UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR MEDIZINISCHE INFORMATIK

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

SoSe 2025

Übungszettel 3

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/10ViE9Us3aW5JXZQAyGPAZ7KI9Yp8knFJ?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_praktikum3_aufgabeX_nachname1_nachname2.ibynb`

Abgabetermin für diesen Übungszettel: **16.06.2025, 16:00 Uhr.**

Praktikumsorganisation/-durchführung: Julia Andresen

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

Öffnen Sie das Notebook `PraktUe3.ipynb` und wählen Sie als Hardwarebeschleuniger GPU aus (→ Laufzeit → Laufzeittyp ändern → Hardwarebeschleuniger GPU).

Das Ziel dieser Programmieraufgabe ist die Implementierung, das Training und die ausführliche Evaluation eines CNNs zur Klassifikation von Röntgen-Thorax-Aufnahmen. Außerdem sollen Sie die Entscheidungen ihres Netzwerkes mit GradCAM visualisieren und Ihr Netzwerk fine-tunen.

Aufgabe 1: Klassifikations-CNN (8 P.)

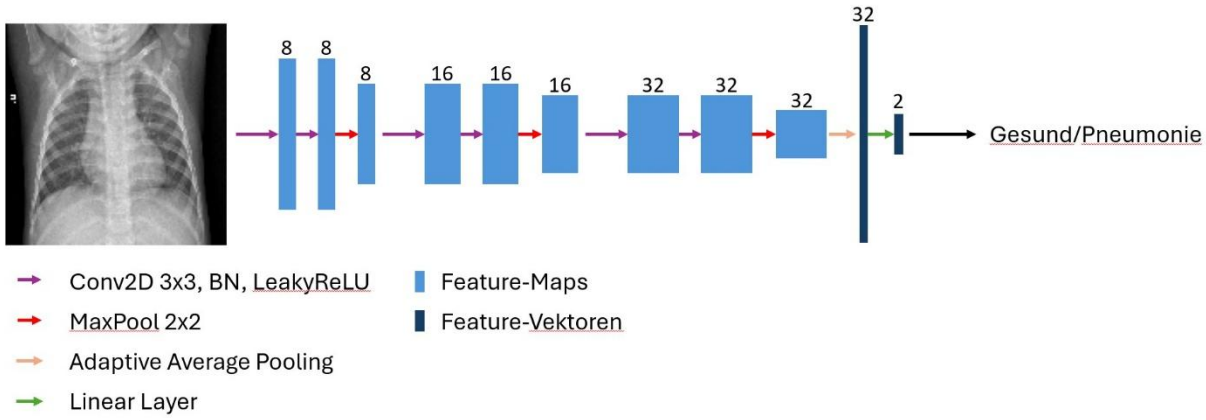
a) Vervollständigen Sie die Klasse `ConvBlock()`. Der Faltungsblock soll zwei Layer-Gruppen beinhalten, bestehend aus:

- `nn.Conv2d` (Kernelsize von 3, Padding von 1 und kein Bias)
- `nn.BatchNorm2d` (Kernelsize und Stride 2)
- `nn.LeakyReLU`

Die erste Gruppe soll die Anzahl an Feature-Maps verdoppeln, die zweite behält die Anzahl an Feature-Maps bei.

Abschließend soll eine Maximum-Pooling vorgenommen werden.

b) Implementieren Sie nun das Klassifikations-CNN. Halten Sie sich dabei an folgende Architektur:



Nutzen Sie für die Implementierung der drei Faltungsblöcke den in a) erstellten ConvBlock.

Aus dem letzten Faltungsblock resultieren 32 Merkmalskarten. Diese sollen mit Adaptive-Averaging-Pooling in einen Merkmalsvektor der Länge 32 überführt werden. Jede Merkmalskarte wird also in einen skalaren Wert überführt.

Anschließend folgt eine vollverbundene Schicht, die aus den 32 Fetures die Klassifikation in gesund oder Pneumonie vornimmt.

c) Nutzen Sie `print()` und `torchinfo.summary()`, um sich die Architektur Ihres Netzes ausgeben zu lassen. Ihr Netz sollte 18218 trainierbare Parameter haben.

d) Trainieren Sie Ihr Modell für die Klassifikation in gesund und Pneumonie. Verwenden Sie als Loss `nn.CrossEntropyLoss()` mit Gewichten, die die Klassenimbalance im gegebenen Datensatz ausgleichen.

