



Automatische Klassifikation von Thorax-Röntgenbildern zur Erkennung von Pneumonie und Tuberkulose

Projektbericht

Gruppe:

Felix Zauner, Timofey Luzin

Klasse & Schuljahr:

5AHETS 2025/26

Abgabe:

15.12.2025

Inhaltsverzeichnis

1	Einleitung	2
2	Datensatzbeschreibung	3
3	Datenaufbereitung	4
4	Modellauswahl und -training	5
4.1	Modellauswahl	5
4.2	Modelltraining	6
4.3	Code-Dokumentation	8
4.4	Probleme und Herausforderungen	10
5	Evaluation des Modells	11
6	Probleme und Herausforderungen im gesamten Projekt	14
6.1	Technische Herausforderungen	14
6.2	Modellierungsprobleme	14
7	Schlussfolgerung und Reflexion	15
8	Quellen und Literaturverzeichnis	17
9	Anhang	18

1 Einleitung

In diesem Projektbericht wird die Entwicklung eines Modells zur automatischen Klassifikation von Thorax-Röntgenbildern zur Erkennung von Pneumonie und Tuberkulose beschrieben.

Die KI-basierte Bildanalyse hat in den letzten Jahren erhebliche Fortschritte gemacht und bietet vielversprechende Möglichkeiten zur Unterstützung medizinischer Diagnosen. Manche Experten sehen in der automatischen Bildanalyse sogar das Potenzial, die Genauigkeit und Effizienz von Diagnosen zu verbessern, insbesondere in ressourcenarmen Umgebungen.

Ziel dieses Projekts ist es, ein Modell zu entwickeln, das in der Lage ist, Thorax-Röntgenbilder zu analysieren und zwischen gesunden Patienten, Patienten mit Pneumonie und Patienten mit Tuberkulose zu unterscheiden.

Der Bericht gliedert sich in mehrere Abschnitte, die den gesamten Entwicklungsprozess abdecken, von der Datensammlung und -aufbereitung über die Modellauswahl und das Training bis hin zur Evaluation des Modells. Abschließend werden die Herausforderungen und Probleme, die während des Projekts aufgetreten sind, sowie eine Reflexion über die Ergebnisse und mögliche zukünftige Verbesserungen diskutiert.

2 Datensatzbeschreibung

Der Datensatz beinhaltet insgesamt 25.600 Thorax-Röntgenbilder, die in drei Klassen unterteilt sind: Gesunde Patienten, Patienten mit Pneumonie und Patienten mit Tuberkulose. Die Bilder sind schon in Trainings-, Validierungs- und Testsets aufgeteilt. In unserer Anwendung verwenden wir *ausschließlich* das *Trainingsset* und das *Testset*.

Die Bilder liegen im JPEG-Format vor und haben keine standardisierte Auflösung. Hier ist eine Übersicht über die Struktur des Datensatzes dargestellt:

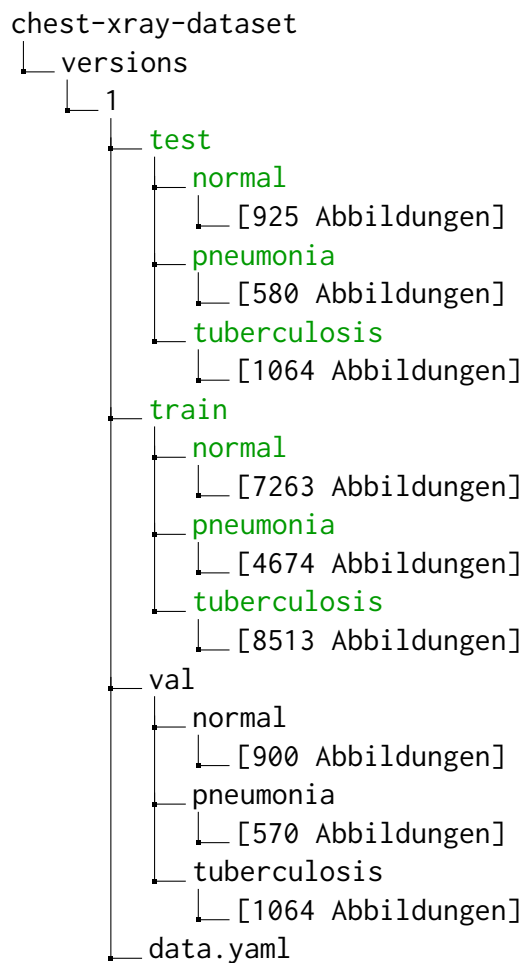


Abbildung 1: Struktur des Datensatzes

Das Dataset wurde von Kaggle bereitgestellt und ist öffentlich zugänglich unter folgendem Link:

