UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR MEDIZINISCHE INFORMATIK

CS4332: Modell- und KI-basierte Bildverarbeitung in der Medizin - Praktikum

SoSe 2025

Übungszettel 1

Organisatorisches:

Die Aufgaben sind selbstständig in Zweiergruppen zu bearbeiten. Zur Lösung der Aufgaben benötigen Sie einen Google-Drive-Account. Die Bearbeitung der Notebooks erfolgt dann via Google Colaboratory.

Die für die Bearbeitung benötigten Code-Skeletons sind über folgende Links abrufbar (alternativ finden Sie die Dateien zum Download in Moodle):

https://colab.research.google.com/drive/1JaMGg8f66_zrUsaRvSO7oS6L4Ontj-5b?usp=sharing

https://colab.research.google.com/drive/1D4eAobHBZ-EU9CWMCmej6nEJ9mh8QkgO?usp=sharing

Legen Sie bitte eine **Kopie der Notebooks in Ihrem Google-Drive-Ordner** an:

- Entweder: Öffnen der Notebooks in Google Colaboratory, „In Google Drive kopieren“
- Oder: Download der Notebooks, Hochladen aus Google Drive über „+ Neu“.

Anschließend können Sie die Notebooks mit Google Colaboratory bearbeiten.

Die Abgabe der Aufgaben erfolgt über Moodle. Benennen Sie dazu das Notebook, das Sie abgeben möchten, nach folgendem Schema:
`abgabe_praktikum1_aufgabeX_nachname1_nachname2.ibynb`

Abgabetermin für diesen Übungszettel: **19.05.2025, 16:00 Uhr.**

Praktikumsorganisation/-durchführung: Julia Andresen

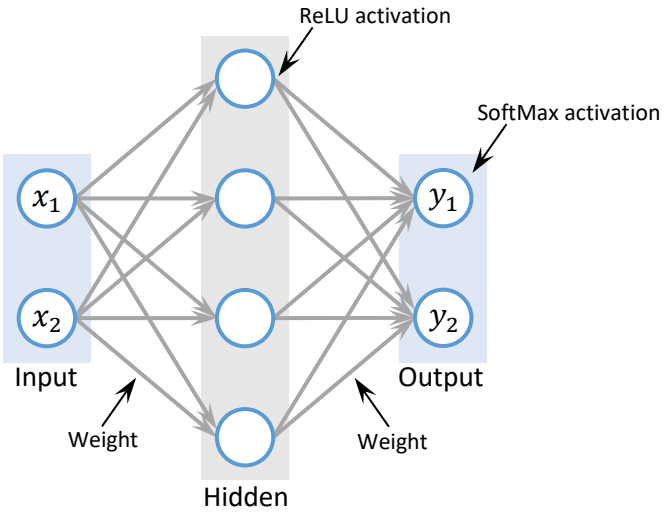
Für organisatorische Fragen: j.andresen@uni-luebeck.de

Aufgabe 1: Minimales XOR-Netzwerk (10 P.)

Öffnen Sie das Notebook `PraktUe1_Aufg1.ipynb` und wählen Sie als Hardwarebeschleuniger GPU aus (→ Laufzeit → Laufzeittyp ändern → Hardwarebeschleuniger GPU).

In dieser Aufgabe entwickeln Sie ein erstes neuronales Netzwerk in PyTorch, das das XOR-Problem löst. Die Definition des XOR-Problems ist in der Tabelle gegeben. Rechts daneben sehen Sie das zu entwickelnde Netzwerk.

Eingabe x_1	Eingabe x_2	Ausgabe y
0	0	0
0	1	1
1	0	1
1	1	0



Das Netzwerk soll **zwei vollverbundene Schichten** haben und **4 versteckte Neuronen**.

Die Eingaben sind x_1 und x_2 und die Ausgabe $y = [y_1, y_2]$. Dabei entspricht y_1 der Wahrscheinlichkeit, dass XOR(x_1, x_2) = 0 ist. Analog entspricht y_2 der Wahrscheinlichkeit, dass XOR(x_1, x_2) = 1 ist.

Nach der ersten vollverbundenen soll eine **ReLU-Aktivierung** stattfinden.

Es soll eine **BinaryCrossEntropy-Funktion** als Loss benutzt werden, diese fügen Sie in der Main-Methode hinzu. Vor der Lossfunktion soll eine **SoftMax-**Aktivierung benutzt werden.

Außerdem müssen Sie sich für einen **Optimizer** entscheiden (`torch.optim`), Sie können ein paar unterschiedliche ausprobieren.

Ein weiterer zu wählender Parameter ist eine sinnvolle **Anzahl von Trainingsepochen**.

Hinweis: Die Dokumentation von PyTorch https://pytorch.org/docs/stable ist sehr hilfreich für diese Aufgabe.

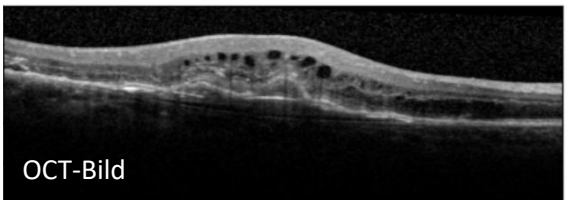
Als Bonusaufgabe können Sie das Netzwerk auf der Grafikkarte trainieren, indem Sie das Netz, die benötigten Tensoren und Funktionen mit `.cuda()` auf die GPU verlagern.

