



Fakultät Informationstechnik  
PML - Projektlabor Maschinelles Lernen

---

Projekt SS 2022/2023  
Projektlabor Maschinelles Lernen

Klassifizierung von Hautkrebs

---

Studiengang Informationstechnik, Technische Informatik

Name Ali AL-Dam (2212835), Thomas Liebgott (2225000)  
Betreuer Prof. Dr. Wei Yap Tan

# Contents

<b>Einführung</b>	<b>2</b>
<b>1 Transfer-Lernen</b>	<b>3</b>
1.1 Was ist Transfer-Lernen? . . . . .	3
1.2 Warum verwendet man Transfer-Lernen? . . . . .	3
1.3 Funktionsweise des Transfer-Lernen . . . . .	4
<b>2 Vorbereitung und Verarbeitung des Datensatzes</b>	<b>5</b>
2.1 Präsentation des HAM10000-Datensatzes . . . . .	5
2.2 Vorbereitung: Extraktion und Verteilung des Datensatzes . . . . .	6
2.3 Haarentfernungsme thode . . . . .	7
2.4 Warum ist die Datensatzerweiterung wichtig? . . . . .	10
<b>3 Präsentation der Ergebnisse</b>	<b>11</b>
3.1 Evaluationsmetriken: Rolle und Auswahl der Metriken . . . . .	11
3.2 Erster Trainingsversuch . . . . .	12
3.3 Darstellung der Modellergebnisse bei varierenden Trainingsparametern . . . . .	13
3.3.1 Training mit verschiedenen Epochen . . . . .	14
3.3.2 Vergleich verschiedener ResNets bei fester Anzahl von Epochen . . . . .	17
3.3.3 Gradient optimization . . . . .	23
3.4 Vergleich verschiedener RestNets mit dem von der Haarentfernungsme thode verar- beiteten Datensatz . . . . .	25
3.5 Visuelle Untersuchung der falsch klassifizierten Hautkrebsklassen . . . . .	32
<b>4 Verbesserungsmöglichkeiten</b>	<b>37</b>
<b>5 Vergleich mit internet Ergebnis</b>	<b>39</b>
<b>Fazit</b>	<b>40</b>

# Einführung

Hautkrebs ist eine weit verbreitete Krebserkrankung in Deutschland, und weltweit leiden immer mehr Menschen daran. Es gibt verschiedene Arten von Hautkrebs, darunter das Basalzellkarzinom, Plattenepithelkarzinom, Bowen-Karzinom, malignes Melanom (schwarzer Hautkrebs), Merkelzelltumoren, Dermatofibrosarcoma protuberans und Kaposi-Sarkom. Einige dieser Hautkrebsarten wie das maligne Melanom und das Kaposi-Sarkom sind selten, aber besonders gefährlich und erfordern eine frühzeitige Identifikation und Behandlung. Generell gilt für alle Hautkrebsarten: Je früher Hautkrebs diagnostiziert und behandelt wird, desto erfolgreicher und schonender ist die Behandlung. Bei einer späten Hautkrebsdiagnose verringern sich die Behandlungsmöglichkeiten. [1] In diesem Projekt geht es nicht um ein Screening-Verfahren für gesunde Haut, sondern um die Unterscheidung der verschiedenen Hautkrebsarten. Im Rahmen dieses Projekts ist das Ziel, Probleme bei Hautkrebsklassifizierung zu untersuchen und ein optimales maschinelles Lernen Model zur Klassifizierung von Hautkrebstypen zu trainieren. Dabei werden die Bilddaten aus ISIC Challenge mit 10015 Bildern von den 7 obenbenannten Hautkrebstypen ausgenutzt. Vor dem Training vom Modell werden die Bilderdaten verarbeitet und zum Training vorbereitet. Das Model-Training erfolgt in diesem Projekt durch die Methode Transfer Learning von PyTorch. Dabei werden viele Modelle mit verschiedenen Bilddaten und verschiedenen gefalteten Neuronalen Netzwerken evaluiert. Nach der Evaluation von den Modellen könnten die Modelle anhand einer der Ensemble Methoden kombiniert werden.

# Chapter 1

## Transfer-Lernen

### 1.1 Was ist Transfer-Lernen?

NN-Modelle von Anfang an zu trainieren erfordert eine enorme Menge an markierten Daten und Rechenleistung. Dabei werden NN-Modelle typischerweise mit zufälligen Zahlen initialisiert und anschließend trainiert. Durch die Methode des Transfer-Lernen spart man viel Ressourcen, indem man ein bereits ähnlich trainiertes NN-Modell als Startmodell übernimmt und es weitertrainiert. Dadurch wird die Erfahrung des trainierten NN-Modells auf das neue NN-Modell übertragen und dabei könnte es schnell zu besseren Ergebnissen führen. Das Transfer-Lernen ist eine Technik im maschinellen Lernen, bei der Wissen von einem NN-Modell auf ein anderes übertragen wird. Dabei profitiert ein neues Modell von den Erfahrungen eines trainierten NN-Modells, um eine neue Aufgabe zu bewältigen. Transfer-Lernen ist auch bei Menschen zu bemerken. Ein Mensch mit PKW-Führerschein kann das LKW-Fahren leichter lernen als ein Mensch ohne Führerschein. Transfer-Learning-Methode macht nur dann Sinn, wenn beide NN-Modelle ähnliche Aufgaben erfüllen. Man kann zum Beispiel ein Modell zur Klassifizierung von PKWs und Motorrädern wiederverwenden, um einen LKW-Klassifizierer zu trainieren. [2]

### 1.2 Warum verwendet man Transfer-Lernen?

Die Transfer-Learning-Methode führt zu Zeit- und Ressourceneinsparung, da nicht mehrere Modelle für maschinelles Lernen von Grund auf trainiert werden müssen, um ähnliche Aufgaben zu bewältigen. Darüber hinaus verbessert sie die Effizienz von NN-Modellen, die große Mengen an Ressourcen wie Bilder und natürliche Sprache benötigen. Im Diagramm in Abbildung 1.1 ist es zu sehen, dass trainierte NN-Modelle mit Transfer Learning im Allgemeinen eine bessere Leistung erzielen und schneller den Höhepunkt erreichen als NN-Modelle ohne Transfer Learning. Anhand des Leistungsschwellenwerts (Threshold performance) wird deutlich, dass NN-Modelle ohne Transfer Learning im Vergleich zu NN-Modellen mit Transfer Learning mehr Zeit, Daten, Rechenleistung benötigen. [3]

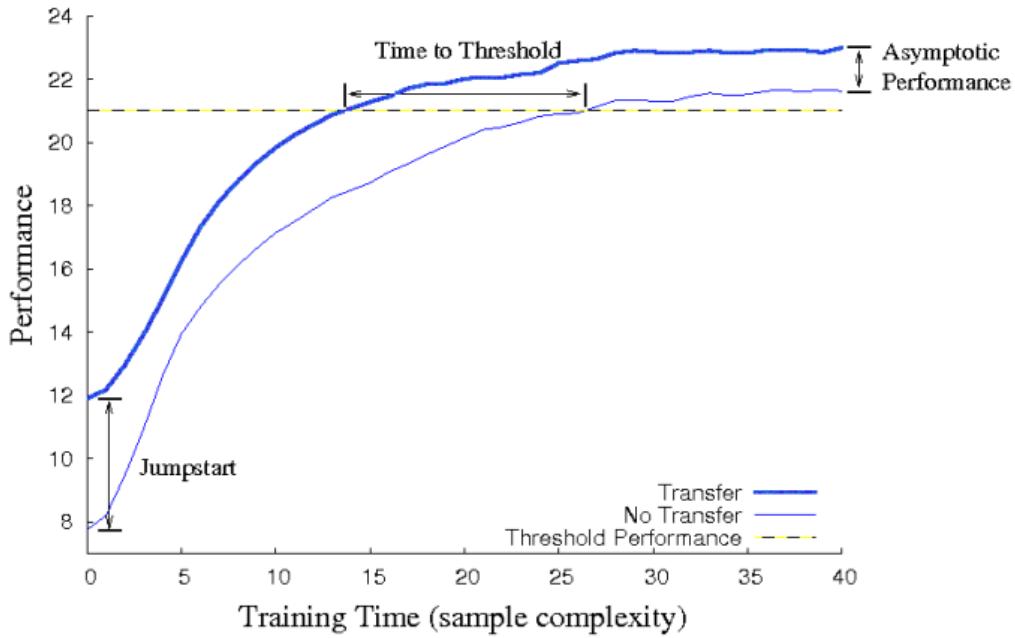


Figure 1.1: Leistungsvergleich zwischen trainierten NN-Modellen mit Transfer-Lernen und ohne Transfer-Lernen während der Trainingsphase. [4]

### 1.3 Funktionsweise des Transfer-Lernen

Beim Transfer-Learning werden relevante Teile eines zuvor trainierten Modells auf ein neues, aber ähnliche Aufgabe angewendet. Dabei handelt es sich in der Regel um die Kerninformationen, die für die Funktionalität des Modells erforderlich sind, und es werden dem Modell neue Aspekte hinzufügen, um eine bestimmte Aufgabe zu lösen. Dabei müssen Programmierer die relevanten Teile des vortrenierten Modells ins neue Modell für die gewünschte Aufgabe übertragen und entscheiden, welche Teile des Modells erneut trainiert werden müssen. [3]

## Chapter 2

# Vorbereitung und Verarbeitung des Datensatzes

### 2.1 Präsentation des HAM10000-Datensatzes

Im Rahmen des Projekts wird ein gefaltetes neuronales Netz angepasst, um Bilder von Hautkrebs mithilfe von Transferlernen zu klassifizieren. Dazu wird ein Datensatz verwendet, der bei einer internationalen Konferenz über Computer Vision im Jahr 2018 (ISIC-International Symposium on Biomedical Imaging) [5] eingesetzt wird. Dieser Wettbewerb soll die Forschung und Entwicklung von künstlicher Intelligenz zur Diagnose von Hautkrankheiten fördern. Im anschließenden Teil wird ein Vergleich der erzielten Ergebnisse mit denen dieses Wettbewerbs durchgeführt.

Die Bilder in diesem Dataset stammen aus dem HAM10000 Dataset.[6] Es gibt zwei komprimierte Ordner : HAM10000-images-part1.zip (5000 JPEG-Dateien) und HAM10000-images-part2.zip (5015 JPEG-Dateien), die aus den verschiedenen Bildern der Hautkrebsarten bestehen. Diese Bilder sind mit einer Datei HAM10000-metadata verknüpft, die den Typ der Hautverklebung eines Bildes aus sieben verschiedenen Klassen beschreibt:

- AKIEDC - Bowen-Karzinom
- BCC - Basalzellkarzinom
- BKL - Benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses)
- DF - Dermatofibrosarcoma
- MEL - Malignes Melanom
- NV - Melanocytic nevi
- VASC - Vascular lesions (Angiome, Angiokeratomas, pyogenic granulomas and hemorrhage, vasc)

## 2.2 Vorbereitung: Extraktion und Verteilung des Datensatzes

Das erste Ziel des Projekts ist es, den Datensatz zu verarbeiten, um das Training des Modells für künstliche Intelligenz zu optimieren. Wie bereits erwähnt, ist die Vorbereitung eines Datenbestands ein wichtiger Schritt, um die Leistung des Modells zu maximieren. In diesem Teil des Berichts wird die Extraktion von Daten zur Vorbereitung des Modelltrainings vorgestellt.

Im ersten Schritt werden die verschiedenen Bilder mithilfe eines in Python erstellten Skripts in Dateien eingeordnet, die ihrer Hautkrebsklasse entsprechen. Das Skript läuft durch die Textdatei HAM10000-metadata.txt, die Bildernamen und Hautkrebsklasse jedes Bildes verknüpft (datensatzt dx-OriginalVerteiltImages). Die Bilder werden dann aus der Quelldatei, in der sie gemischt sind, in den Ordner kopiert, der der Bildklasse entspricht. Es werden zwei Skripte erstellt, die die Bilder aus den Dateien HAM10000-images-part1.zip und HAM10000-images-part2 zu extrahieren, damit sie in den Trainingsphasen (train) und den Validierungsphasen (val) verwendet werden können. Außerdem wird ein weiteres Skript, extractClassTest.py, entwickelt, um die Klassen aus den Bildern im Ordner ISIC2018-Task3-Testbildern zu extrahieren, die für die Tests bestimmt sind.

Anschließend wird im Rahmen der Angleichung der Bilderanzahl eine Reihe von Bildverarbeitungsmethoden angewandt. Damit werden die Bilder vom orginalen Datensatz manipuliert und erweitert. Zu diesem Zweck wird das Skript blendClass.py implementiert. Es ermittelt die Bilderanzahl der größten Klasse aus den sieben Klassen. Anschließend werden die anderen Klassen mit den Bildverarbeitungsmethoden kontinuierlich erweitert, bis sie ungefähr gleiche Bilderanzahl wie die größte Klasse haben. Das Skript passt die Anzahl der Bildverarbeitungen an jedem Bild so an, dass die Anzahl der Bilder in jeder Klasse die Anzahl der Bilder in der größten Klasse erreicht. So kann jedes Bild mindestens einer Bildverarbeitung unterzogen werden.

