INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Segmentierung des Magen-Darm-Trakts anhand von MRT Bildern mithilfe von Deep Learning

Viktor Leonardo Krawutschke

Bachelorarbeit

Beginn der Arbeit:	01. April 2010
Abgabe der Arbeit:	25. September 2022
Gutachter:	Univ.-Prof. Dr. S. Conrad
	Univ.-Prof. Dr. M. Kollmann

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 25. September 2022

Viktor Leonardo Krawutschke

Einleitung

Diese Bachelorarbeit beschäftigt sich mit der semantischen Segmentierung von MRT Bildern im medizinischen Bereich. Im Jahr 2015 wurde bei 5 Millionen Menschen Krebs im Gastro-Intestinaltrakt festgestellt, wovon die Hälfte für eine Strahlentherapie in Frage kommen. Diese dauert in der Regel 1 bis 6 Wochen und findet täglich für 15 Minuten statt. Dabei soll den Krebszellen gezielt eine hohe Strahlendosis verabreicht werden, gleichzeitig ist es zu vermeiden nicht betroffene Organe dabei in Mitleidenschaft zu ziehen. Problematisch ist, dass die Position der Organe und somit auch der Tumore von Tag zu Tag unterschiedlich ist. Dank Magnetresonanztomographen (MRT) und Linearbeschleunigern können Onkologen jedoch die genaue Position der Organe feststellen. Dies ist jedoch ein Zeitintensiver Prozess und dauert je nach Patienten zwischen 15 und 60 Minuten. Deep Learning könnte dabei helfen, diesen Prozess erheblich zu beschleunigen, um so Patienten zu entlasten und Ärzten die Chance zu geben mehreren Menschen im selben Zeitraum ihre Hilfe anzubieten.

Das UW-Madison Carbone Cancer Center hat im Rahmen einer Challenge auf Kaggle.com anonymisierte Scans von über 150 Patienten zur Verfügung gestellt, in denen manuell der Dickdarm, Dünndarm und der Magen von Fachleuten segmentiert worden ist. Ziel ist es, ein Modell zu entwerfen welches gegeben den Trainingsdaten dazu in der Lage ist in Bildausschnitten von vorher ungesehenen Patienten die Lokalität der genannten Organe in Form von einer Maske zu bestimmen. Baseline dieser Arbeit bildet ein Convolutional Neural Netwrok, einem UNet Modell, welches auf 2 dimensionalen Daten antrainiert wurde, da es sich bewiesen hat Segmentierungen im Medizinischen Bereich vorzunehmen. Es besteht aus einem Ënecoderünd einem "Decoder"Part, wobei wir verschiedene votrainierte Encoder ausprobieren werden.

Auf einer detaillierten Datenanalyse aufbauend werden dann weitere Methoden angewendet, um die Performance zu verbessern. Es werden Trainigsdaten bereinigt und mit verschiedenen Eingabegrößen experimentiert. Ein weiterer Ansatz ist das intelligente beschneiden der Ausschnitte mit dem Ziel, dass sich das Modell besser auf die Aussagekräftigen Bereiche konzentrieren kann, sowie die Verwendung von "2.5 dimensionalen"Daten, auf die ich später genauer eingehe. Ein weiteres feature ist das intelligente Zuschneiden des Inputs. Zuletzt erfolgt eine Auswertung der Ergebnisse und es werden weitere Ideen genannt, die es nicht in die Arbeit geschafft haben.

Inhaltsverzeichnis

1 Verwandte Arbeiten

1.1 Semantische Segmentierung im medizinischen Bereich

1.1.1 Semantische Segmentierung

Die semantische Segmentierung erfolgt in drei Schritten:

Klassifizierung: Klassifizierung eines bestimmten Objekts im Bild. Lokalisieren: Auffinden des Objekts und Zeichnen eines Begrenzungsrahmens um das Objekt.

Segmentierung: Gruppierung der Pixel in einem lokalisierten Bild durch Erstellung einer Segmentierungsmaske.

Im Wesentlichen kann man die Aufgabe der semantischen Segmentierung als Klassifizierung einer bestimmten Bildklasse und deren Abgrenzung von den übrigen Bildklassen durch Überlagerung mit einer Segmentierungsmaske bezeichnen.

Allgemein kann man es sich auch als Klassifizierung von Bildern auf Pixelebene vorstellen.

1.1.2 Medizinischer Bereich

Die medizinische Bildgebung umfasst viele Kategorien, von 2- bzw. 3 dimensionalen MRT Scans des Herzens, Gehirns und des Magen-Darm Trakts, 2 dimensionalen Röntgen Bildern sowie Ausschnitte aus der Videoendoskopie.

Bereits in den 90er Jahren wurde sich mit mit Segmentierungsarchitekturen auseinandergesetzt und baselines geschaffen. Zu dieser Zeit lieferten Methoden im Bereich „Pattern recogniction“die besten Ergebnisse. Die Autoren legten viel Wert auf preprocessing der Daten und werteten die zu jener Zeit besten Modelle aus. Es handelte sich um einen Feature basierten Ansatz, es wurden Algorithmen wie kNN, Maximum Likelihood und Characteristic vector getestet.