<https://www.kaggle.com/datasets/muhammadrehan00/chest-xray-dataset>

3 Datenaufbereitung

Aufgrund der unterschiedlichen Auflösungen der Bilder im Datensatz war eine sorgfältige Datenaufbereitung erforderlich, um eine konsistente Eingabe für das Modell zu gewährleisten.

Die Bilder wurden sowohl auf eine einheitliche Größe von 128x128 Pixel skaliert, als auch in schwarz-weiß umgewandelt, um die Verarbeitung zu erleichtern und die Leistung des Modells zu optimieren.

Dies wurde mit der OpenCV-Bibliothek in Python durchgeführt.

Das folgende Code-Snippet zeigt die Implementierung der Bildskalierung und der Umwandlung in Graustufen:

```
1 import cv2
2 # Transform and load images
3 def load_images(folder, label, img_size=(128, 128)):
4     X, y = [], []
5
6     for file in os.listdir(folder):
7         path = os.path.join(folder, file)
8         img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
9         img = cv2.resize(img, img_size)
10        X.append(img)
11        y.append(label)
12
13    return X, y
```

Listing 1: Image Resizing und Grayscale Conversion

4 Modellauswahl und -training

4.1 Modellauswahl

Zu Beginn des Projektes wurde der bereits im Exposé genannte MLP Classifier implementiert und trainiert.

Nach ersten Tests und Evaluierungen wurde jedoch festgestellt, dass die Ergebnisse nicht zufriedenstellend waren.

Daraufhin wurde ein zweiter MLP Classifier mit mehr Schichten und optimierten Hyperparametern ausprobiert, um die Leistung zu verbessern.

Auf der Suche nach weiteren Verbesserungen wurde auch ein SVM (Support Vector Machine) Classifier mit & ohne Feature Extraction getestet.

Die folgenden Modelle wurden evaluiert:

- **MLP Classifier:**

Ein mehrschichtiger Perzeptron-Klassifikator (MLP) ist ein Feedforward-Neuronales Netzwerk, das aus mehreren Schichten von Neuronen besteht: einer Eingabeschicht, einer oder mehreren versteckten Schichten und einer Ausgabeschicht. Jede Verbindung zwischen den Neuronen hat ein Gewicht, das während des Trainings angepasst wird. Der MLP verwendet Aktivierungsfunktionen (z.B. ReLU, Sigmoid), um Nichtlinearitäten einzuführen und komplexe Muster in den Daten zu lernen. Das Training erfolgt durch Backpropagation und Optimierungsalgorithmen wie Adam oder SGD, um die Gewichte zu aktualisieren und die Vorhersagegenauigkeit zu maximieren.

- **SVM ohne Feature Extraction**

Ein Support Vector Machine (SVM) ist ein klassischer, margin-basierter Klassifikator. Grob funktioniert er so: die SVM sucht eine (lineare) Trennfläche im Merkmalsraum, die den Abstand (Margin) zwischen den Klassen maximiert; nur die Trainingspunkte, die die Margin am stärksten bestimmen (Support-Vectors), beeinflussen das Modell. Für nichtlineare Probleme wird der Kernel-Trick (z.B. RBF, polynomial) genutzt, der implizit in einen höherdimensionalen Raum abbildet, sodass eine lineare Trennung dort möglich wird. Mit dem Regularisierungsparameter C lässt sich die Toleranz gegenüber Fehlklassifikationen (Soft-Margin) steuern.

4.2 Modelltraining

Das Training der Modelle erfolgte mit dem vorbereiteten Datensatz. Verschiedene Hyperparameter wurden getestet, um die Leistung zu optimieren.