Die eingesetzten Bildverarbeitungsmethoden umfassen folgende Methoden:

- Spiegeleffekt: Das Bild wird horizontal.
- Einfache Erosion: Verwendung eines elliptischen Strukturelements der Größe 1, um die Konturen des Bildes zu reduzieren.
- Einfache Dilatation: Verwendung eines elliptischen Strukturelements der Größe 1, um die Konturen des Bildes zu strecken.
- Haarentfernung: Haare im Bild werden entfernt
- Drehung des Bildes um 90°, gefolgt von einer Größenänderung des Bildes.
- Drehen des Bildes um 270°.
- Drehen des Bildes um 180°, gefolgt von einer Größenänderung des Bildes.
- Erhöhung der Helligkeit um 25%.
- Verringerung der Helligkeit um 25%.

Diese verschiedenen Bildverarbeitungen werden kombiniert, um verschiedene Datasets zu erhalten, mit denen verschiedene Trainings durchgeführt werden können. Es werden zwei verschiedene Datasets erstellt, deren Bilder von der Methoden Rotationen, Helligkeitsänderungen, und Spiegelung verarbeitet werden. Der Unterschied zwischen beiden Datasets besteht darin, dass ein Datensatz noch mit der Haarentfernungs methode erweitert wird, während der andere mit den Methoden Erosion und Dilation erweitert wird.

Um die Vorbereitung des Trainings abzuschließen, werden die verschiedenen Bilder in zwei Ordner aufgeteilt: einen für die Validierung des Modells und einen für die Training des Modells. Das Skript `separateForTrain.py` wird verwendet, um die Bilder der verschiedenen Datensätze, die erstellt werden, zu verschieben. Die Testdaten stammen aus einer separaten ZIP-Datei mit dem Namen "ISIC2018-Task3-Test-Images" und werden direkt, ohne Bildverarbeitung, in den Ordner "Test" manuell verschoben. Der "Test" Ordner befindet sich auf der gleichen Ebene wie die Ordner "train" und "val".

## 2.3 Haarentfernungs methode

Die Hautkrebsbilder aus den Datensätzen HAM10000-images-part1 und HAM10000-images-part2 sind in der Regel von Haaren bedeckt. Diese Eigenschaft ist für unser NN-Modell erwünscht und muss erlernt werden. Dabei sollte das Modell auch lernen, dass Hautkrebsbilder nicht zwangsläufig Haare enthalten müssen. Daher wird der Datensatz um Hautkrebsbilder ohne Haare erweitert. Die Entfernung von Haaren im Bild, insbesondere auf verschiedenen Hautfarben, stellt eine große Herausforderung dar, da jede individuelle Haut im Datensatz ihre eigenen Eigenschaften aufweist und jedes Haar auf der Haut eine unterschiedliche Form und Größe hat. Auch die Position der Haare spielt eine wichtige Rolle. Wenn sich zum Beispiel Haare oder Teile davon auf dem Tumor im Bild befinden, gehen Informationen über den Hautkrebs verloren. Mit der Haarentfernungs methode versucht man ähnliche Informationen wie die verlorenen zu reproduzieren. In diesem Projekt ist die Haarentfernungs methode in zwei Teile unterteilt. Im ersten Teil liegt das Ziel darin, die Positionen der Haare zu bestimmen und sie von anderen Objekten zu unterscheiden. Im zweiten Teil werden die Haarpixel erodiert und mit maximalen Wert der benachbarten Pixeln ersetzt. Für die Haarentfernungs methode wird die Kantendetektionsmethode Canny von OpenCV mit eingestellten Schwellenwerten auf die Bilder angewendet. Die Schwellenwerte werden anhand von Stichproben aus dem Datensatz justiert. Diese Schwellenwerte sind nicht optimal für alle Bilder und können zu Fehlern führen. Mit dem kantendetektierten Bild werden die Konturen der Objekte auf der Haut mithilfe der OpenCV-Funktion "findContours" ermittelt. An diesem Punkt hat man die Positionen der Konturen und ihre Größe. Nun müssen Kriterien festgelegt werden, um die Haare aus den ermittelten Konturen zu extrahieren. In diesem Projekt sind zwei Kriterien festgelegt. Erstens hat ein Haar im Vergleich zur Fläche eines Rechtecks um das Haar eine sehr kleine Fläche. Das zweite Kriterium ist, dass Haare in der Regel lang und dünn sind. Bei den Ergebnissen der Haarentfernungs methode ist zu beachten, dass die Methode kleine Haare nicht entfernt, da sie dem zweiten Kriterium widersprechen, wie beim Rechteck 2 in Abbildung 2.1 dargestellt.

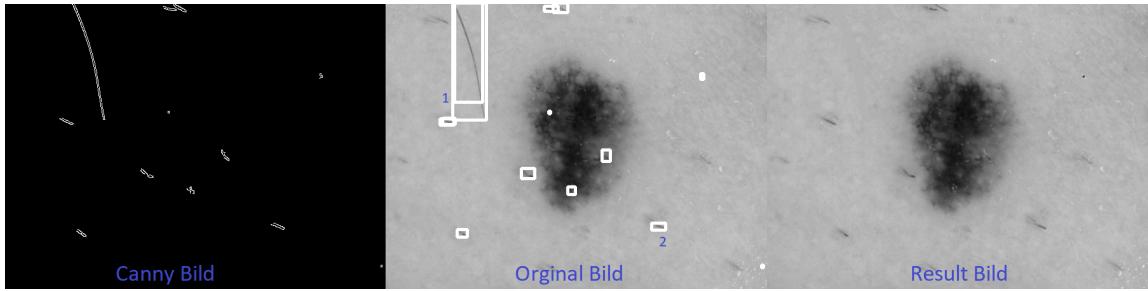


Figure 2.1: Die Haarentfernungsmethode liefert sowohl Zwischenergebnisse als auch das Endergebnis. Auf dem Canny-Bild sind links die Ergebnisse der Kantendetektion zu sehen. In der Mitte ist das Originalbild mit den gefundenen Konturen anhand des Canny-Bildes zu sehen. Auf dem rechten Bild ist das Ergebnis der Haarentfernungsmethode dargestellt.

Wie im Rechteck 1 dargestellt, ein langes und dünnes Haar im großen Rechteck ist entfernt. Die zwei Kriterien sind bei diesem Haar erfüllt. Die Methode scheint in den meisten Fällen mit den obenbenannte Kriterien zu funktionieren. Außerdem sind die Haare im Vergleich zum Hintergrund glatt entfernt. Die beiden festgelegten Kriterien funktionieren nicht für alle Haare auf dem Bild. In einigen Fällen, wie in Abbildung 2.2, werden kleine dunkle Teile der Tumorstruktur fälschlicherweise als Haare erkannt.

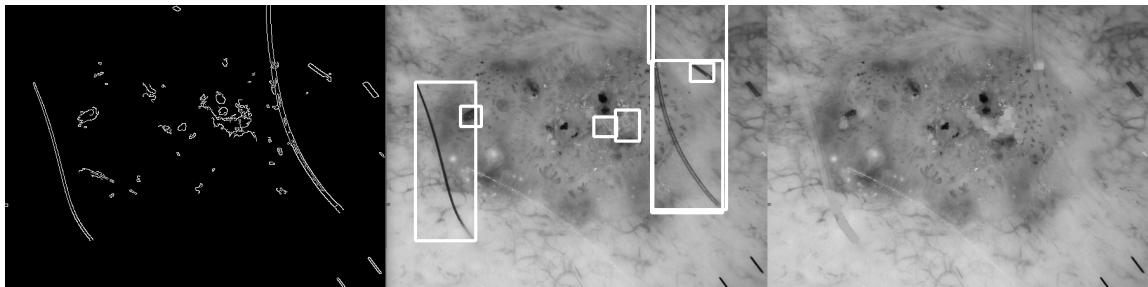


Figure 2.2: kleine Tumorstrukturen sind als Haare detektiert und verarbeitet

Dieses Problem ist nicht einfach zu lösen. Dehalb werden alle Objekte oberhalb des Tumors anhand segmentierter Bilder vom Tumor nicht berücksichtigt. Dabei werden alle Haare im segmentierten Bereich wie in Abbildung 2.3 nicht erodiert.

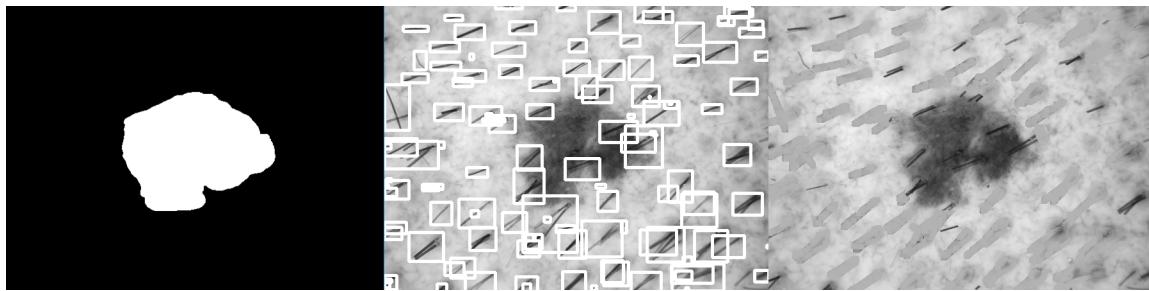


Figure 2.3: Kleine Tumorstrukturen sind als Haare detektiert aber nicht verarbeitet. Auf der linken Seite befindet sich das segmentierte Bild vom Tumor. Mittig ist das Orginalbild mit Rechtecken um die Haaren. Auf der rechten Seite sind die Haaren nur außerhalb der Tumor-Region entfernt

Man bemerkt in Abbildung 2.3, dass die Haarentfernung durch Erosion in manchen Fällen keine glatte Region im Vergleich zum Hintergrund hinterlassen hat, daher muss die Erosion Funktion noch untersucht und verbessert werden. Das erste Kriterium kann auch verletzt werden, wenn ein horizontales oder vertikales Haar, wie in Abbildung 2.4 gezeigt, vorhanden ist. In diesem Bild auf der linken Seite sieht man ein waagerechtes langes Haar, dessen Rechtecks ungefähr die glich Fläche wie die Haarfläche hat.

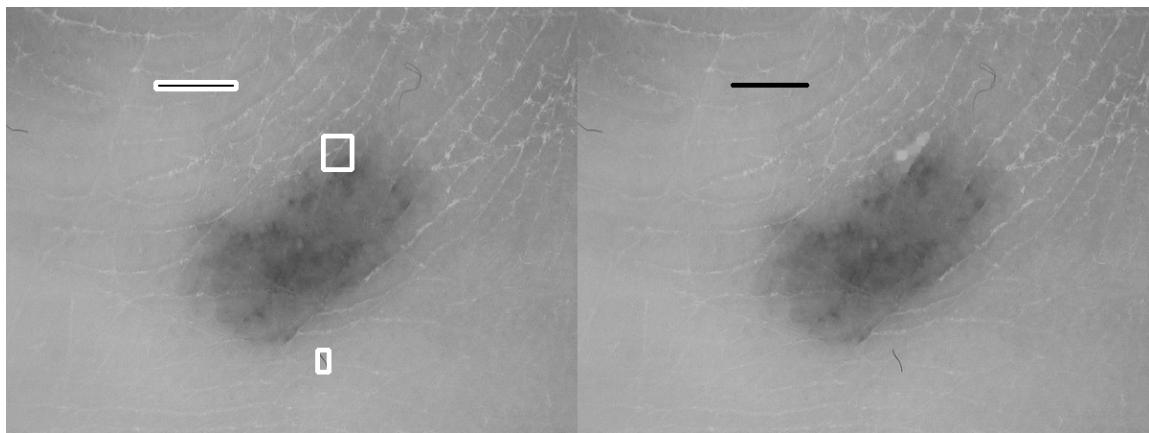


Figure 2.4: Ein synthetisch waagerecht skizziertes Haar wird nicht erodiert, da die Haarfläche im Vergleich zur Fläche des Rechtecks um das Haar ungefähr gleich ist

## **2.4 Warum ist die Datensatzerweiterung wichtig?**

Beim Training des NN-Modells ist das Ziel, dass es alle Merkmale der Hautkrebsklassen in verschiedenen realistischen Umgebungssituationen gut erlernen kann. Daher sollten die Trainings- und Validierungsdaten bestmöglich vorbereitet werden, um sicherzustellen, dass alle gewünschten Merkmale der Klassen in verschiedenen Umgebungssituationen gut erkennbar sind. Wenn zum Beispiel alle Bilder einer Klasse im Datensatz dunkel sind, sollte der Datensatz durch Hinzufügung von hellen Bildern erweitert werden, da in der Praxis Bilder in verschiedenen Beleuchtungsszenarien aufgenommen werden können.