Trotz aller Bemühungen wiesen die Modelle nur eine Accuracy von 3% bis 34% auf. (**Clarke:Mri1995**)

Jedoch waren die Autoren davon überzeugt, dass Segmentierung eine wichtige Rolle im Beriech der Verarbeitung von MRT Daten darstellen wird und mit wachsender Akzeptanz auf klinischer Seite sowie dem technischen Fortschritt der Hochleistungsrechner es in Zukunft möglich sein wird, akkurate Segmentierungen von MRT Scans im Livebetrieb vorzunehmen.

1.2 Deep Learning-Techniken für die Segmentierung medizinischer Bilder: Errungenschaften und Herausforderungen

Dank der immensen Verbesserung der Rechenleistung der Hardwarecomponenten wurde das Thema Deep Learning immer interessanter und gehört heutzutage zur standard Herangehensweise. Architekturen wie das U-Net Modell (**U-Net**) haben sich hierbei besonders bewiesen und wurden mit den Jahren nach der Veröffentlichung in 2015 stetig erweitert, so dass es für viele verschiedene (medizinische) Bereiche Architekturen vorliegen, die auf die jeweiligen Problemstellungen angepasst wurden, jedoch trotzdem Anwendung auf ähnlichen Gebieten finden können (**Hesamian**)

1.3 Intelligentes Zuschneiden des Inputs

Dimitrios G. Zaridis et al. (**SmartCrop**) haben sich ebenfalls mit verschiedenen State of the Art U-Net Architekturen auseinandergesetzt und jene als Baseline für ihre Arbeit genutzt. Trotz der beachtlichen Leistung, die die Modelle liefern, sind sie der Meinung dass noch Verbesserungspotential vorhanden ist.

Sie fanden heraus, dass das Vorhandensein eines Klassenungleichgewichts, wo der Anteil der Hintergrundpixel dem Anteil des zu segmentierenden Organs überwiegt zu Problemen führt. Mithilfe eines Deep Learning Modells haben die Autoren es geschafft, dieses Klassenungleichgewicht zu reduzieren, in dem das Neuronale Netzwerk die gesuchten Organe lokalisiert, das Originalbild zuschneidet und am Ende das Verhältnis von Vorder- und Hintergrundpixeln normalisiert. Dies führte bei allen gängigen Deep Learning Netzwerken zu erheblichen Verbesserung im Bezug des Dice-Scores. U-Net+ und ResU-Net++ wiesen mithilfe diese Verbesserungen von bis zu 8% auf.

2 Grundlagen

2.1 Magen-Darmtrakt

Um die Daten zu verstehen und Erkentnisse aus den Ergebnissen zu gewinnen, ist es wichtig ein Grundverständnis für den Bereich des Körpers zu haben, den unsere Daten beschreiben. Wir interessieren uns in unserem Fall für den Dick- bzw. Dünndarm sowie den Magen. Hierbei liegt die Herausforderung, dass je nach Ernährungsverhalten, Schlafposition, anderen Krankheiten und Verdauung die Position der Organe stark (bzw. stärker als andere) variieren kann. Mithilfe der Grafik kann man sich ein Bild vom groben Aufbau machen. Figure 1

2.2 U-Net

U-Net wurde im Jahr 2015 von Olag Ronneberger et al. (**U-Net**) vorgestellt als Segmentierungsarchitektur für den biomedizinischem Bereich und bildet die Baseline dieser Arbeit. Neu in der Herangehensweise ist der Encoder-Decoder Part, die dem Netzwerk ermöglicht räumliche Merkmale anzutrainieren. Klassisch handelt es sich beim Encoder um ein

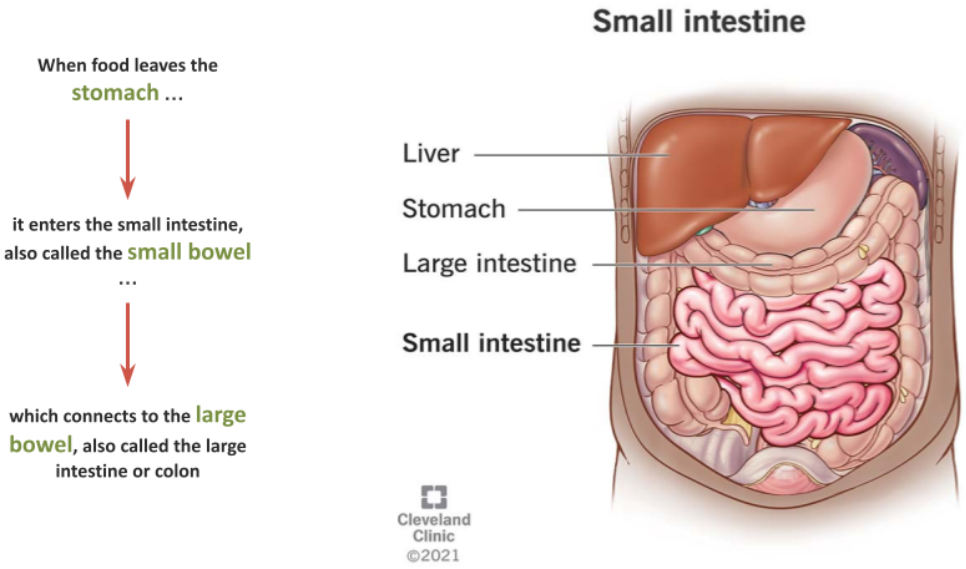

Abbildung 1: Magen-Darm Trakt

Convolutional Neural Network, bestehend es aus 5 bis 6 „downsampling“Schichten und der gleichen Menge an „upsampling“Schichten im Decoder-part.