- MLP Classifier:

Der Code initialisiert und trainiert einen mehrschichtigen Perzeptron-Klassifikator (MLP) mit einem versteckten Layer. Dazu werden die Bilddaten zunächst abgeflacht, standardisiert und anschließend zum Trainieren des Modells verwendet.

```
1 # Train MLP Classifier
2 mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=50,
3                     alpha=1e-4, solver='adam', verbose=True, random_state=912)
4 # Flatten images to (n_samples, n_features) because
5   MLPClassifier expects 2D input
6 X_train_flat = X_train.reshape(X_train.shape[0], -1)
7 X_test_flat = X_test.reshape(X_test.shape[0], -1)
8 # Scale features
9 scaler = StandardScaler()
10 X_train_scaled = scaler.fit_transform(X_train_flat)
11 X_test_scaled = scaler.transform(X_test_flat)
12
13 # Fit model
14 mlp.fit(X_train_scaled, y_train)
```

Listing 2: MLP Classifier

- MLP Classifier mit mehr Schichten:

Der Code definiert und trainiert einen optimierten MLP-Klassifikator mit zwei versteckten Schichten und erhöhter Modellkapazität. Durch angepasste Trainingsparameter wird eine stabilere und leistungsfähigere Klassifikation auf den skalierten Trainingsdaten erreicht.

```
1 # Optimized MLP with early stopping, higher max_iter and tuned
2   learning rate/batch size
3 mlp2 = MLPClassifier(hidden_layer_sizes=(100,100),
4                     max_iter=100, alpha=1e-4, solver='adam', verbose=True,
5                     random_state=912)
6
7 # Fit model
8 mlp2.fit(X_train_scaled, y_train)
```

Listing 3: MLP Classifier mit mehr Schichten

- SVC ohne Feature Extraction

Der Code initialisiert einen Support Vector Classifier mit RBF-Kernel und balancierten Klassengewichten. Das Modell wird anschließend mit den skalierten Trainingsdaten trainiert, um eine nichtlineare Klassifikationsgrenze zu erlernen.

```
1 from sklearn.svm import SVC
2
3 svc = SVC(
4     kernel="rbf",
5     C=10,
6     gamma="scale",
7     class_weight="balanced"
8 )
9
10 svc.fit(X_train_scaled, y_train)
```

Listing 4: SVC ohne Feature Extraction

- SVC mit PCA Feature Extraction:

Der Code extrahiert HOG-Features aus den Bildern, um relevante Merkmale für die Klassifikation zu gewinnen. Diese Features werden anschließend für das Training eines Support Vector Classifiers verwendet, um die Leistung des Modells zu verbessern.

```
1 from skimage.feature import hog
2
3 def extract_hog(images):
4     features = []
5     for img in images:
6         hog_feat = hog(
7             img,
8             orientations=9,
9             pixels_per_cell=(8, 8),
10            cells_per_block=(2, 2),
11            block_norm='L2-Hys'
12        )
13         features.append(hog_feat)
14     return np.array(features)
```

Listing 5: SVC mit PCA Feature Extraction

4.3 Code-Dokumentation

Hier sind die während der Vorverarbeitung und dem Training verwendeten Code-Snippets dokumentiert:

```
1 # Libraries importieren
2 import os
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from sklearn.datasets import fetch_openml
6 from sklearn.model_selection import train_test_split
7 from sklearn.neural_network import MLPClassifier
8 from sklearn.metrics import classification_report,
9                             confusion_matrix
10 from sklearn.preprocessing import StandardScaler
11 import kagglehub
```

Listing 6: Hauptlibraries

```
1 # Download latest version of dataset from Kaggle
2 path = kagglehub.dataset_download("muhammadrehan00
3                                     /chest-xray-dataset")
4
5 print("Path to dataset files:", path)
6
7 # Print the paths to the train and test directories
8 train_dir = os.path.join(path, "chest_xray", "train")
9 test_dir = os.path.join(path, "chest_xray", "test")
10 print("Train directory:", train_dir)
11 print("Test directory:", test_dir)
```