## Chapter 3

# Präsentation der Ergebnisse

### 3.1 Evaluationsmetriken: Rolle und Auswahl der Metriken

Nach dem Training eines KI-Modells muss es anhand verschiedener Bewertungsmetriken beurteilt werden. Dies ermöglicht es uns, die Leistung unseres Modells zu bestimmen. Einer der wichtigsten Indikatoren zur Bewertung der Leistung eines Modells ist die Accuracy. Die Accuracy berücksichtigt jedoch nicht die Vorhersage über alle Stichproben hinweg. [7] Beim Trainieren des Modells kann das Phänomen des Overfitting auftreten. Overfitting kann auftreten, wenn das Modell irrelevante Informationen lernt, die man als Rauschen bezeichnen kann. Auch wenn die Trainingsdaten nicht den gesamten Varianzbereich der vollständigen Daten repräsentieren.

Um dieses Overfitting zu vermeiden, werden verschiedene Evaluations metrics berechnet und gespeichert:

- Train-losses: Diese Metrik misst die Abweichung zwischen den vorhergesagten und den tatsächlichen Werten während der Trainingsphase. Eine Verringerung dieser Metric zeigt eine Verbesserung des Modells an [8].
- Train-accs: Diese Metrik gibt den Anteil der Stichproben an, die vom Modell richtig klassifiziert werden. Eine Erhöhung dieser Metrik deutet auf eine Verbesserung des Modells hin.
- Val-losses: Diese Metric führt die gleiche Aufgabe wie Train-losses aus, und zwar auf den Validierungsdaten [8].
- Val-accs: Diese Metrik führt ebenfalls die gleiche Aufgabe wie val-accs aus, allerdings auf den Validierungsdaten.

Aus diesen Ergebnissen lässt sich eine Accuracy curve erstellen, die Aufschluss über die Relevanz des Modells gibt.

Da das gewünschte Modell mit sieben verschiedenen Klassen trainiert wird, sind die TP-Werte (True Positive values) interessant, um die korrekten Vorhersagen der verschiedenen Klassen zu identifizieren. Die anderen Werte ermöglichen es, die Klassen zu identifizieren, denen die meisten Bilder falsch zugeordnet sind.

Es ist auch interessant, zwei weitere Evaluation Metrik zu verwenden:

- Recall: Hier wird der Prozentsatz der korrekt vorhergesagten Positiven ermittelt. Dieses Maß ist für das behandelte Thema besonders interessant, da im medizinischen Bereich die Vorhersage von TP priorisiert wird. Eine hohe Recall-Rate entspricht einer niedrigen Rate an falsch negativen Ergebnissen.
- F1-Score: Diese Metric gibt einen Eindruck von der Ausgewogenheit des Modells unter den verschiedenen Klassen. Sie wird unter Berücksichtigung von FN und FP (false negative and false positive) berechnet. Je höher der F1-Score ist, desto ausgeglichener ist die Erkennung der Klassen.

Diese verschiedenen Evaluation Metric werden später verwendet, um die Leistung der erhaltenen Modelle zu beschreiben. [9] [10]

## 3.2 Erster Trainingsversuch

Wie bereits erwähnt, werden verschiedene Scripts verwendet, um die Bilder zu trennen (`separateForTrain.py`) und die Größe der Datenbank zu erhöhen (`blendClass.py`). In den folgenden Abschnitten werden die verschiedenen Trainingsprogramme beschrieben, die durchgeführt werden. Diese verschiedenen Trainingsprogramme werden mit unterschiedlichen Datensätzen und Trainingsparametern ausgeführt, die im Abschnitt Präsentation der Ergebnisse näher erläutert werden.

Um das erste Datensatz zu erstellen, auf dem das Modell trainiert wird, wird die Funktion `blendClass` als erstes verwendet, um die Datensatz aller Bilder zu erhöhen. Nachdem die Anzahl der Bilder erhöht wird, wird die erste Version des Programms `separateForTrain.py` verwendet, um die Bilder im Ordner "dx3-firstTrainReady" in einem Verhältnis von 70 % für den Training, 20 % für den Validierung und 10 % für den Test zu verteilen.

Um die Ergebnisse zu visualisieren, wird eine Funktion namens "confusion-matrix-generate-test" in den Code `transfer-learning.py` implementiert. Für das erste Training werden die Netzwerkarchitektur ResNet18 und 10 Epochen verwendet.

Nach Abschluss des Trainings ist ein Fehler festgestellt: Die Testdaten ähneln den Trainings- und Validierungsdaten. Tatsächlich stammen die Testbilder ebenfalls aus dem erweiterten Datensatz, sodass sie den Bildern ähneln, die während des Trainings und der Validierung verwendet werden. Dies führt zu einer Konfusionsmatrix mit nahezu perfekten Ergebnissen:

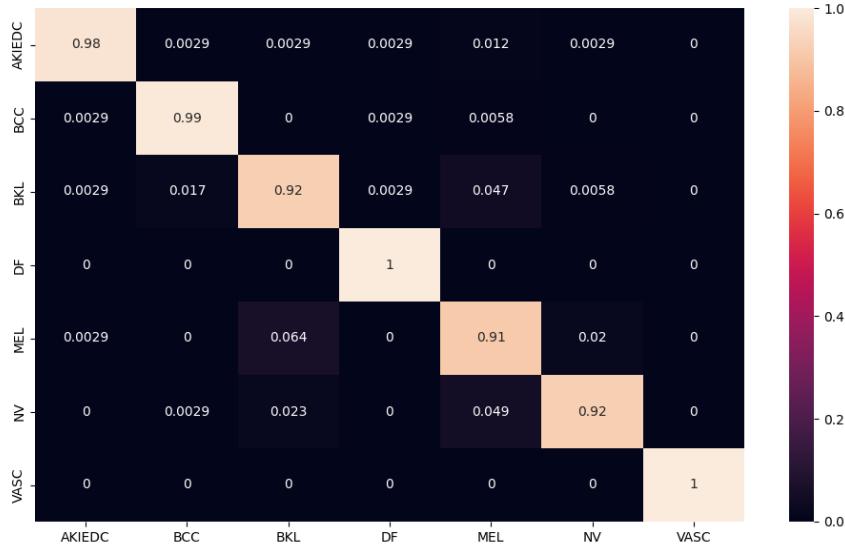


Figure 3.1: Konfusionsmatrix model ResNet18 10 Epochen dx3

### 3.3 Darstellung der Modellergebnisse bei varierenden Trainingsparametern

In diesem Abschnitt werden verschiedene Tests vorgestellt, um das Training zu optimieren. Um unsere Ergebnisse zu überprüfen, werden die Testbilder aus der Datei "ISIC2018-Task3-Test-Images" des ISIC-International Symposium on Biomedical Imaging Challenge Datensatz extrahiert. Um diese Verteilung der Bilder in die richtige Klasse zu erreichen, wird das Script `extractTestClass.py` verwendet. Unser Datensatz besteht nun aus Trainings- und Validierungsbildern, die anschließend (wiederum mit der Funktion `separateForTrain.py`) im Verhältnis 80% Training, 20% Validierung aufgeteilt werden (Datensatz dx4-ohneHaareEntfernung). Sowie Testbilder aus der Datei "ISIC2018-Task3-Test-Images".

### 3.3.1 Training mit verschiedenen Epochen

Um die verschiedenen Trainings eines Modells der künstlichen Intelligenz zu testen, wird zunächst die Anzahl der Epochen bewertet. Ein Epoch ist die Anzahl der Durchläufe, die der Algorithmus des maschinellen Lernens über den gesamten Datensatz durchführt [11]. In diesem Teil wird das Residualnetzwerk mit der geringsten Tiefe verwendet: ResNet18 [12].

ResNet18 1 epoch dx4-OhneHaareEntfernung

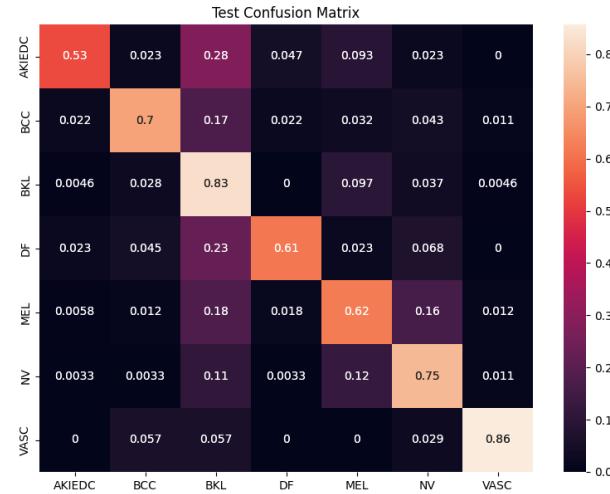


Figure 3.2: Confusion matrix ResNet18 1 epoch dx4 Ohne Haare Entfernung

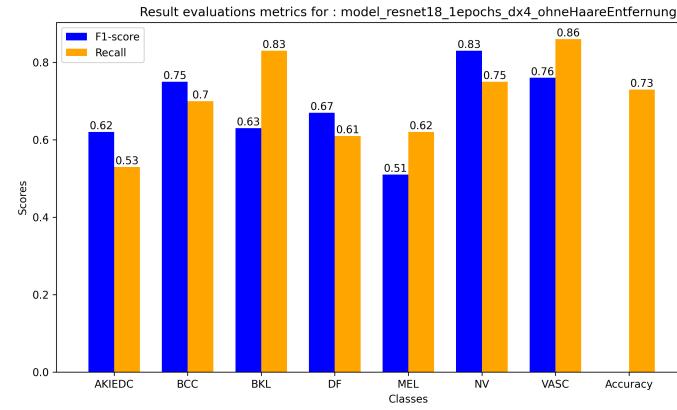


Figure 3.3: Metrics ResNet18 1 epoch dx4 Ohne Haare Entfernung

## ResNet18 10 epoch dx4-OhneHaareEntfernung

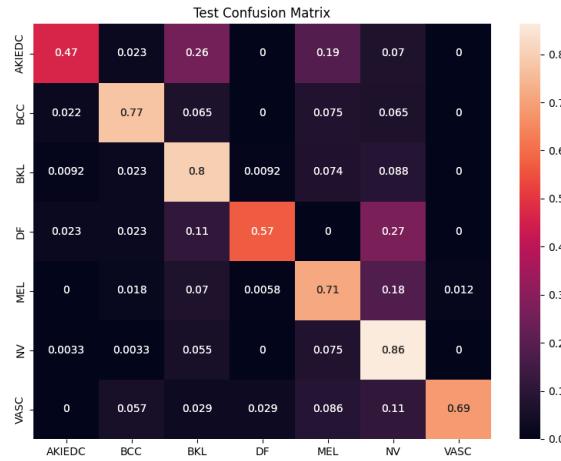
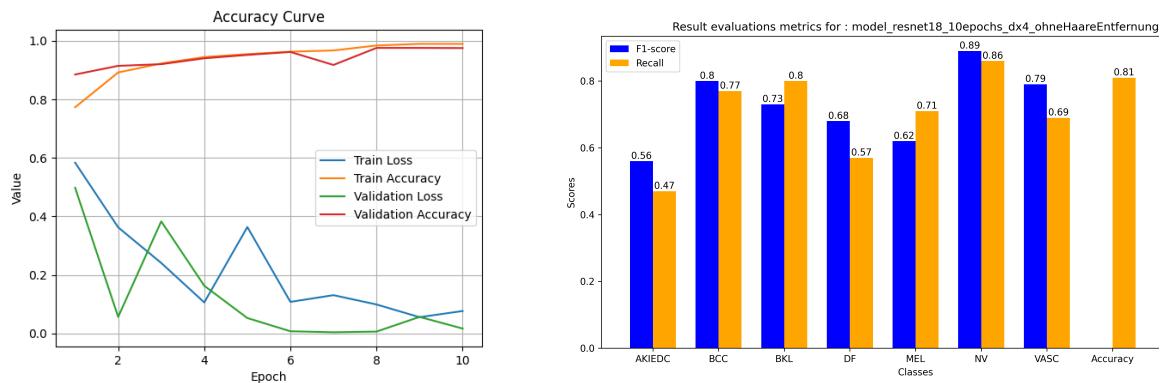


Figure 3.4: Confusion Matrix ResNet18 10 Epochen dx4-OhneHaareEntfernung



(a) Accuracy curve ResNet18 10 Epochen Ohne Haare Entfernung      (b) Metrics ResNet18 10 Epochen dx4 Ohne Haare Entfernung