Eine „downsample“Operation besteht zum einen aus einer „Convolution Operation“sowie einer „Max Pooling“an deren Ende jeweils eine „ReLu“Aktivierungsfunktion zum Einsatz kommt. In diesem Schritt lernt das Modell Eigenschaften auf Pixelebene, wie zum Beispiel Formen, Ecken, Kanten. Hierbei halbieren sich die Eingabedimensionen und die Tiefe des Bildes wird erhöht. Veranschaulicht kann man sagen, dass das Modell das „Was“lernt, was im Bild zu sehen ist. (**Con97**) Zur Semantischen Segmentierung fehlt dann nur noch das „Wo“, hierbei wird das Zusammenspiel von En- und Decoder deutlich.

Im Decoder wird das Bild wieder auf seine ursprüngliche Größe mittels „Transpose Convolution operations“gebracht. Hier ersetzt die transponier Operation die Maxpooling Operation (vergleich Grafik). Dieser Part erlaubt dem Modell das lokalisieren der zuvor gelernten Eigenschaften. Output ist eine Maske, bestehend aus Nullen und Einsen, die der Eingabegröße gleicht und bestenfalls den Wert Eins an der Stelle des gesuchten Organs enthält.

2.3 Encoder 1

Eventuell

2.4 Kaggle

Diese Arbeit baut auf der UW-Madison GI Tract Image Segmentation Challenge auf, die von der „University of Wisconsin Carbone Cancer Center Pancreas Pilot Research Grant “organisiert wurde und von denen die Daten auch bereitgestellt wurden.

Evaluiert wurden die Ergebnisse mit einer Gewichtung von 0.4 des Dice-Koeffizienten (subsubsection 3.5.3) sowie 0.6 der Hausdorff-Metrik (subsubsection 3.5.1) auf unbekannten Testdaten.

3 Methodik

3.1 Datenanalyse

Im Folgenden beziehen sich die Werte X immer auf auf die Vorhersage und Y auf die Groundtruth.

Index	ID	Class	Segmentation
1	case134_day0_slice_0085	large_bowel	NaN
2	case134_day0_slice_0085	small_bowel	41591 5 41599 7 41949 27 ...
3	case134_day0_slice_0085	stomach	NaN
4	case123_day0_slice_0001	large_bowel	35223 6 74352 7 32312 12 ...
5	case123_day0_slice_0001	small_bowel	63432 5 12354 7 41949 12 ...
6	case123_day0_slice_0001	stomach	NaN

Tabelle 1: Beispieldaten für zwei Slices

Der Datensatz besteht aus 115 488 Zeilen und enthält drei features: ID, Klasse und einem Run-lenght codierten String, der die Maske enthält. Table 1. Jeder ID sind drei Zeilen gewidmet, jeweils für die drei Klassen: small_bowel, large_bowel und stomach. Zu jeder ID existiert ein graustufen Bild, welche sich im train Ordner befinden, Ein Beispiel der Ordnerhierarchie kann man hier sehen Figure 4.

input\uw-madison-gi-tract-image-segmentation\train\case101\
case101_day20\scans\slice_0001_266_266_1.50_1.50.png

Jedes slice enthält vier Zahlen (z.B. 266_266_1.50_1.50.png), die ersten beiden stehen für die Auflösung des Bildes und die letzten beiden für den physischen Abstand der Pixel. Der Großteil der Aufnahmen stammt von Tag null oder Tag eins Figure 7 und die durchschnittliche Anzahl an Bildern pro Fall beträgt X.

Beim betrachten der Verteilung der Segmentierungen fällt auf, dass das Vorkommen für jede Klasse stark variiert, am häufigsten sind gar keine segmentierten Slices. Figure 5.

3.2 Metadatenextraktion

Anhand der ID eines jedes Slices war es möglich verschiedene Metadaten dem originalem Dataframe zu hinzuzufügen, zudem wurde die Gesamtlänge gedrittelt, dadurch dass die Klassen einer ID zugewiesen worden sind, die Gesamtlänge des Dataframes beläuft sich somit auf 38 496 Zeilen. Table 2

Idx	ID	large_bowel	small_bowel	stomach	case	day	slice	path	width	height	pixel_x	pixel_y
1	case134_day0_slice_0085	NaN	41591 5 ...	NaN	134	0	85	input\	266	266	1.5	1.5
2	case134_day0_slice_0086	41591 27 ...	NaN	NaN	134	0	86	input\	266	266	1.5	1.5
3	case134_day0_slice_0086	NaN	NaN	NaN	134	0	87	input\	266	266	1.5	1.5

Tabelle 2: Metadaten von drei slices

3.3 Vorarbeit

3.3.1 2.5 dimensionale Daten

Klassisch haben die Slices die Dimensionen

$$H \times W \times 1 \tag{1}$$

da es sich um graustufen Bilder handeln. U-Net arbeitet klassisch aber mit RGB Bildern, also einem 3 Channel Input

$$H \times W \times 3 \tag{2}$$

somit ist die Idee, aufeinanderfolgende Slices mit einem bestimmten Stride (Abstand) zu Verketten und dem Input so mehr Tiefe zu geben.