Listing 7: Herunterladen des Datensatzes und Pfadzuweisung

```
1 # Prepare training data
2 X_train, y_train = [], []
3
4 # Train
5 normal_X, normal_y = load_images(os.path.join(path,
6     "train/NORMAL"), "Normal")
7 pneu_X, pneu_y = load_images(os.path.join(path,
8     "train/PNEUMONIA"), "Pneumonia")
9 tub_X, tub_y = load_images(os.path.join(path,
10     "train/TUBERCULOSIS"), "Tuberculosis")
11
12 X_train.extend(normal_X + pneu_X + tub_X)
13 y_train.extend(normal_y + pneu_y + tub_y)
14
15 X_train = np.array(X_train)
16 y_train = np.array(y_train)
17
18 # Prepare testing data
19 X_test, y_test = [], []
20
21 # Test
22 normal_X, normal_y = load_images(os.path.join(path,
23     "test/NORMAL"), "Normal")
24 pneu_X, pneu_y = load_images(os.path.join(path,
25     "test/PNEUMONIA"), "Pneumonia")
26 tub_X, tub_y = load_images(os.path.join(path,
27     "test/TUBERCULOSIS"), "Tuberculosis")
28
29 X_test.extend(normal_X + pneu_X + tub_X)
30 y_test.extend(normal_y + pneu_y + tub_y)
31
32 X_test = np.array(X_test)
33 y_test = np.array(y_test)
```

Listing 8: Einteilung der Bilder in Arrays und Labels zur weiteren Verarbeitung durch das entwickelte Modell

4.4 Probleme und Herausforderungen

Die Ergebnisse sowohl bei MLP Classifier als auch bei SVM waren in unserem Fall unzureichend, mit einem F1 Score von nur etwa 0,7. Dies könnte auf verschiedene Faktoren zurückzuführen sein, darunter die Komplexität der Bilder, die begrenzte Größe des Datensatzes oder die Wahl der Modelle und Hyperparameter.

Es wäre sinnvoll, in unserem Fall ein CNN (Convolutional Neural Network) zu verwenden, da diese Modelle speziell für die Verarbeitung von Bilddaten entwickelt wurden und in der Regel bessere Ergebnisse bei der Bildklassifikation erzielen.

5 Evaluation des Modells

Die Evaluation der Modelle erfolgte anhand verschiedener Metriken, darunter Genauigkeit, Präzision, Recall und F1-Score.

Außerdem wurden für die beiden MLP Classifier ROC Curves erstellt, um die Leistung der Modelle visuell darzustellen. Hier sind die Ergebnisse der Evaluierung für die verschiedenen Modelle mit verschiedenen Metriken dargestellt:

- **MLP Classifier** mit `hidden_layer_sizes=(128,128,64)`, `max_iter=300`

Confusion Matrix:

$$\begin{bmatrix} 680 & 139 & 106 \\ 66 & 507 & 7 \\ 366 & 3 & 695 \end{bmatrix}$$

Classification Report:

Class	Precision	Recall	F1-Score	Support
Normal	0.61	0.74	0.67	925
Pneumonia	0.78	0.87	0.83	580
Tuberculosis	0.86	0.65	0.74	1064
Accuracy			0.73	2569
Macro Avg	0.75	0.75	0.75	2569
Weighted Avg	0.75	0.73	0.73	2569

- **MLP Classifier** mit `hidden_layer_sizes=(10)`, `max_iter=50`

Confusion Matrix:

$$\begin{bmatrix} 692 & 120 & 113 \\ 140 & 428 & 12 \\ 384 & 6 & 674 \end{bmatrix}$$

Classification Report:

Class	Precision	Recall	F1-Score	Support
Normal	0.57	0.75	0.65	925
Pneumonia	0.77	0.74	0.75	580
Tuberculosis	0.84	0.63	0.72	1064
Accuracy			0.70	2569
Macro Avg	0.75	0.75	0.75	2569
Weighted Avg	0.75	0.73	0.73	2569

- SVM mit kernel="rbf", C=10 ohne Feature Extraction

Confusion Matrix:

$$\begin{bmatrix} 582 & 164 & 179 \\ 109 & 466 & 5 \\ 405 & 1 & 658 \end{bmatrix}$$

Classification Report:

Class	Precision	Recall	F1-Score	Support
Normal	0.53	0.63	0.58	925
Pneumonia	0.74	0.80	0.77	580
Tuberculosis	0.78	0.62	0.69	1064
Accuracy			0.66	2569
Macro Avg	0.68	0.68	0.68	2569
Weighted Avg	0.68	0.66	0.67	2569

- SVM mit kernel="rbf", C=10 mit Feature Extraction

Confusion Matrix:

$$\begin{bmatrix} 372 & 164 & 389 \\ 111 & 469 & 0 \\ 404 & 1 & 659 \end{bmatrix}$$

Classification Report:

Class	Precision	Recall	F1-Score	Support
Normal	0.42	0.40	0.41	925
Pneumonia	0.74	0.81	0.77	580
Tuberculosis	0.63	0.62	0.62	1064
Accuracy			0.58	2569
Macro Avg	0.60	0.61	0.60	2569
Weighted Avg	0.58	0.58	0.58	2569