Figure 3.5: ResNet18 10 Epochen dx4-OhneHaareEntfernung

## ResNet18 25 epoch dx4-OhneHaareEntfernung

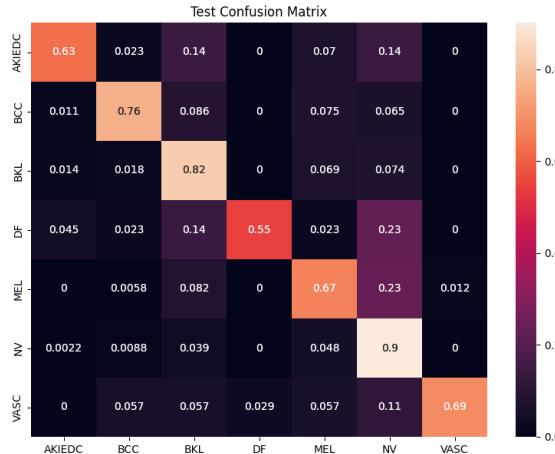
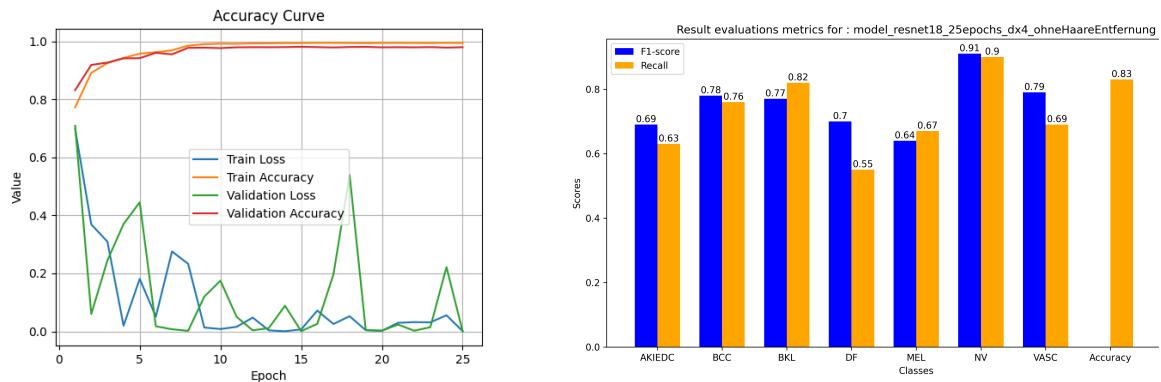


Figure 3.6: Confusion Matrix ResNet18 25 Epochen dx4-OhneHaareEntfernung



(a) Accuracy curve ResNet18 25 Epochen Ohne Haare Entfernung      (b) Metrics ResNet18 25 Epochen dx4 Ohne Haare Entfernung

Figure 3.7: ResNet18 25 Epochen dx4-OhneHaareEntfernung

Mithilfe der accuracy curves (Abbildung 3.5) kann man feststellen, ob es ein Overfitting gibt. Wenn die Kurve von train-loss sinkt, während der Wert von val-loss steigt, kann dies auf Overfitting hindeuten. Ein Rückgang der beiden Werte val-loss und train-loss ist ein Zeichen dafür, dass sich die Leistung des Modells mit zunehmender Anzahl von Epochen verbessert hat. Ab der 15. Epoch ist ein Höhepunkt dieser Werte zu erkennen, die sich nur noch wenig verändern. Auch die Werte

train-acc und val-acc weisen ein Höhepunkt auf, das zwischen der 13. und 15 (Abbildung 3.5(a)). Epoch erreicht wird. Trotz eines Spitzenwerts von val-loss in der 24. Epoche (Abbildung 3.7(a)), der auf ein Overfitting hindeuten könnte, bestätigen die accuracy curves eine Verbesserung des Modells mit zunehmender Anzahl der Epoche.

In Bezug auf die Ergebnisse der Matrix-Vergleiche ist eine Verbesserung des Recalls für die meisten Klassen zwischen 1 und 25 Epochen zu beobachten (Abbildung 3.2 - 3.4 - 3.6). Für die Klassen AKIEDC und MEL wird jedoch ein Rückgang des Recall-Wertes zwischen 10 und 25 Epochen beobachtet (Abbildung 3.5(b) und 3.7(b)). Außerdem wird für die Klasse DF eine Verschlechterung der Leistung des Modells mit zunehmender Anzahl von Epochen beobachtet (Abbildung 3.2 - 3.4 - 3.6). Die unterschiedlichen Leistungsverschlechterungen bei einigen Metriken könnten auf ein Overfitting hindeuten.

Jetzt wird die Bewertung von F1-score durchgeführt. Diese Bewertungsmetrik misst die Leistung des Modells für jede Klasse unter Berücksichtigung von FN und FP. Ein hoher Wert dieser Bewertungsmetrik weist auf ein Gleichgewicht zwischen den verschiedenen Klassen in dem Ergebnis hin. Es ist in Abbildung 3.3 - 3.5(b) - 3.7(b) zu sehen, dass sich diese Metrik für jede Klasse verbessert, mit Ausnahme der Klasse DF, bei der sie mit zunehmenden Epochen schlechter wird. Es ist jedoch zu beachten, dass die Metrik der Klasse BCC zwischen 10 und 25 Epochen um 2% abnimmt (Abbildung 3.5(b) - 3.7(b)).

Zusammenfassend lässt sich also feststellen, dass sich die verschiedenen Metriken mit steigender Anzahl an Epochen mehrheitlich verbessern. Die Kurven verlauf von val-acc und train-acc in accuracy-curves zeigt an, dass es nicht notwendig ist, die Anzahl der Epochen mehr als 25 Epochen zu erhöhen. Angesichts der begrenzten Verfügbarkeit von leistungsstarken Computerressourcen wird entschieden, mit 25 Epochen zu trainieren. Diese Entscheidung kann jedoch in Frage gestellt werden, da die accuracy curve im weiteren Verlauf des Experiments ab 15 Epochen denselben höchsten Punkt aufweist.

### 3.3.2 Vergleich verschiedener ResNets bei fester Anzahl von Epochen

Der transfer-learning Beispiel [13], der uns als Beispiel zur Verfügung gestellt ist, verwendet das ResNet-Restnetzwerk. Mit diesem Restnetz können Probleme mit dem Verschwinden des Gradienten gelöst werden. Durch die Verwendung von "Shortcuts" bietet das ResNet-Residualnetz die Möglichkeit, ein tieferes neuronales Netz zu erstellen, ohne die Leistung zu beeinträchtigen. In diesem Abschnitt werden daher mehrere Trainings durchgeführt, um die Leistung des Trainings mit der Hinzufügen zusätzlicher Schichten zu vergleichen.

## ResNet18 25 epoch dx4-OhneHaareEntfernung

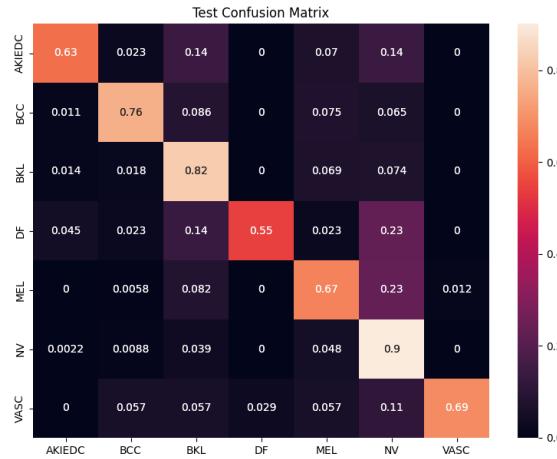
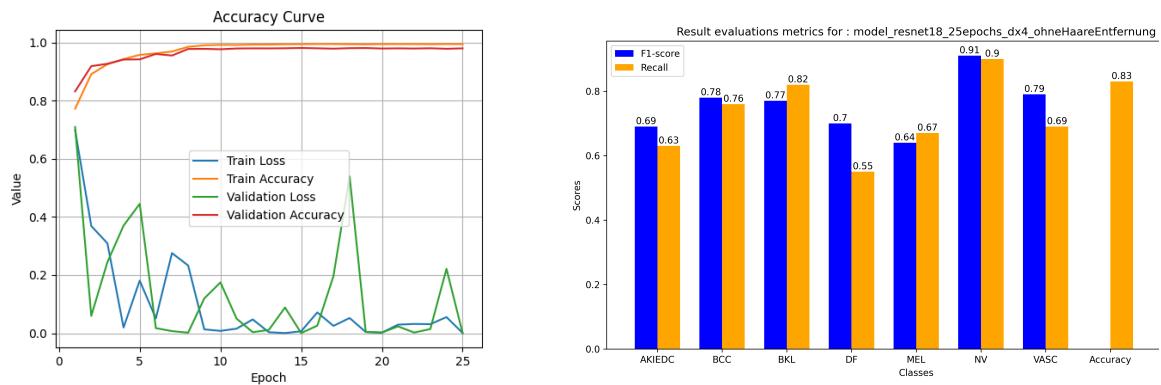


Figure 3.8: Confusion Matrix ResNet18 25 Epochen dx4-OhneHaareEntfernung



(a) Accuracy curve ResNet18 25 Epochen Ohne Haare Entfernung      (b) Metrics ResNet18 25 Epochen dx4 Ohne Haare Entfernung

Figure 3.9: ResNet18 25 Epochen dx4-OhneHaareEntfernung

## ResNet34 25 epoch dx4-OhneHaareEntfernung



Figure 3.10: Confusion Matrix ResNet34 25 Epochen dx4-OhneHaareEntfernung

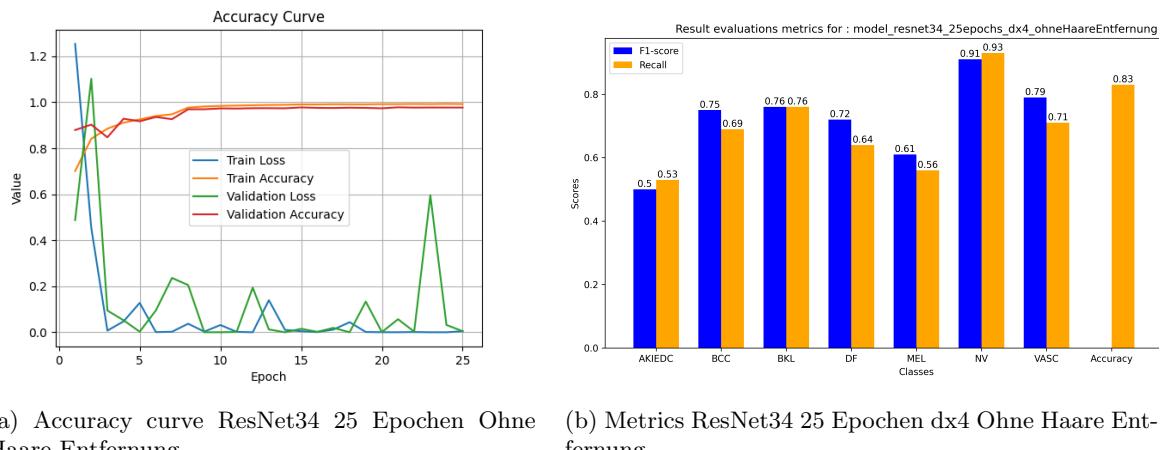


Figure 3.11: ResNet34 25 Epochen dx4-OhneHaareEntfernung

## ResNet50 25 epoch dx4-OhneHaareEntfernung

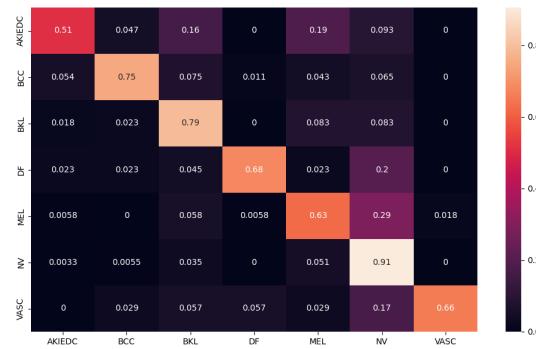


Figure 3.12: Confusion Matrix ResNet50 25 Epochen dx4-OhneHaareEntfernung

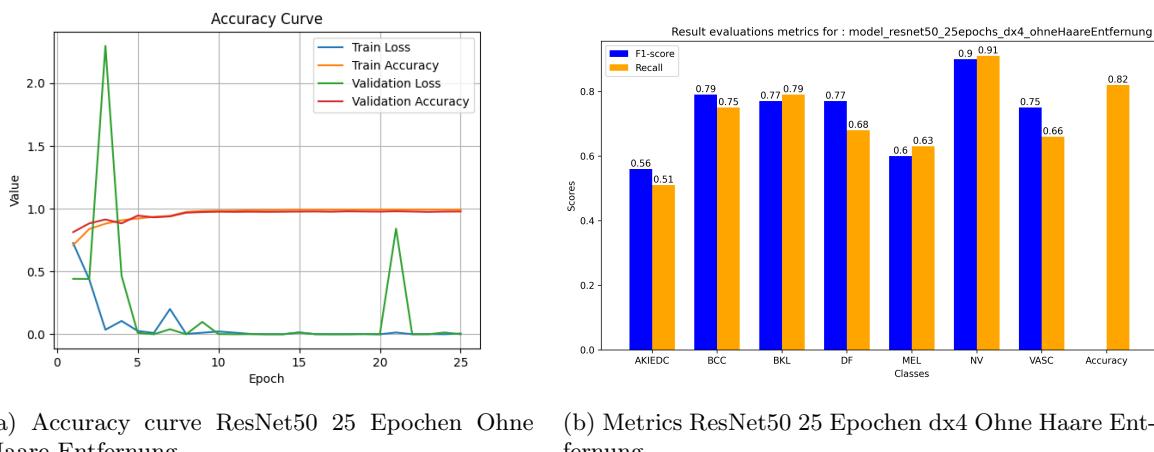
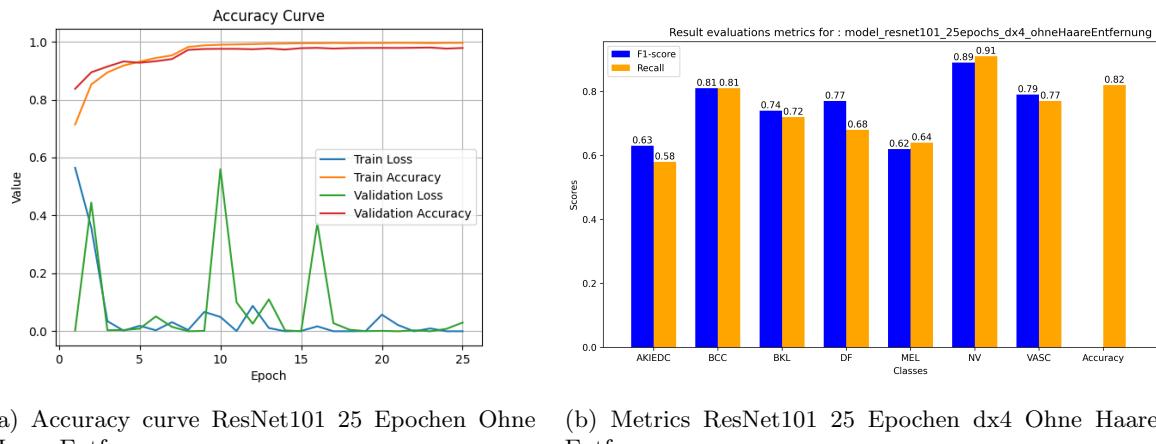