Diese Methode hat gegenüber 3 dimensionalen Daten einige Vorteile: es wird weniger Rechenleistung benötigt, die Trainingspipeline muss nur minimal angepasst werden, die Auswertung bleibt ebenfalls gleich und die Daten können jedem Modell übergeben werden, welche darauf ausgelegt sind RGB Bilder zu segmentieren.

3.3.2 Intelligentes Zuschneiden

Basierend auf den Ergebnissen von Zaridis et al. sowie dem Vorhandensein eines Klassen-Ungleichgewichts implementierte ich einen Algorithmus, in der Hoffnung eine Verbesserung der Performance des Models zu erreichen.

Die Eingabebilder wurden für jeden Tag pro Case individuell zugeschnitten. Zu beginn wird das Hintergrundrauschen mit einem Medianfilter beseitigt, um das kleinste Rechteck aus der Menge zu finden, welches den Wertebereich enthält, der ungleich null ist. Dieser Prozess wird für jedes Slice aus dem Tag wiederholt, zum Schluss werden die Werte der Zeile angehangen, im Dataloader werden die Slices je nach Konfiguration dann zugeschnitten. Ein Beispiel von Case 134, Tag 21, Slice 70 ist hier zu sehen: Table 3

ID	Case	Day	RS	RE	CS	CE
1	134	21	84	10000	0	359
2	113	22	50	10000	0	332

Tabelle 3: Ausschnitt aus der Zuschnitt-Tabelle

RS, RE, CS und CE stehen in diesem Kontext für Zeilenanfang und Ende sowie Spaltenanfang und Ende.

So konnte das Ungleichgewicht für jede Klasse gesunken werden (Werte wurden gerundet):

Klasse	ohne Zuschnitt	mit Zuschnitt
large_bowel	144	134
small_bowel	158	146
stomach	286	265
Total	60	56

Tabelle 4: Verhältnis von Vorder- zu Hintergrundpixeln

3.3.3 Bereinigen der Daten

Zwei Cases wurden falsch annotiert und wurden deshalb aus dem Trainingsdatensatz entfernt, somit beläuft sich die Anzahl der Zeilen auf 38 208: Figure 13

3.4 Nacharbeit

Bestimmte Slices weisen nie eine Segmentierung auf. Dieses Wissen kann man dazu nutzen, um die Vorhersage zu bereinigen, indem man die Maske jener Slices entfernt. Mithilfe dieser Methode kann man seinen Score bei Kaggle um e-3 bis e-4 verbessern. Ob dies jedoch good practice im Klinikalltag ist, ist fragwürdig, da diese Beobachtungen auf diesem Datensatz gemacht wurden und es wahrscheinlich ist aufgrund von z.B. einer anderen Statur, Operationen oder ähnlichem dies bei anderen Patienten nicht garantieren kann,

3.5 Algorithmik

3.5.1 Hausdorff-Metrik

Die Hausdorff-Metrik ist eine Methode zur Berechnung des Abstands zwischen den Segmentierungsobjekten X und Y, indem der am weitesten entfernte Punkt auf Objekt X vom nächstgelegenen Punkt auf Objekt Y berechnet wird

$$d_{\mathrm{H}}(X,Y) = \max\left\{\sup_{x\in X} d(x,Y), \sup_{y\in Y} d(X,y)\right\} \tag{3}$$

3.5.2 Binäre Kreuzentropie

Die Kreuzentropie ist definiert als ein Maß für die Differenz zwischen zwei Wahrscheinlichkeitsverteilungen für eine bestimmte Zufallsvariable oder eine Reihe von Ereignissen. Sie eignet sich besonders gut Klassifizierungen auf Pixelebene, weist jedoch bei einer unausgewogenen Klassenverteilung der Daten ihre schwächen auf. (**Jadon_2020**)

Sie ist definiert als:

$$\mathcal{L}_{BCE}\left(X_{il}, Y_i l\right) = \sum_{c=1}^{C} X_{icl} \log\left(Y_{icl}\right) \tag{4}$$

3.5.3 Dice-Koeffizient

Der Dice-Koeffizient kann verwendet werden, um die pixelweise Übereinstimmung zwischen einer vorhergesagten Segmentierung und der entsprechenden Groundtruth zu vergleichen. Die Formel ist gegeben durch:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{5}$$

wobei X die vorhergesagte Menge von Pixeln und Y die Groundtruth ist. Der Dice-Koeffizient ist definiert als 0, wenn sowohl X als auch Y leer sind.