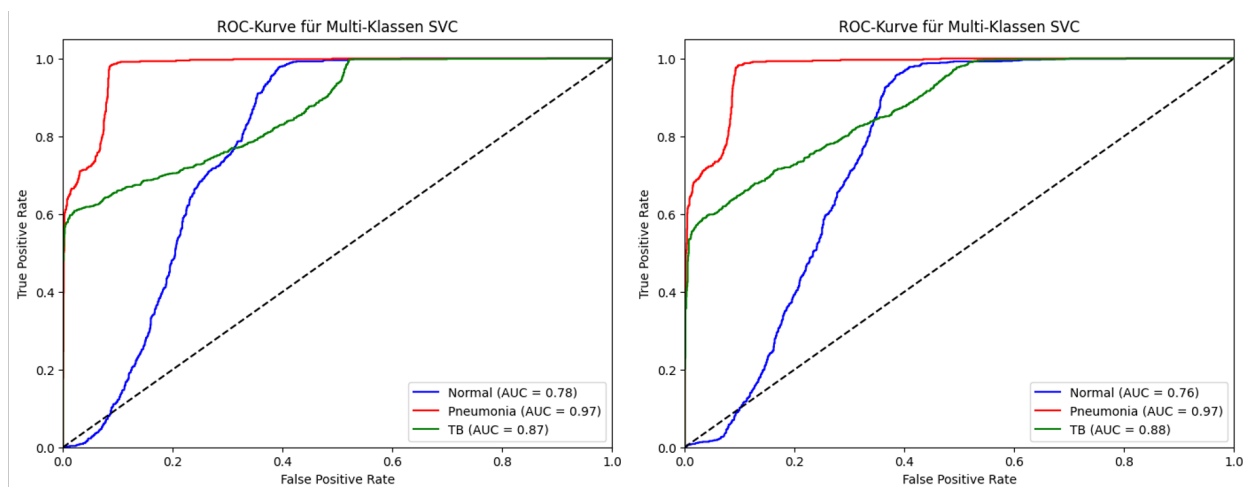


Abbildung 2: ROC Curves der beiden MLP Classifier

Die ROC Curves in Abbildung 1 zeigen die Leistung der beiden MLP Classifier. Eine ROC-Kurve (Receiver Operating Characteristic Curve) ist ein grafisches Werkzeug zur Bewertung der Leistung eines Klassifikationsmodells. Sie stellt die Beziehung zwischen der True Positive Rate (TPR) und der False Positive Rate (FPR) bei verschiedenen Schwellenwerten dar.

- **Diagonal:** Eine ROC-Kurve entlang der Diagonalen (45°-Linie) zeigt, dass das Modell keine bessere Leistung als zufälliges Raten erbringt.
- **Oberhalb der Diagonalen:** Je weiter die Kurve von der Diagonalen entfernt ist, desto besser ist die Trennleistung des Modells.
- **AUC (Area Under the Curve):** Die Fläche unter der ROC-Kurve quantifiziert die Gesamtleistung des Modells. Ein AUC-Wert von 1 bedeutet perfekte Klassifikation, während ein Wert von 0,5 auf Zufallsraten hinweist.

6 Probleme und Herausforderungen im gesamten Projekt

6.1 Technische Herausforderungen

Während des Projekts traten im Grunde keine technischen Probleme auf. Die Implementierung und das Training der Modelle verliefen größtenteils reibungslos, dank der Verwendung etablierter Bibliotheken wie Scikit-Learn und OpenCV.

Auch im Bezug auf die Hardware gab es keine nennenswerten Einschränkungen, da die Modelle mit Wartezeiten von bis zu 20 Minuten auf einem handelsüblichen Laptop trainiert werden konnten, was für uns akzeptabel war.

