gesperrten Accounts. Auch sie brachten sich aktiv in die Projektorganisation ein und konnten mit ihrem Fachwissen jederzeit beim Verständnis und der Einarbeitung behilflich sein.

12 Glossar

Kapitel Autor: Daniel Bröcker, Thorben Lorenzen

abdominalen Organe

Organe, die sich zwischen Brustkorb und Becken befinden.

Annotation

Markierung in einem medizinischen Bild, welche auf bestimmte Merkmale hinweist. Beispielsweise auf ein bestimmtes Organ.

CNN

Convolutional Neural Networks gehören zu den maschinellen Lernalgorithmen. Sie sind von menschlichen Neuronen inspiriert und haben mehrere Schichten von Neuronen. Werden sehr häufig in der Bild- und Spracherkennung angewandt.

Covid-19

Pandemie, welche insbesondere im Frühjahr 2020 viele Kontaktbeschränkungen erforderte. Und somit die zwischenmenschliche Zusammenarbeit in großen Teilen auf Webkonferenzsysteme verlagerte.

CT

Computertomographie verfahren um Schnittbilder eines Menschen zu gewinnen.

Gitlab

Eine weit verbreitete Webanwendung zur Kollaboration und Versionsverwaltung. Insbesondere für textbasierte Dateien.

GUI

Grafische Oberfläche eines Softwareprogrammes.

Ground Truth

Bei supervised Learning werden Referenz Label als Ground Truth bezeichnet.

Jaccard-Wert

Ein Wert von 0 bis 1, welcher die Übereinstimmung von zwei Mengen repräsentiert. Hier die Übereinstimmung der Reference-Label mit den Ergebnissen eines Trainings eines CNNs. 1 repräsentiert die volle Übereinstimmung der 2 Mengen, währrend bei 0 keine Übereinstimmung existiert.

Label

Benannte Zuordnung einer Annotation zu einer bestimmten Klasse. Beispielsweise *Leber*.

MeVisLab

Eine vom Fraunhofe MeVis entwickelte Software für die Verarbeitung von medizinischen Bilddaten.

MeVisLab-Modul

MeVisLab ist nach dem Baukastenprinzip konzipiert. So können sämtliche Funktionen als Modul verwendet und mit anderen Modulen kombiniert oder zu neuen Modulen entwickelt werden.

MeVisLab-Netzwerk

Mehrere MeVisLab-Module, welche miteinander verbunden sind und häufig in Wechselwirkung miteinander stehen. Wird i.d.R. erstellt um die Funktionalitäten einzelner Module zu kombinieren. Beispielsweise ein Modul, welches Bilder lädt. Verbunden mit einem Modul, welches die Bilder anzeigt.

MRT

Magnetresonanztomographie verfahren um Schnittbilder eines Menschen zu gewinnen.

Patch

Bildausschnitt, mit dem ein CNN traniert wird.

Preprocessing

Menge der Operationen die vor dem Training des neuronalen Netzes angewandt werden um dieses zu ermöglichen und zu verbessern (Bsp.: Bildgrößen zuschneiden) .

Segmentierung

Verfahren welches den Pixeln eines Bildes ein Label hinzufügt oder verschiedene Regionen eines Bildes markiert.

Serie

Sammlung von Bildern. Beispielsweise Bilder eines bestimmtes Knies an einem bestimmten Tag.

Slice

Ein Bild, das eine aufgenommene Schicht repräsentiert, aus einer Daten-Serie eines Patienten

U-Net

Ein CNN mit einer bestimmten Architektur. Optimiert für biomedizinische Bilder-Segmentierung.

Voxel

Bild-Punkte, welche im CNN klassifiziert werden.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Alexander Koehn. Challengr help page, 2020.

[2] Fabian Isensee, Jens Petersen, André Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra Michela Antonelliand Sebastian J. Wirkert, and Klaus H. Maier-Hein. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *CoRR*, abs/1809.10486, 2018.

[3] Amber L. Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A. Landman, Geert J. S. Litjens, Bjoern H. Menze, Olaf Ronneberger, Ronald M. Summers, Patrick Bilic, Patrick Ferdinand Christ, Richard K. G. Do, Marc Gollub, Jennifer Golia-Pernicka, Stephan Heckers, William R. Jarnagin, Maureen McHugo, Sandy Napel, Eugene Vorontsov, Lena Maier-Hein, and M. Jorge Cardoso. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *CoRR*, abs/1902.09063, 2019.

[4] Bram van Ginneken, Sjoerd Kerkstra, and James Meakin. Grand challenge: Medical segmentation decathlon, results, 2019.

[5] Ragnar Bade. Mevislab, 2020.