3.6 Modell

Das Projekt wurde gänzlich in Python mithilfe der TensorFlow API sowie dem Segmentation Models (**Yakubovskiy:2019**) Package aufgezogen, welches verschiedene Architekturen inklusive an- und nicht antrainierte Encoder zur verfügung stellt.

Trainiert wurde mit dem Adam Optimizer, und einer initialen Learning Rate von von 5e-4, die sich um den Faktor e-1 verringert, wenn der Validation-loss nach fünf Epochen unverändert schlecht blieb.

Die Loss function ist eine Mischung aus Binary Crossentropy und zu einem Faktor von 0.5 der Dice-Koeffizient. Diese Kombination eignet sich laut Jadon (**Jadon_2020**) besonders gut, um einen höheren Dice Score zu erzielen. Außerdem ermöglicht sie dem Modell, sich auf mehrere Klassen zu konzentrieren, so ist das Modell in der Lage das Klassenungleichgewicht zu berücksichtigen.

$$\mathcal{L}\left(X_{il}, Y_i l\right) = 0.6 * \sum_{c=1}^{C} X_{icl} \log\left(Y_{icl}\right) + 0.4 * \left(1 - \frac{2\left|X_{il} \cap Y_i l\right|}{|X_{il}| + |Y_i l|}\right) \tag{6}$$

Es wurde mit einer Batchsize von 16 und 32 auf verschiedenen Grafikkarten vom HPC des ZIMs Trainiert: RTX 8000, RTX 6000 und GTX 1080ti.

Die Daten wurden mithilfe von der StratifiedGroupKFold Methode von Sklearn in 5 Folds aufgeteilt nach Case gruppiert mit einem festem Seed von 42. Somit wurde das Verhältnis der Klassendistribution im Vergleich zum ganzen Datensatz in jedem Fold beibehalten.

3.7 Experimente

Zunächst wurde eine Baseline mit folgenden Standardkonfigurationen geschaffen:

Input	Batchsize	Epochen	Datenbereinigung
256x256x1	16	50	Wahr

Tabelle 5: Baseline Einstellungen

Zunächst wurden mit diesen Konfigurationen 12 verschiedene Modelle trainiert, wessen Encoder auf ImageNet angelernt wurden.

Encoder	Beste Epoche	Loss	Val. loss	Dice	Val. Dice	IoU	Val. IoU	Privat	Öffentl.
efficientnetb7	14	0.0500	**0.1064**	0.8848	**0.7636**	0.8153	0.8223	0.8293	**0.8467**
efficientnetb2	22	0.0369	0.1123	0.9146	0.7489	0.9037	0.8195	0.8278	0.8434
efficientnetb1	18	0.0425	0.1158	0.9018	0.7386	0.8889	0.8179	0.8266	0.8416
efficientnetb5	25	0.0369	0.1160	0.9146	0.7378	0.9089	0.8161	0.8315	0.8457
inceptionv3	30	0.0381	0.1186	0.9118	0.7326	0.8937	0.8108	0.8201	0.8385
efficientnetb3	22	0.0382	0.1181	0.9117	0.7324	0.8958	0.8206	0.8235	0.8434
densenet121	33	0.0385	0.1198	0.9109	0.7300	0.8873	0.8093	0.8179	0.8346
efficientnetb6	19	0.0401	0.1194	0.9073	0.7277	0.8925	**0.8240**	**0.8357**	0.8463
efficientnetb4	35	**0.0317**	0.1211	**0.9267**	0.7266	0.9235	0.8147	0.8243	0.8426
resnet50	35	0.0324	0.1244	0.9250	0.7206	**0.9259**	0.7974	0.8081	0.8278
densenet201	43	0.0383	0.1553	0.9114	0.6406	0.8942	0.7893	0.8176	0.8322
efficientnetb0	21	0.0387	0.1641	0.9104	0.6173	0.8804	0.7864	0.8227	0.8406

Tabelle 6: Baseline Performances nach Validation loss absteigend sortiert.

Hier sticht efficientnetb7 klar hervor mit großem Abstand, wenn man den Validation loss betrachtet.

Encoder	Größe	Beste Epoche	Loss	Val. loss	Dice	Val. Dice	IoU	Val. IoU	2.5D	Crop	Batch Size	Epochen
efficientnetb2	256	49	0.0245	0.1032	0.9434	0.7727	0.9406	0.8177	False	True	32	50
efficientnetb2	256	46	0.0252	0.1052	0.9419	0.7651	0.9394	0.8289	True	True	32	50
efficientnetb2	256	27	0.0323	0.1391	0.9251	0.6798	0.9121	0.8130	False	False	32	50

Tabelle 7: Baseline Performances nach Validation loss absteigend sortiert.

-> iou score für predictions vom validation set berechnen und beste + schlechteste masken plotten + score bei kaggle

-> efb2 und dense121

-> die werden erneut trainiert mit bereinigten Daten, 25D Daten und / oder zugeschnittenen Daten

-> inputsize erhöht

-> iou score für predictions vom validation set berechnen und beste + schlechteste masken plotten + score bei kaggle

-> was hat wo was gebracht (25d daten, zuschnitt, datenreinigung)

4 Evaluation

5 Fazit

5.1 Ausblick

Ideen, die es nicht in die Arbeit geschafft haben oder nicht schafffen konnten.

quotes:
Ein Beispiel für deutsche Anführungszeichen „quote“.