6.2 Modellierungsprobleme

Während des Modellierungsprozesses traten mehrere Herausforderungen auf, die die Leistung der Modelle beeinträchtigten:

- **Overfitting:** Insbesondere bei den MLP-Modellen wurde festgestellt, dass sie auf den Trainingsdaten sehr gut abschnitten, jedoch auf den Testdaten eine deutlich schlechtere Leistung zeigten. Dies deutet auf Overfitting hin.
- **Unbalancierte Daten:** Der Datensatz war nicht perfekt ausbalanciert, da die Anzahl der Bilder in den Klassen "Normal", "Pneumonia" und "Tuberculosis" unterschiedlich war.
 - Verwendung des Parameters `class_weight="balanced"` in den SVM-Modellen, um die Klassenungleichheit auszugleichen.
 - Evaluation der Modelle mit Metriken wie F1-Score, die für unbalancierte Daten besser geeignet sind.
- **Feature Extraction:** Der Versuch, HOG-Features für die SVM-Modelle zu extrahieren, führte nicht zu einer signifikanten Leistungsverbesserung sondern zu einer Verschlechterung. Möglicherweise waren die extrahierten Features nicht repräsentativ für die Klassifikationsaufgabe. Was im Bezug auf die Diagnosemerkmale und der Arbeitsweise von SVMs durchaus Sinn ergibt.

Trotz dieser Maßnahmen blieb die Leistung der Modelle begrenzt. Dies deutet darauf hin, dass ein Wechsel zu einem CNN-Modell, das speziell für Bilddaten entwickelt wurde, eine vielversprechende nächste Iteration des Projekts darstellen könnte.

7 Schlussfolgerung und Reflexion

Trotz der Implementierung verschiedener Modelle, darunter MLP Classifier und SVM, konnten die gewünschten Leistungsziele nicht erreicht werden. Die Herausforderungen, die während des Projekts auftraten sind komplexe Bildmerkmale zu erfassen, deuten darauf hin, dass eine andere Modellarchitektur erforderlich ist.

Zukünftige Arbeiten sollten sich auf die Implementierung von Convolutional Neural Networks (CNNs) konzentrieren, die speziell für die Verarbeitung von Bilddaten entwickelt wurden und in der Regel bessere Ergebnisse bei der Bildklassifikation erzielen.

Trotzdem konnte aus den Evaluationsergebnissen wertvolle Erkenntnisse gewonnen werden. So zeigte sich beispielsweise, dass die MLP Classifier im Vergleich zu dem Feature Extraction SVM-Modell wesentlich weniger oft Tuberkulose als Normal klassifizierten, im Bezug auf die Arbeitsweise von SVMs mit Feature Extraction und der Diagnosemerkmale von Röntgenbildern die sich wie folgt darstellen:

- Tuberkulose kann durch charakteristische Muster wie Kavernen und Konsolidierungen (löchrige Optik) im Lungengewebe erkannt werden.
- Pneumonie kann durch flächige Verschattungen und Infiltrate in den Lungenfeldern identifiziert werden.

kann so die Erkenntnis gewonnen werden, dass sich SVMs eher an markanten Strukturen orientieren.

Dadurch wurden die für die Tuberkulose nötigen feinen Unterschiede in den Röntgenbildern nicht erfasst und die Modelle konnten die Klassen nicht ausreichend unterscheiden.

In der ROC Curve der beiden MLP Classifier ist zu erkennen, dass beide Modelle sehr ähnliche Leistungen erbringen, obwohl das zweite Modell mit nur einer versteckten Schicht und weniger Neuronen deutlich einfacher aufgebaut ist.

Dies deutet darauf hin, dass eine weitere Erhöhung der Modellkomplexität

nicht zwangsläufig zu einer besseren Leistung führt.

Auch kann man erkennen, dass die Fläche unter der Kurve (AUC) welche die Trennschärfe zwischen fälschlicherweise richtig zugeordneten und richtig richtig zugeordneten Klassen quantifiziert bei Pneumonie am höchsten ist, was auf eine bessere Klassifikationsleistung für diese Klasse hinweist.

Insgesamt bietet dieses Projekt jedoch wertvolle Einblicke in die Herausforderungen der Bildklassifikation im medizinischen Bereich und zeigt deutlich die Gründe für die Wahl von CNNs für den Einsatz in echten Anwendungen auf.

8 Quellen und Literaturverzeichnis

- Scikit-Learn Dokumentation: <https://scikit-learn.org/stable/index.html>
- Kaggle Datensatz: <https://www.kaggle.com/datasets/muhammadrehan00/chest-xray-dataset>
- Artikel zu MLP Classifier: https://en.wikipedia.org/wiki/Multilayer_perceptron
- Artikel zu SVM: https://en.wikipedia.org/wiki/Support_vector_machine

9 Anhang