Figure 3.13: ResNet50 25 Epochen dx4-OhneHaareEntfernung

## ResNet101 25 epoch dx4-OhneHaareEntfernung



Figure 3.14: Confusion Matrix ResNet101 25 Epochen dx4-OhneHaareEntfernung



(a) Accuracy curve ResNet101 25 Epochen Ohne Haare Entfernung

(b) Metrics ResNet101 25 Epochen dx4 Ohne Haare Entfernung

Figure 3.15: ResNet101 25 Epochen dx4-OhneHaareEntfernung

Trotz einiger Ausnahmen bei einigen Epochen kann man beobachten, dass die Werte für val-loss und train-loss mit zunehmender Tiefe des Modells allmählich abnehmen (Abbildung 3.9(a) - 3.11(a) - 3.13(a) - 3.15(a)). Einige Spitzwerte, wie der am Ende des Trainings des Modells resnet34-25epoch (Abbildung 3.11(a)), können jedoch auf ein mögliches Overfitting hinweisen.

Bei den meisten Klassen ist eine Verbesserung der Recall-Vorhersage zu erkennen, mit einer maximalen Verbesserung von 13% bei der Klasse DF (Abbildung 3.9(b) - 3.11(b) - 3.13(b) - 3.15(b)). Bei einigen Klassen ist jedoch eine Verschlechterung der Leistung festzustellen, wenn die Tiefe des

Restnetzes erhöht wird (Klasse AKIEDC - ResNet18 0.63 und ResNet101 0.58 / Klasse BKL - ResNet82 0.64 und ResNet101 0.72 / Klasse MEL ResNet18 0.71 und ResNet34 0.56 und ResNet101 0.64) (Abbildung 3.9(b) - 3.11(b) - 3.13(b) - 3.15 (b)). Diese Beobachtung könnte auf das Vorliegen eines Overfittings hindeuten. Auch bei der Klasse NV, die die größte Anzahl an Bildern hat und deren Bildern nicht erhöht wird, ist eine hohe Leistung bei allen verschiedenen Tiefen des Restnetzes zu beobachten. Allerdings gibt es eine Lerngrenze ohne prozentuale Verbesserung oder sogar einen Rückgang von 1% zwischen ResNet18 und ResNet101 (Abbildung 3.8 - 3.10 - 3.12 - 3.14), was ebenfalls auf Overfitting hindeuten könnte.

Bei der Bewertung der Metriken des F1-Scores zeigten die vier Klassen BCC, DF, NV und VASC einen Anstieg der Metrik mit zunehmender Netzwerktiefe. Bei den anderen drei Klassen AKIEDC, BKL und MEL wird jedoch ein Rückgang des Wertes dieser Bewertungsmetrik beobachtet (Abbildung 3.9(b) - 3.11(b) - 3.13(b) - 3.15(b)). Obwohl dieser Rückgang gering ist, deutet er auf ein mögliches Overfitting hin.

Es gibt auch eine Variante ResNet152, mit der ein noch tieferes Netz realisiert werden kann. Aufgrund der sehr langen Trainingszeiten werden jedoch selbst bei einer Erhöhung der Batch-size (der maximale Testwert liegt bei 16) für die Modelle ResNet50 und ResNet101 Trainingszeiten von 8 Stunden bzw. 12 Stunden erreicht. Infolgedessen wird es bevorzugt, andere Methoden zur Verbesserung des Modells zu erforschen, insbesondere durch Änderung der Bildverarbeitung des Datenbestands mit der Entfernung von Haaren.

Im Rahmen der folgenden Experimente wird beschlossen, das ResNet50 beizubehalten. Insofern, als es im Vergleich zu kleineren Netzwerken eine bessere Leistung bietet und gleichzeitig eine akzeptable Trainingsdauer ermöglicht.

### 3.3.3 Gradient optimization

Bei der Klassifizierung in der Ausgabeschicht eines Convolutional Neural Networks wie ResNet werden zwei Funktionen verwendet. Die erste Funktion wird, als score-function bezeichnet, die jeder Klasse für ein bestimmtes Bild eine Zugehörigkeitsbewertung zuweist. Die zweite Funktion ist loss-function, die berechnet wird, wie nahe dem Output am Expected Output liegt. Das Ziel des CNN ist es, die Werten der loss-Function zu minimieren. Dies läuft auf eine Optimierung des Satzes von Gewichten hinaus, die die neuronalen Verbindungen des Netzes bilden. Die Optimierung eines Algorithmus für ein neuronales Netz beruht auf dem Gradientenabstieg. Dieser wird zusammen mit der Loss Function verwendet, um die Gewichte des Modells zu aktualisieren. [14]

Es gibt mehrere Optimierungsalgorithmen. In diesem Abschnitt werden die Ergebnisse von Modellen mit unterschiedlichen Optimierungsverfahren verglichen. Der zur Verfügung gestellte Transfer Learning Beispiel [13] verwendet den SGD Gradienten. Hier ist das Ergebnis eines Trainings eines resnet50-Modells mit 25 epoch unter Verwendung des SGD-Optimierers im Datensatz dx4-OhneHaareEntfernung (Abbildung 3.16):

```
# Observe that all parameters are being optimized SGD
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)
```

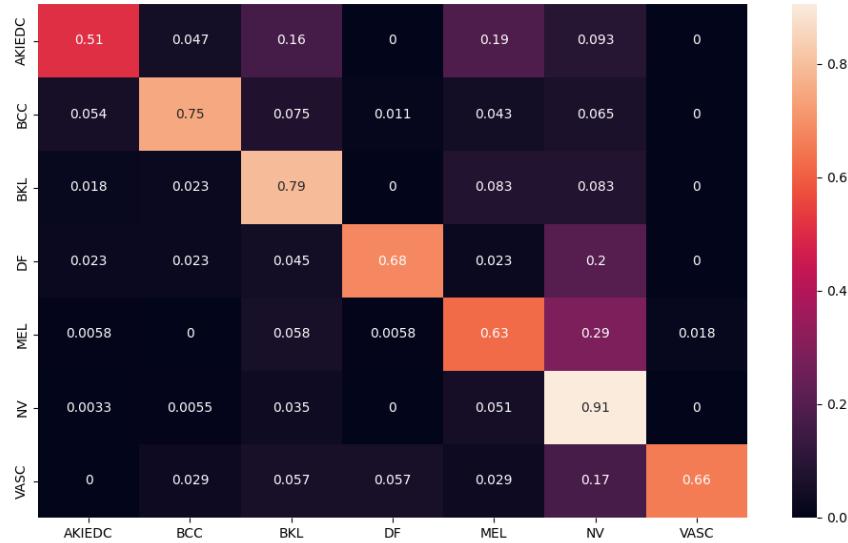


Figure 3.16: Confusion Matrix SGD Optimizer ResNet50 25 Epochen dx4-OhneHaareEntfernung

Ein weiterer weit verbreiteter Optimierer ist Adam. Er wird aufgrund seiner Popularität für den Test in der Maßnahme ausgewählt, da er wenig Speicherplatz benötigt und sich für datenintensive

Probleme eignet. Es wird ein Modell mit resnet50, 25 Epochen und der Verwendung des Adam-Optimierers auf dem Datensatz dx4-OhneHaareEntfernung trainiert. Dazu wird das Skript wie folgt geändert:

```
# Optimizer adams
optimizer_ft = Adam(model_ft.parameters(), lr=0.001)
```

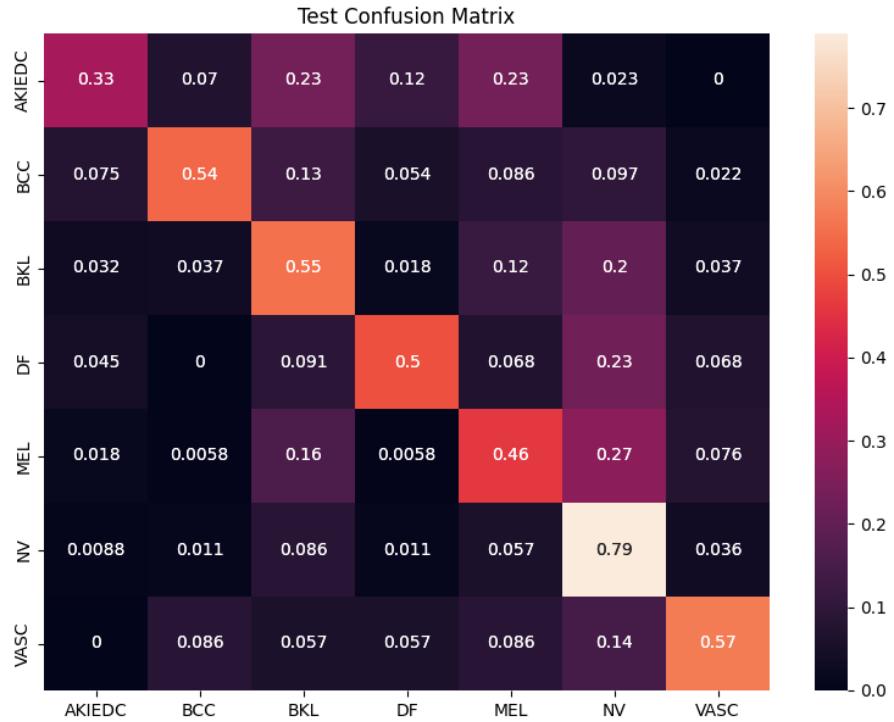


Figure 3.17: Confusion Matrix Adam Optimizer ResNet50 25 Epochen dx4-OhneHaareEntfernung

Es ist ersichtlich, dass die Ergebnisse, die mit dem Adam-Optimierer erzielt werden, schlechter sind als die Ergebnisse, die mit dem SGD-Optimierer erzielt werden (Abbildung 3.17). In der Tat zeigt jede Klasse eine geringere Genauigkeit bei der Schätzung ihres Recalls. Dies ist wahrscheinlich darauf zurückzuführen, dass der Adam-Optimierer über mehrere Hyperparameter verfügt, die in diesem Experiment nicht genutzt werden. Es wäre besser gewesen, mehrere andere Hyperparameter zu testen, um das Modell zu optimieren.[14] [15] [16] Es wird beschlossen, die Verbesserung des Datenbestands fortzusetzen. Dies umfasst die Entfernung von Haaren aus den Bildern. Zusätzlich wird Zeit für die Änderung des Optimizers verwendet, sobald die Verbesserung des Datensatz abgeschlossen ist. Aufgrund der zeitlichen Beschränkungen war es jedoch nicht möglich, weitere Verbesserungen mit diesem Optimierer durchzuführen.

### **3.4 Vergleich verschiedener RestNets mit dem von der Haarentfernungs methode verarbeiteten Datensatz**

In diesem Abschnitt werden die Lernergebnisse der NN-Modelle präsentiert, die mit dem von der Haarentfernungs methode verarbeiteten Datensatz trainiert werden. Dabei wird der Tumor-Bereich anhand segmentierter Bildern nicht von der Haarentfernungs methode verarbeitet. Hier sind die Modelle anhand von zwei Datensätzen trainiert. Beim ersten Datensatz, dx5-imagesMitHaarentfernungs, ist ein Fehler gemacht, da die Daten aus dem Datensatz dx-OriginalVerteiltImages zunächst verarbeitet und anschließend in Trainings- und Validierungsdaten aufgeteilt sind. Dadurch sehen einige Trainings- und Validierungsbilder im Datensatz dx5-imagesMitHaarentfernungs ähnlich aus, da sie aus demselben Datensatz stammen, aber unterschiedlich verarbeitet sind. Beim zweiten Datensatz, dx7-imagesMitHaarentfernungs, sind die Daten aus dem ursprünglichen Datensatz dx-OriginalVerteiltImages zuerst in Trainings- und Validierungsdaten aufgeteilt und anschließend verarbeitet. In diesem Fall sind die Trainings- und Validierungsdaten in dx7-imagesMitHaarentfernungs unterschiedlich. Die Lernergebnisse umfassen die Confusion-Matrizen und Scores des NN-Modells. Die Score-Punkte werden berechnet, indem die Summe der Zahlen auf der Hauptdiagonale der Confusion Matrix ermittelt wird. Die Confusion-Matrizen werden sowohl mit den Testdaten allein als auch mit einer Kombination aus Testdaten und Validierungsdaten erstellt. Darüber hinaus werden die NN-Modelle, die mit und ohne Haarentfernung erzeugt werden, miteinander verglichen.