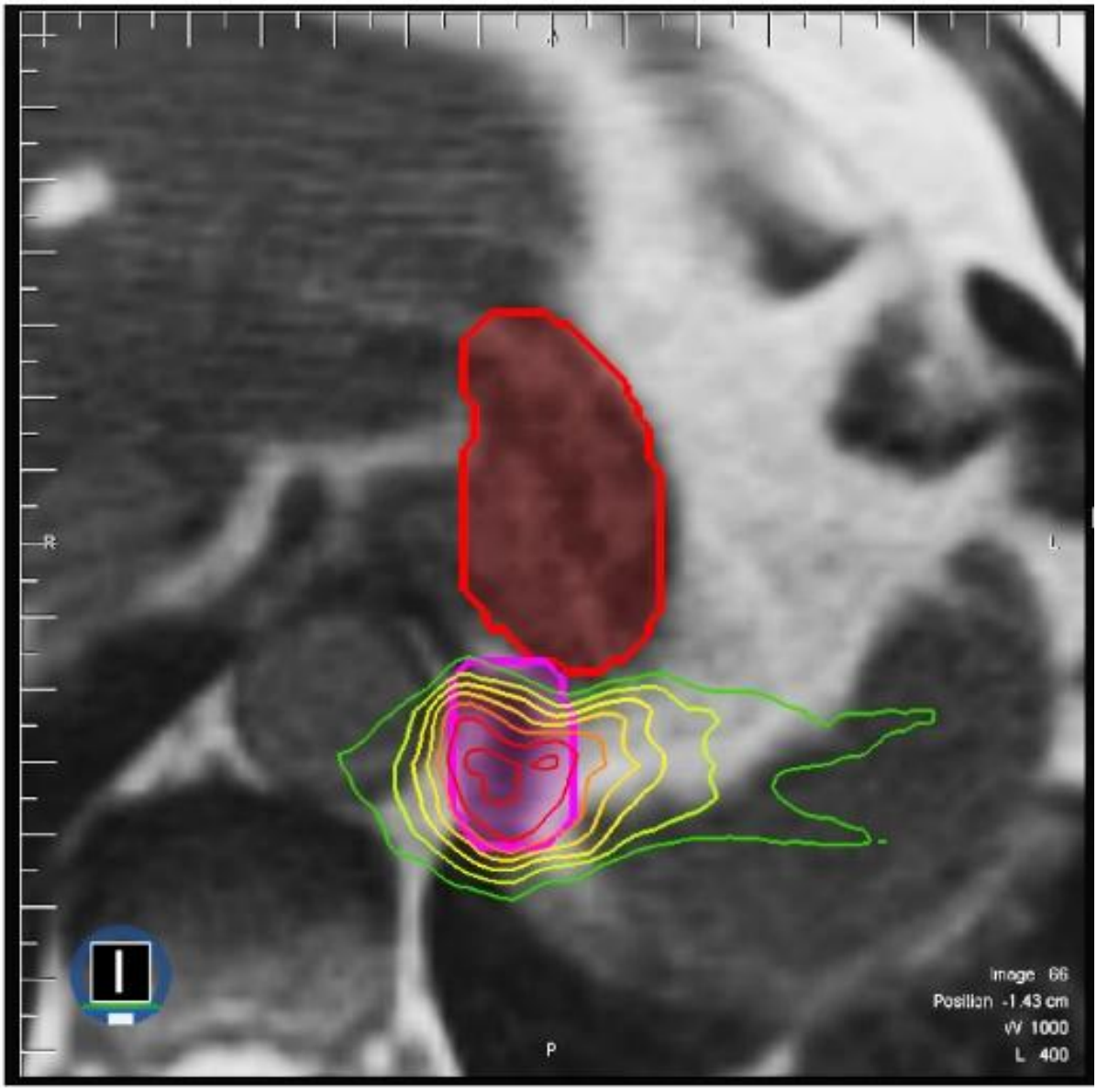

Abbildung 2: Beispiel MRT des Magen-Darm Traks. Zu sehen ist der Magen (rot) und der Tumor (pink) sowie die verabreichte Strahlendosis. Die Strahlenintensivität werden durch den Regenbogen der Umrisse dargestellt, wobei höhere Dosen rot und niedrigere Dosen grün dargestellt werden.