Wie wirkt sich das aus auf die Zeit pro Epoche? Wie erklären Sie sich das?

Aufgabe 2: Einführung in die Bildverarbeitung (20 P.)

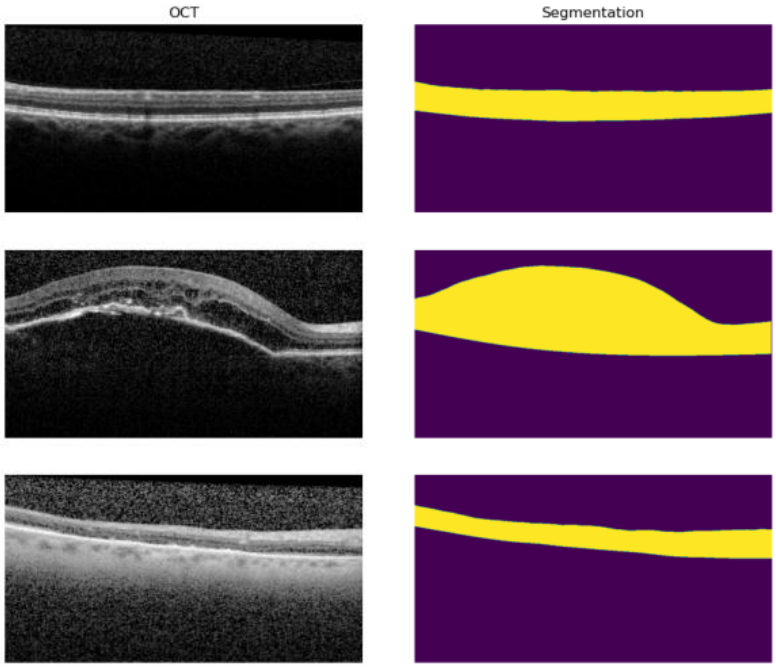
Öffnen Sie das Notebook `PraktUe1_Aufg2.ipynb`.

a) Das Ziel dieser Teilaufgabe ist die Bestimmung des Retinavolumens in einem optischen Kohärenztomographie-Bild (OCT-Bild):



1. Wenden Sie das **Schwellenwertverfahren** auf das OCT-Bild an, um eine erste einfache Segmentierung der Retina zu erzeugen. Plotten Sie das Ergebnis.
2. Berechnen Sie den **DICE-Score** zwischen der manuellen Segmentierung und der Segmentierung, die aus dem Schwellenwertverfahren resultiert. Versuchen Sie anschließend, einen Schwellenwert zu finden, der einen möglichst hohen DICE-Score liefert.
3. Die intraretinalen Fluide (schwarze Punkte innerhalb der Retina) werden durch das Schwellenwertverfahren dem Hintergrund zugeordnet. Gleichzeitig kann es passieren, dass durch das Rauschen im Bild einzelne Hintergrundpixel der Segmentierung zugeordnet werden. Versuchen Sie, den DICE-Score zu verbessern, indem Sie die Löcher in der Segmentierung schließen/verkleinern und sehr kleine segmentierte Bereiche entfernen. Nutzen Sie dazu **morphologische Operationen**.
4. Nutzen Sie Ihre Segmentierung, um das **Volumen** der Retina in Kubikmillimetern zu **berechnen**.
   Hinweis: Typischerweise decken dreidimensionale OCT-Bilder ein Sichtfeld von 6mm x 2mm x 6mm (WxHxT) ab, wobei 25 zweidimensionale Querschnitte aufgenommen werden. Ein solcher Querschnitt ist oben dargestellt.

5. Schauen Sie sich die folgenden Aufnahmen an:



Würde der von Ihnen implementierte Segmentierungsalgorithmus auch für diese OCT-Bilder funktionieren? Begründen Sie Ihre Antwort.
Hinweis: Sie können Fragen als Kommentar im Code oder als Text-Snippet beantworten.

b) Zur Vorbereitung auf das Training von CNNs sollen Sie in dieser Aufgabe verschiedene Bildaugmentationstechniken implementieren. Ergänzen Sie in der Klasse `Image` die folgenden Funktionen:

- `normalize(self, min_value=0, max_value=255)` – Diese Funktion soll die Bildintensitäten linear auf das Intervall [`min_value`, `max_value`] abbilden. Ihre Implementierung sollte unabhängig vom Intensitätswertebereich des Eingangsbildes funktionieren.
- `flip(self)` – Hier soll das Bild horizontal gespiegelt werden.
- `shift(self, amount=10)` – Diese Funktion soll das Bild um `amount` Pixel vertikal nach oben verschieben. Dabei darf sich die Bildgröße **nicht** verändern.
- `center_crop(self, new_size=[20, 20])` – Diese Funktion soll das Bild symmetrisch um das Zentrum zuschneiden, so dass die neue Größe `new_size` ist.

Achten Sie bei allen Funktionen darauf, dass Sie ggf. die relevanten Klassenvariablen anpassen.

Wenden Sie anschließend Ihre neu implementierten Funktionen sukzessive auf das `cameraman`-Bild an und plotten Sie die Ergebnisse.