Im ersten Beispiel wird das Netzwerk mit dem vorge trainierten Netz "ResNet50" initialisiert und für 15 Epochen mit den Trainings- und Validierungsdaten weiter trainiert. Dabei ist der Datensatz dx5-imagesMitHaarentfernungs benutzt. Anschließend wird die Confusion-Matrix nur anhand der Testdaten erstellt. In Abbildung 3.18 ist zu erkennen, dass die Klassifizierung häufig unabhängig von der Datensatz der einzelnen Klassen ist. Zum Beispiel wird die zweitgrößte Klasse MEL schlechter als die Klassen BKL und VASC klassifiziert. Des Weiteren wird die Klasse DF, die die kleinste Datensatz aufweist, besser als die Klasse AKIEDC identifiziert. Diese fehlerhafte Klassifizierung der Bilder in dieser Confusion-Matrix wird im nächsten Abschnitt visuell genauer untersucht.

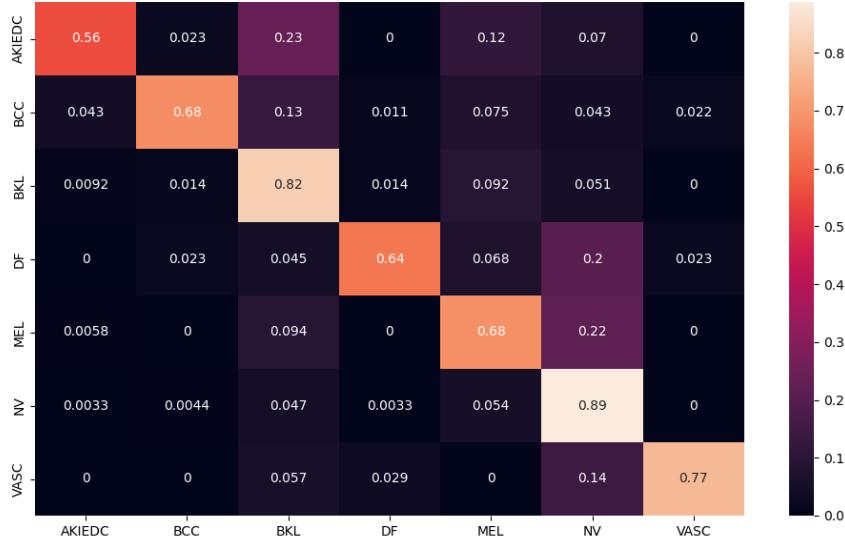


Figure 3.18: Die Confusion-Matrix repräsentiert das resnet50-Netzwerk nach 15 Epochen. Es wird ausschließlich Testdaten verwendet, um die Confusion-Matrizen zu generieren.

Im nächsten Modell handelt es sich um ein ResNet50-Netzwerk mit 25 Epochen. Dabei ist der Datensatz dx7-imagesMitHaarentfernung eingesetzt. Die Confusion-Matrix wird ausschließlich mit Testdaten erstellt. In Abbildung 3.19 ist zu erkennen, dass sich die Klassifizierung im Vergleich zur Confusion-Matrix des resnet50-Modells mit 15 Epochen nur bei den Klassen VASC und BCC verbessert hat, während sie sich bei den Klassen AKIEDC, BKL, DF und MEL verschlechtert hat.

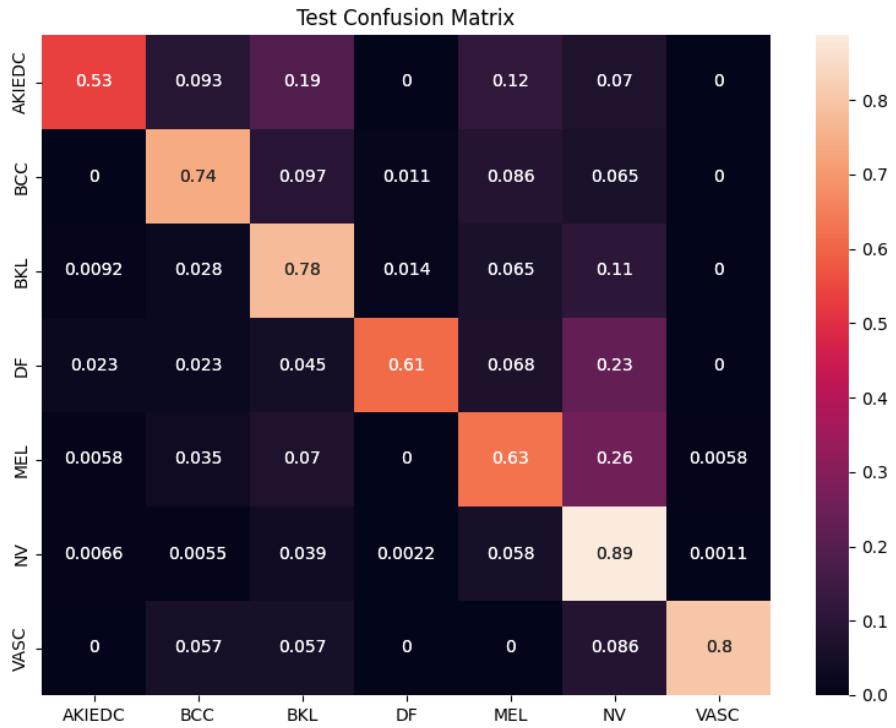


Figure 3.19: Die Confusion-Matrix stellt das Ergebnis des resnet50-Netzes nach 25 Epochen dar. Dabei werden nur die Testdaten verwendet, um die Confusion-Matrix zu erstellen. Die Trainings- und Validierungsdaten, die während der Trainingsphase verwendet werden, sind völlig unterschiedlich.

Beim nächsten Modell handelt es sich um ein kleineres Netzwerk, ResNet34, das für 25 Epochen trainiert wird. Dabei ist die Confusion-Matrix dieses Modells, wie in Abbildung 3.20 dargestellt, schlechter als die Confusion-Matrizen in den Abbildungen 3.19 und 3.18.

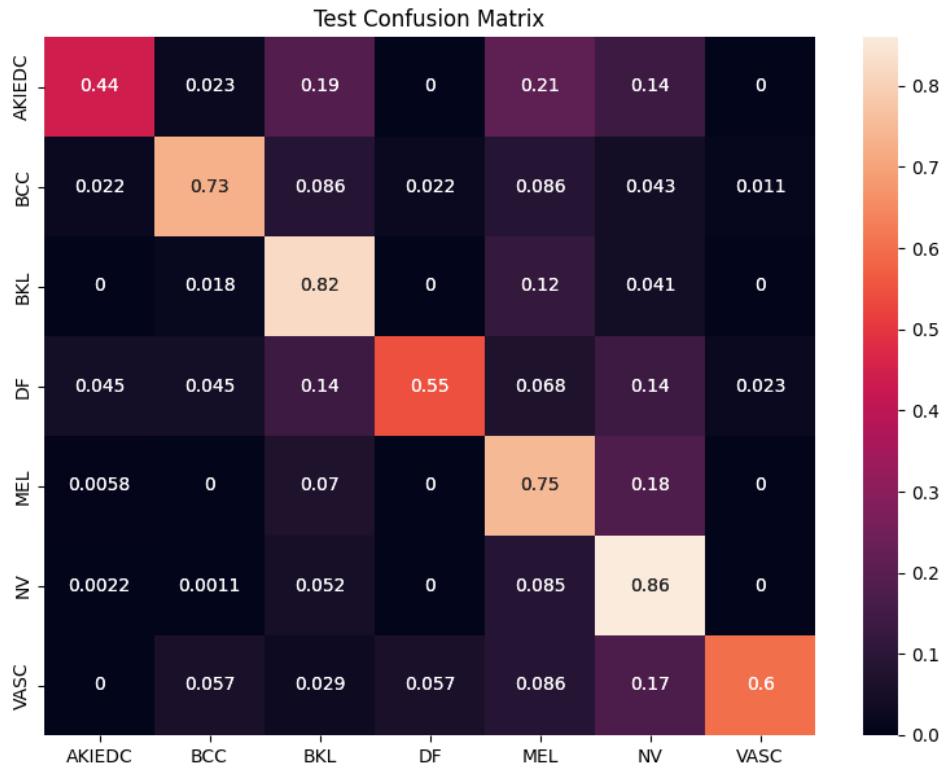


Figure 3.20: Der confusion Matrix steht für das ResNet34 Netz mit 25 Epochen. Hier sind nur die Testdaten zur Erzeugung der Confusion Matrizen eingesetzt.

Im Vergleich zu den confusion Matrizen der Modelle im letzten Abschnitt 3.3 lässt es sich im Allgemeinen wie in der Tabelle 3.4 feststellen, dass der Einsatz der Haarentfernung Methode zu bessere Klassifizierung führt.

Haarunterdrückung	Modell	Epochen	scores	scores in Prozent
nein	ResNet50	25	4,92	70,3
nein	Adam Optimizer ResNet50	25	3,74	53,4
nein	ResNet34	25	4,82	68,86
nein	ResNet18	25	5,02	71,71
nein	ResNet101	25	5,11	73
Ja	ResNet50	15	5,04	72
Ja	ResNet50	25	4,98	71,1
Ja	ResNet34	25	4,74	67,7

Table 3.4: Diese Tabelle zeigt den Vergleich zwischen die Modelle, deren Datensatz mit der Haar-

entfernungsmethode verarbeitet werden, und die Modelle, deren Datensatz nicht mit der Haarentfernungs methode verarbeitet werden.

Die Tabelle zeigt auch, dass mit zunehmender Größe des Netzwerks die Klassifizierungsergebnisse besser werden. Dabei fällt auf, dass beim Einsatz der Haarunterdrückungsmethode das Modell ResNet50 mit 15 Epochen bessere Ergebnisse liefert als das Modell mit 25 Epochen, obwohl das Modell mit 15 Epochen anhand ähnlicher Trainings- und Validierungsdaten (dx5-imagesMitHaarentfernung) trainiert wird.

In den nächsten Beispielen werden die Confusion-Matrizen der Modelle mit Haarentfernungs methode anhand der Validierungs- und Testdaten berechnet. Das erste Modell, ResNet50 mit 15 Epochen, erzielt eine sehr gute Bewertung mit über 90% für alle Klassen, wie in Abbildung 3.21 dargestellt. Der Grund für dieses gute Ergebnis liegt darin, dass die Validierungsdaten und Trainingsdaten (dx5-imagesMitHaarentfernung) in der Trainingsphase sehr ähnlich waren. Das bedeutet, dass die Validierungsdaten dem Modell bereits bekannt sind und nicht zur Berechnung der Confusion-Matrix verwendet werden sollten.

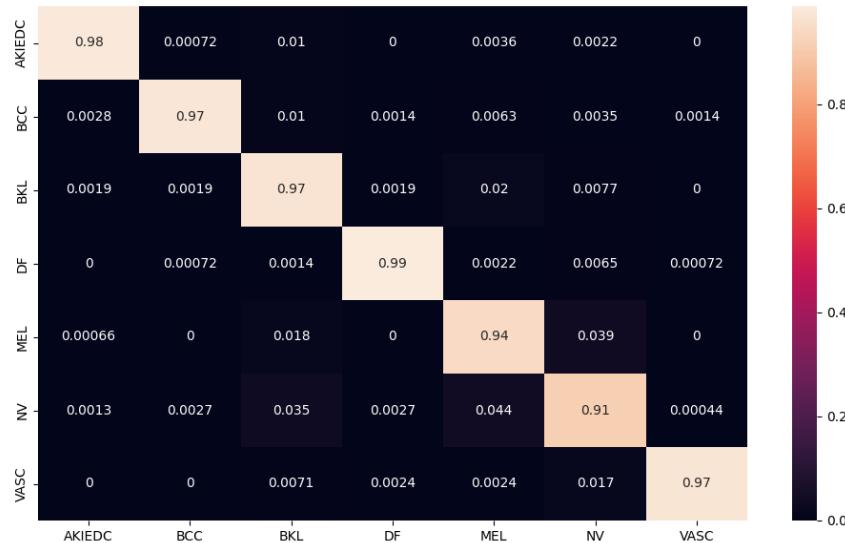


Figure 3.21: Erzeugtes confusion matrix vom Modell ResNet50 nach 15 Epochen anhand Valid- und Testdaten

Beim Modell ResNet50 mit 25 Epochen sind die Trainings- und Validierungsdaten (dx7-imagesMitHaarentfernung) in der Trainingsphase völlig unterschiedlich, daher können die Validierungsdaten zur Ermittlung der Confusion-Matrix verwendet werden. Dabei ist zu beachten, dass die Klassifizierung in der Confusion-Matrix in Abbildung 3.22 durch die Einbeziehung der Validierungsdaten im Vergleich zu Abbildung 3.19 eine deutliche Verbesserung für alle Hautkrebsklassen zeigt.