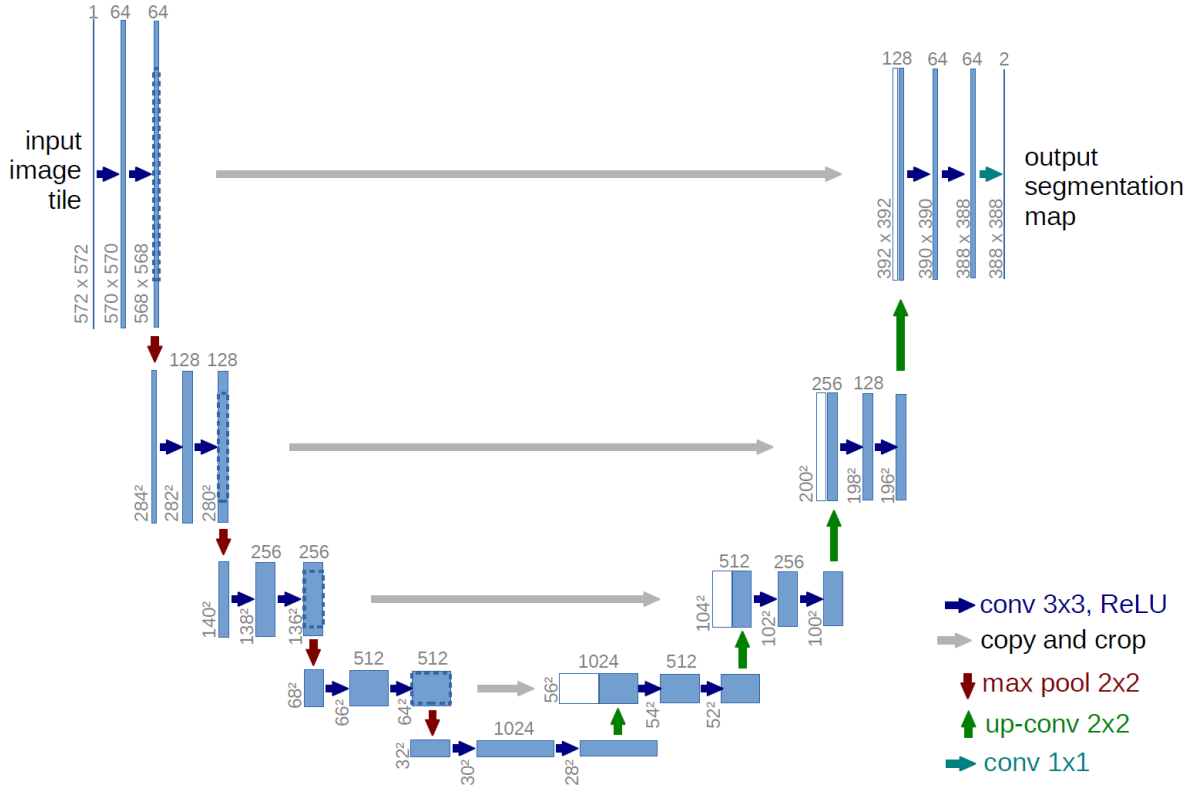

Abbildung 3: U-Net Architektur

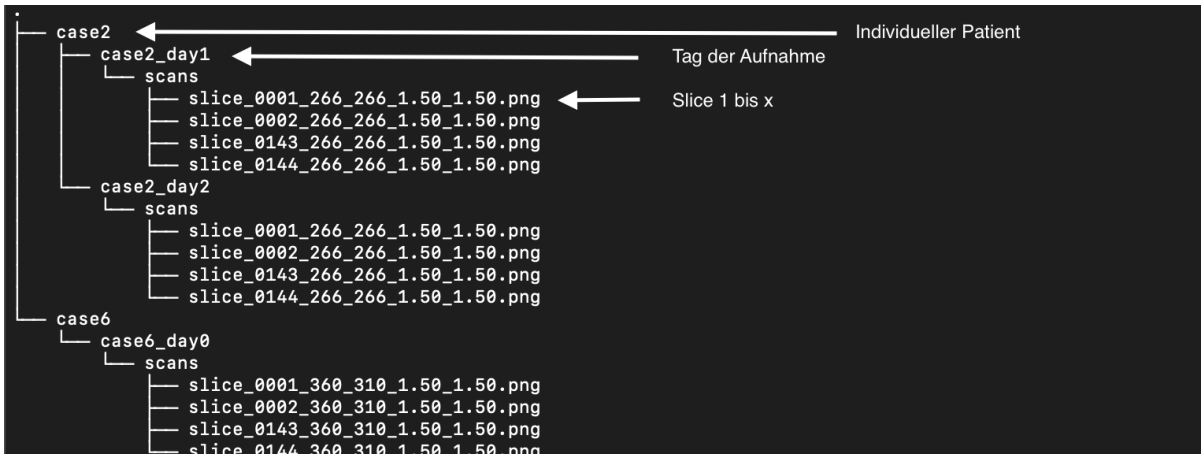

Abbildung 4: Trainingsdaten Hierachie

Abbildung 5: Verteilung der Klassen im Testset

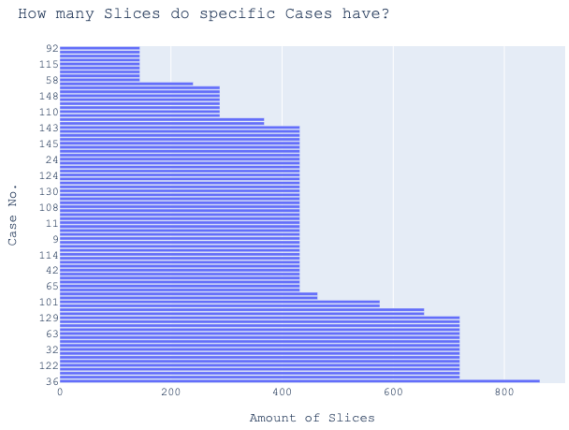

Abbildung 6: Anzahl Bilder pro Fall

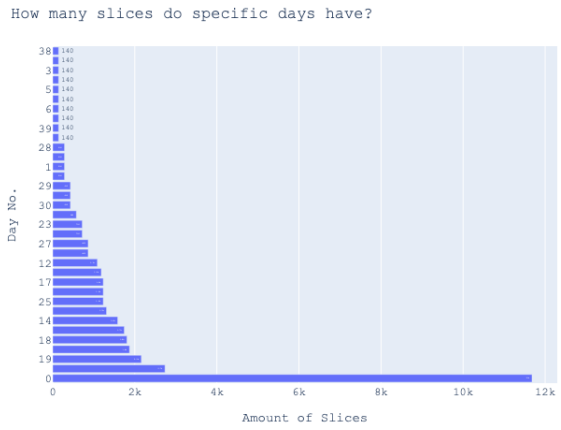

Abbildung 7: Anzahl Bilder pro Tag

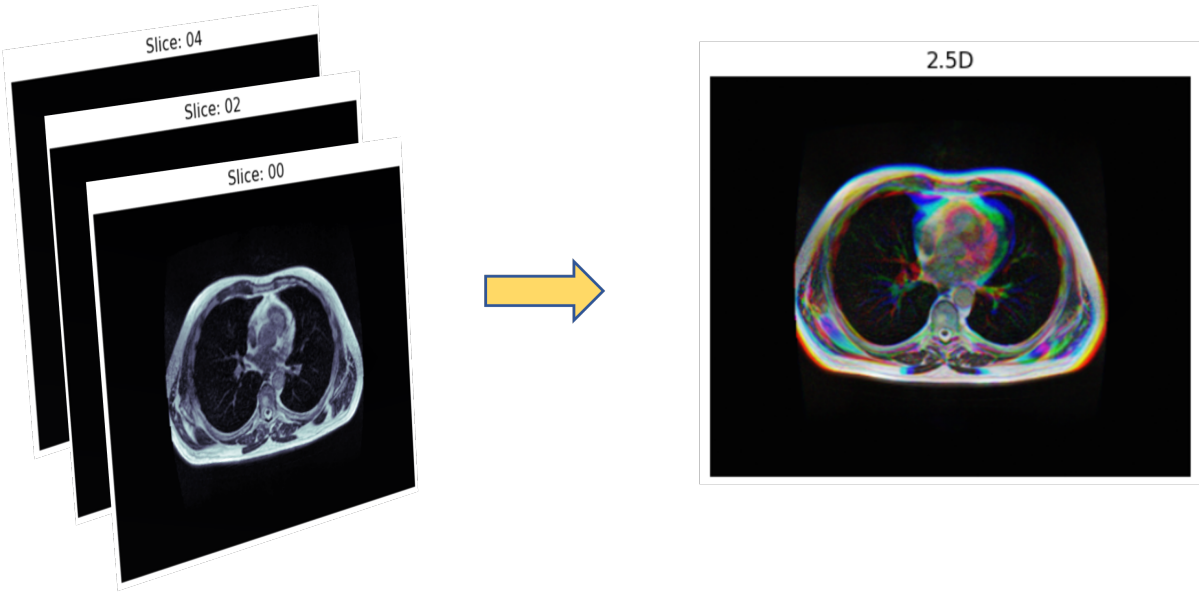

Abbildung 8: Semi-3-dimensionaler Input