Aufgabe 2: Testen des Modells (6 P.)

Nutzen Sie `sklearn.metrics`, um folgende Metriken auf den Testdaten auszuwerten:

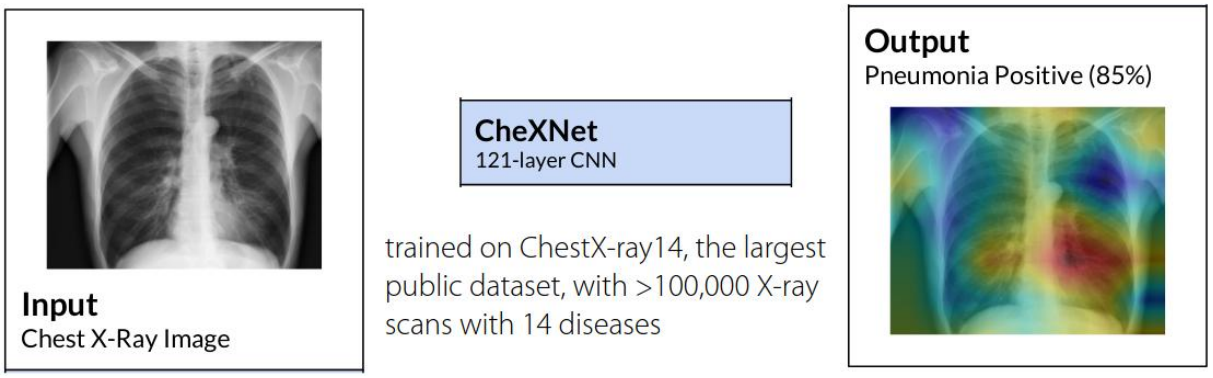
- Accuracy
- Precision
- Recall
- F1-Score

Stellen Sie außerdem die Klassifikationsergebnisse in einer Konfusionsmatrix dar.

Wie bewerten Sie die Performance ihres Modells?

Aufgabe 3: Visualisierung der Modellentscheidungen mit GradCAM (6 P.)

GradCAM ist eine Visualisierungstechnik zur Erklärung von Modellentscheidungen. Dabei wird eine Heatmap erstellt, die die für eine Entscheidung relevanten Bildregionen hervorhebt:



GradCAM kommt erst später in der Vorlesung, aber funktioniert grob wie folgt:

1. Input: Ein Bild wird durch ein CNN geschickt
2. Zielklasse: Man wählt eine Zielklasse (z.B. Pneumonie), für die man wissen will, was das Modell „gesehen“ hat
3. Gradientenberechnung: Man berechnet die Gradienten der Ausgabe der Zielklasse bezüglich der Feature-Maps einer bestimmten Faltungsschicht

4. Gewichtung: Man mittelt die Gradienten über die Höhe und Breite jeder Feature-Map, um Gewichte zu erhalten. Diese Gewichte zeigen, wie wichtige eine Feature-Map für die Zielklasse ist
5. Lineare Kombination: Man bildet die gewichtete Summe der Feature-Maps
6. ReLU: Negative Werte werden durch ReLU entfernt, da man nur die Einflüsse haben will, die positiv für die untersuchte Klasse wirken
7. Upscaling: Das Ergebnis wird auf die ursprüngliche Bildgröße hochskaliert, um die Heatmap zu erhalten

Implementieren Sie GradCAM, um die Entscheidungen Ihres Netzes zu veranschaulichen. Der Code führt Sie Schritt für Schritt durch die oben beschriebenen Punkte. Plotten Sie abschließend für zehn Testbilder die GradCAM-Heatmaps als Overlay über den Röntgen-bildern für beide Ausgabeklassen.

Wie interpretieren Sie die Heatmaps?

Aufgabe 4: Fine-Tuning (5 P.)

Versuchen Sie, Ihre Ergebnisse weiter zu verbessern, indem Sie

- mindestens eine Architekturveränderung vornehmen (mehr/weniger Schichten, Dropout, …)
- die Lernrate verändern
- mindestens einen weitere Hyperparameter anpassen (z.B. die Anzahl an Epochen, die Batch-Größe, …).

Dokumentieren Sie Ihre Ergebnisse. Welche Konfiguration liefert die beste Performance?

Hinweis: Sie können Fragen als Kommentar im Code oder als Text-Snippet beantworten.