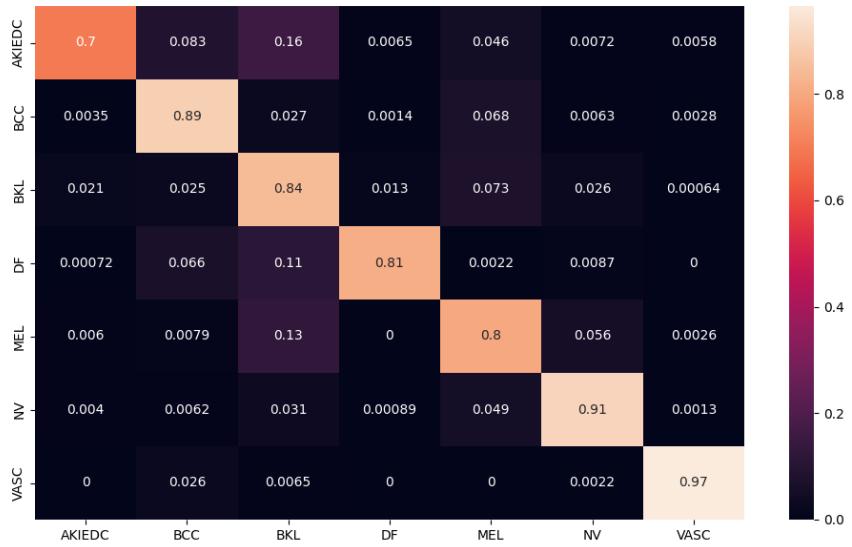


Figure 3.22: Erzeugtes confusion matrix vom Modell ResNet50 mit 25 Epochen anhand Valid- und Testdaten

Bei dem letzten Model ResNet34 mit 25 Epochen haben die Train-und Validaten in der Trainphase keinen Zusammenhang. Dabei liefert die erzeugte Confusion-Matrix in Abbildung 3.23 basierend auf Validierungs- und Testdaten bessere Ergebnisse als in Abbildung 3.20.

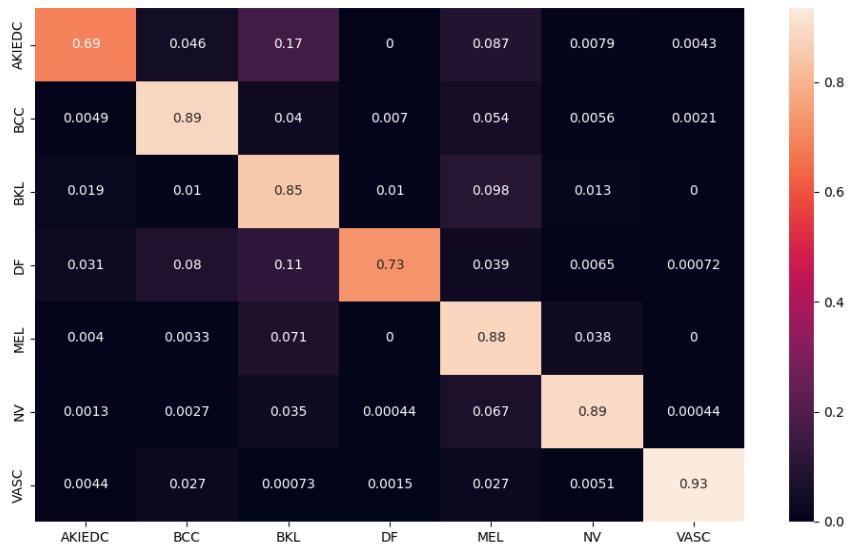


Figure 3.23: Erzeugtes confusion matrix vom Modell ResNet34 mit 25 Epochen anhand Valid- und Testdaten

Die erstellten Confusion-Matrizen basierend auf Validierungs- und Testdaten sind deutlich besser als diejenigen, die nur auf Testdaten basieren. Dieser Vergleich der Confusion-Matrizen zeigt, dass die Modelle die Validierungsdaten im Vergleich zu den Testdaten besser klassifizieren können. Es ist jedoch wichtig zu beachten, dass in der Praxis normalerweise nur die Testdaten zur Modellbewertung verwendet werden.

### **3.5 Visuelle Untersuchung der falsch klassifizierten Hautkrebsklassen**

Bei dieser Untersuchung wird die Anzahl der orginalen Datensatz jeder Klasse in Betracht genommen. In dem orginalen DatenSatz dx-OriginalVerteiltImages vor der Verarbeitung sind die Klassen nach der Größe deren Bilderanzahl wie gefolgt sortiert: NV, MEL, BKL, BCC, AKIEDC, VASC und DF. In der Wärmebild in Abbildung 3.18 sieht man das erzeugte confusion Matrix für die Testdaten. Dabei bemerkt man, dass das Modell die Klasse NV mit dem größten Bilderdaten im Datensatz besser klassifizieren konnte. Gefolgt ist sie aber mit der drittgrößten Klasse BKL an Bilderdaten. Dabei sieht man, dass die zweitgrößte Klasse MEL an Bilderdaten um 20% als die Klasse NV klassifiziert ist. Mit dem Skript "comparisonPredictedError.py" können Bilder jeder Klasse zusammen mit den Klassifizierungsergebnissen gespeichert werden. Dies ermöglicht die Untersuchung von falsch klassifizierten Bildern einer bestimmten Klasse. Nach der Untersuchung von diesen falsch klassifizierten Bildern lässt es sich feststellen, dass manche MEL Krebsbildern den NV Krebsbildern wie in Abbildung 3.24 ähneln. Es ist offensichtlich in Abbildung 3.24, dass kleine hellen Blasen innerhalb der Hautkrebs und um den herum in allen 4 Bildern sich befinden. Ein Unterschied ist in Abbildung zu sehen, dass NV Hautkrebs größer als MEL Hautkrebs ist. Diese Information kann nicht vertraut werden, da der Abstand der Kamera zum Tumor eine große Rolle spielt. Außerdem ist es nicht garantiert, dass MEL und NV Bilder mit der gleichen Kamera aufgenommen sind, da die Kamera-Auflösung und -Intrinsik einen Unterschied verursachen können.

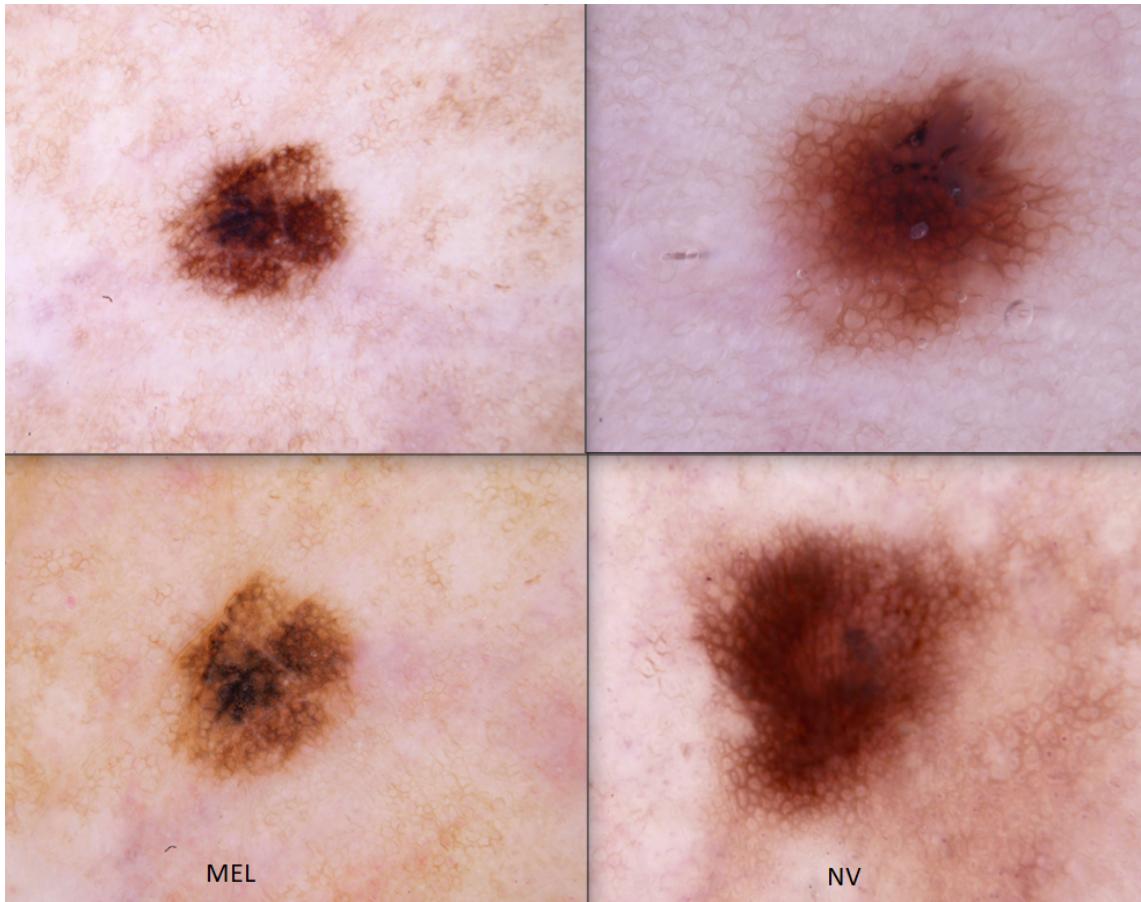


Figure 3.24: Ähnliche Krebsstrukturen zwischen der Klassen MEL und NV. Die zwei Bildern auf der linken gehören zur MEL Klasse. Die zwei rechten sind aus der NV Klasse.

Bei der Klasse BKL ist die Klassifizierung um 82% erfolgreich. Die Klasse BCC ist um 68% richtig und um 68% als die Klasse BKL klassifiziert. Bei der Untersuchungen von den beiden Klassen BCC und BKL sind keine gemeinsame Strukturen im Tumor gefunden, außer dass die meisten Bilder der beiden Klassen Wunden enthalten.

Bei der Klasse AKIEDC ist die Klassifizierung am schlechtestens. Dabei sieht man, dass die Klasse AKIEDC viel mit der Klasse BKL und MEL verwechselt wird. Nach der Untersuchung lässt es sich feststellen, dass einige Bilder in den Klassen AKIEDC und BKL ähneln. Ein Beispiel ist die Abbildung 3.25. Da sieht man ähnliche Blasen, die manchmal von Innen Dunkel und Hell von Außen und manchmal hell von Innen und Dunkel von Außen sind.

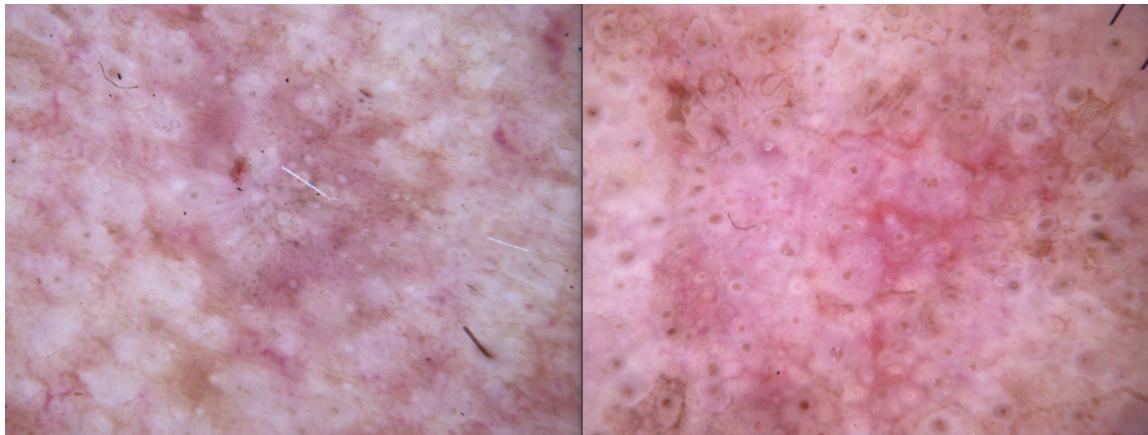


Figure 3.25: Ähnliche Krebsstrukturen zwischen der Klassen AKIEDC und BKL sind zu sehen. Das Bild auf der linken Seite gehört zur BKL Klasse, das als AKIEDC klassifiziert. Das rechte gehört zur AKIEDC Klasse.

Die Klasse VASC ist der Klasse NV bei bestimmten Bildern visuell sehr ähnlich. Vergleicht man die Bilder der Klasse VASC, die als NV Klasse zugeordnet, mit der Klasse NV, findet man ähnliche Bildern mit einem kleinen hellroten oder violetten Fleck wie in Abbildung 3.26.