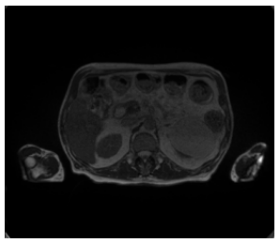

Abbildung 9: Original

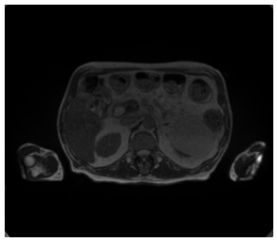

Abbildung 10: Zugeschnitten

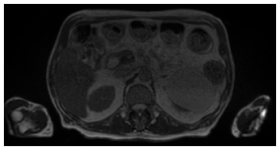

Abbildung 11: Zugeschnitten mit Medianfilter

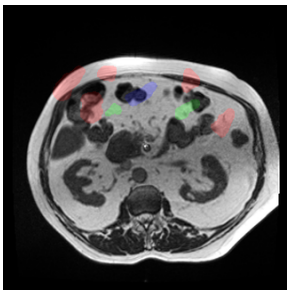

Abbildung 12: Case 7, Tag 0, Slice 96

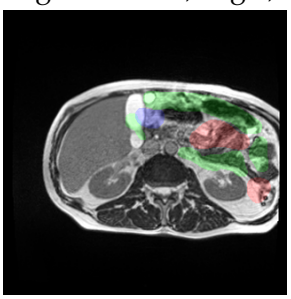

Abbildung 13: Case 81, Tag 30, Slice 96

Abbildungsverzeichnis

Tabellenverzeichnis