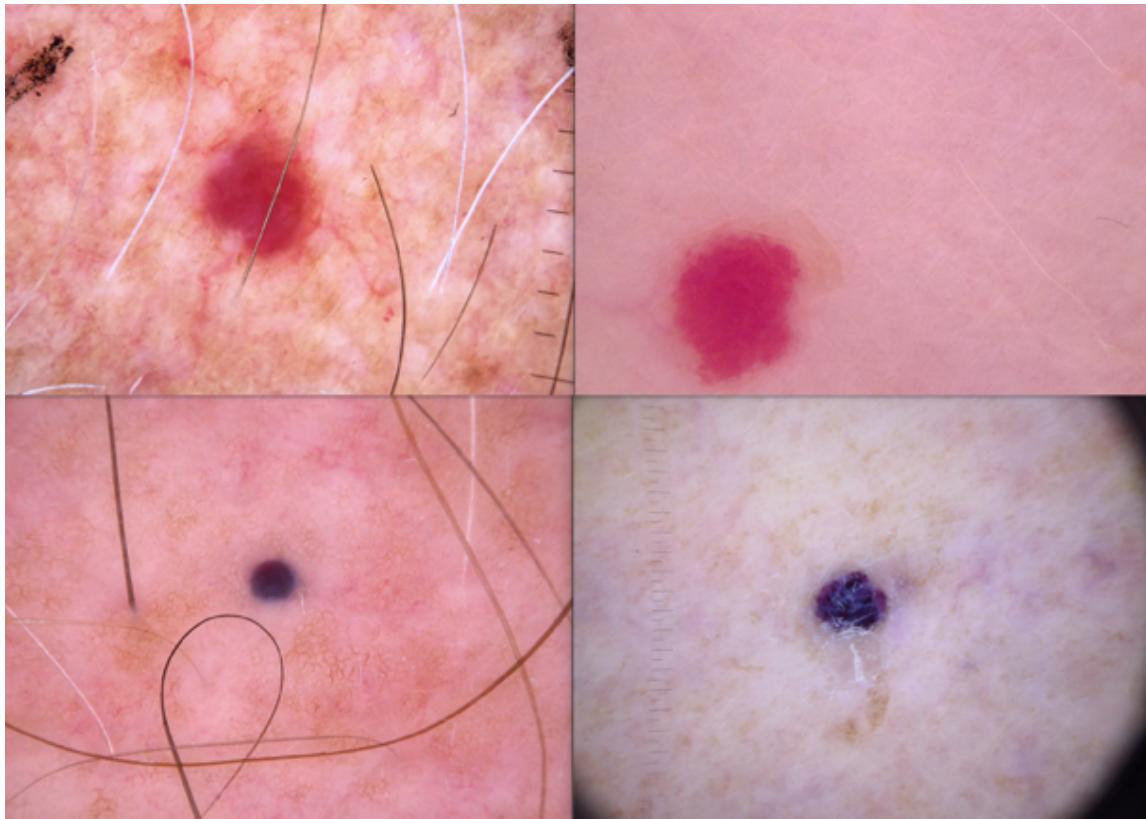


Figure 3.26: Ähnliche Bilder der Klassen MEL und NV sind zu sehen. Die zwei Bilder auf der linken gehören zur VASC Klasse, die als NV Klasse klassifiziert sind. Die zwei rechten sind aus der NV Klasse.

Die Klasse DF hat die kleinste Datensatz und schlechteste Klassifizierung in der confusion matrix. Dabei wird die Klasse DF um 20% als NV klasse verwechselt. Wenn man die Bilddaten beider Klassen vergleicht, sieht man, dass beide Bilddaten helle Region Mitte des Hautkrebs enthalten. Obwohl die Helligkeit dieser Region bei DF Klasse offensichtlich höher als bei der Klasse ist und einen deutlichen Unterschied macht, sind die DF- Bilder als NV Klasse identifiziert ist. Man trifft hier die Annahme, dass diese falsche Klassifizierung an der geringen Datensatz von der Klasse DF liegt. Ähnliche Bilder von beiden Klassen sind in Abbildung 3.27 und 3.28 zu sehen.

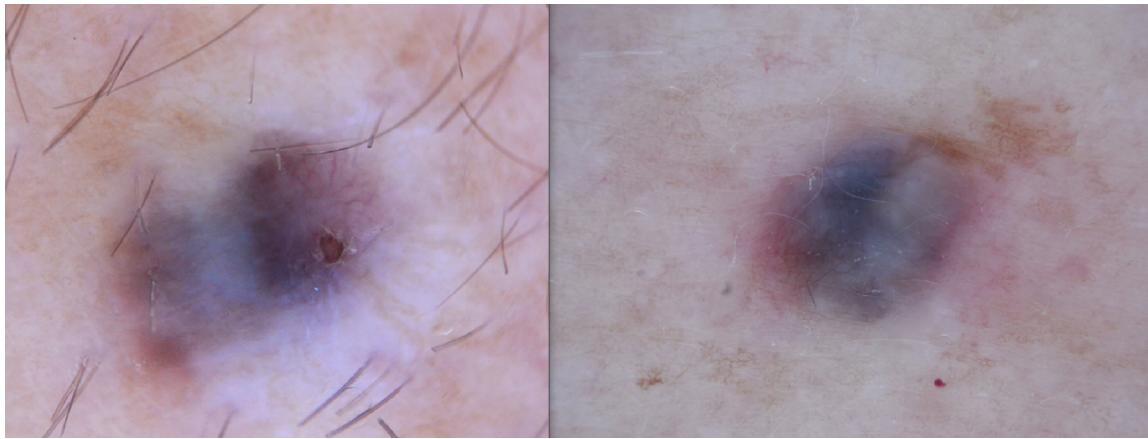


Figure 3.27: Ähnliche Bilder zwischen der Klassen DF und NV sind zu sehen. Das Bild auf der linken Seite gehört zur DF Klasse, das als NV klassifiziert. Das rechte gehört zur NV Klasse.

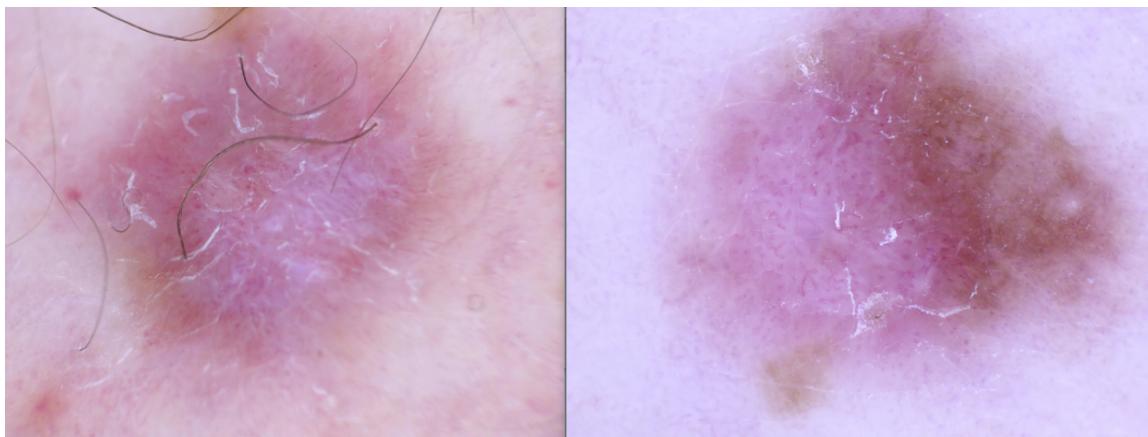


Figure 3.28: Ähnliche Bilder zwischen der Klassen DF und NV sind zu sehen. Das Bild auf der linken Seite gehört zur DF Klasse, das als NV klassifiziert. Das rechte gehört zur NV Klasse.

## Chapter 4

# Verbesserungsmöglichkeiten

Wie bereits im Abschnitt 3.3.3 über die Optimierung des Gradientenabstiegs erwähnt, gibt es mehrere Parameter des Neuronalen Netzes, die geändert werden können, um Verbesserungen des Modells zu beobachten. In diesem Abschnitt werden die verschiedenen möglichen Verbesserungen der Modellparameter vorgestellt, die aus Zeitgründen nicht implementiert werden:

### Fine Tuning

Im Laufe dieses Projekts werden die Modellen mithilfe eines Transfer-Learning-Algorithmus trainiert [13]. Dabei werden die verschiedenen Layer eingefroren. Am Ende der eingefrorenen Modellschichten werden mehrere Schichten hinzugefügt, die das Training mit den Daten des Datenset ermöglichen. Beim Fine-Tuning-Ansatz wird ein Teil oder das gesamte Modell aufgetaut und dann mit einer geringeren Trainingsrate unter Verwendung des Datenbestands erneut trainiert.

### EnsembleMethoden

Das Ziel dieser Methode ist es, mehrere Modellen mit verschiedenen Parametern und Datenbeständen zu erstellen, um sie zu kombinieren und die Vielfalt der Vorhersagen in einem einzigen Modell zu nutzen. Zu den am weitesten verbreiteten Methoden gehören : Bagging, Staking, Boosting. [17]

### Last layer

Eine weitere Einstellung, die geändert werden kann, betrifft die letzte Schicht des Modells, die bei der Ausgabe des eingefrorenen Modells hinzugefügt wird. Bei der Verwendung des Transfer Learning Codes ist die letzte Schicht eine Linear-Schicht, die wie folgt definiert ist:

```
model_ft.fc = nn.Linear(num_ftrs, 7)
```

Es wäre denkbar, diesen Teil zu modifizieren, indem man zusätzliche Schichten einführt, aber auch andere nichtlineare Aktivierungsfunktionen wie Sigmoid, ReLu oder Tangent hyperbolicus verwendet. [18]

Das vortrainierte Modell, dessen Schichten eingefroren werden, enthält bereits Schichten mit nichtlinearen Aktivierungsfunktionen. Das Hinzufügen von Schichten am Ausgang des Modells, das mit einer nichtlinearen Aktivierungsfunktion vortrainiert wird, würde es insbesondere ermöglichen, dem Training, das auf dem Dataset durchgeführt wird, Nichtlinearität hinzuzufügen. Dies würde es dem Modell ermöglichen, komplexere Darstellungen von Bildmustern zu erstellen und somit seine Fähigkeit, bestimmte Klassen genauer zu unterscheiden, zu verbessern.

## Chapter 5

# Vergleich mit internet Ergebnis

Wie bereits erwähnt, stammt das in diesem Projekt verwendete Dataset aus der ISIC 2018 Challenge. [6] Daher ist es möglich, die besten Ergebnisse, die während dieser Challenge erzielt werden, mit dem Ansatz zu vergleichen, der im Laufe dieses Projekts realisiert wird. Zu diesem Zweck werden die drei besten Ergebnisse mit den Ergebnissen verglichen.

Um ihre Ergebnisse zu bewerten, verwendeten die drei besten Teilnehmer der Challenge zunächst eine Cross-Validation [19][20][21]. Die Daten werden in k Teilmengen unterteilt und als Trainings-, Validierungs- und Testmenge verwendet. Anschließend wird die Genauigkeit des Modells geschätzt, indem der Durchschnitt über alle k-Tests berechnet wird.

Eine Vergrößerung des Datasets wird ebenfalls durchgeführt [20][21]. In den Ergebnissen kann man die Anwendung verschiedener Bildbearbeitungen beobachten, die zufällig ausgewählt werden [20]. Unter diesen 15 Bildbearbeitungen werden auch die Methoden Rotation und Brightness Modifikation verwendet.

Darüber hinaus wird bei den drei besten Ergebnissen die Ensemble-Methode verwendet. Tatsächlich werden verschiedene neuronale Netze verwendet: Für [21] EfficientNet-B6 - EfficientNet-B7, für [20] EfficientNet B3-B7, se-resnext101, resnest101 und für [22] 8 verschiedene nicht näher erläuterte Modelle.

Außerdem verwenden die Ergebnisse [20][21] Metadaten über Alter, Geschlecht und Position des Hautkrebses.

In Ergebnis [21] wird schließlich der AdamW-Optimizer verwendet, der als interessanter Ansatz genannt wird, den man nutzen könnte, wenn mehr Zeit zur Verfügung stünde.

# Fazit

Die Klassifizierung der Hautkrebsarten ist ein äußerst wichtiges Thema in der Medizin. Da einige Hautkrebsarten zeitnah erkannt und anschließend behandelt werden müssen, um eine schlechtere Prognose zu verhindern. Die Klassifizierung der Hautkrebsarten erfolgt durch hochentwickelte diagnostische Verfahren. In der Praxis werden Gewebeproben von den verdächtigen Läsionen entnommen und nach der Verarbeitung unter dem Mikroskop untersucht. Bei der Verarbeitung wird das Gewebe zentrifugiert, mit Alkohol fixiert und entweder durch Lufttrocknung oder mit verschiedenen Färbemitteln gefärbt. Dadurch können mit dem Mikroskop die Form und Farbe der einzelnen Zellen genauer betrachtet werden. [23] [24] Dadurch können mit dem Mikroskop die Form und Farbe der einzelnen Zellen genauer betrachtet werden. Durch den Einsatz von Hautkrebsklassifizierern können zeitaufwändige Maßnahmen vermieden werden, und die Ergebnisse werden in weniger als einer Sekunde angezeigt. In diesem Projekt hat das beste Modell, ResNet50, nach 25 Epochen eine Genauigkeit von %72 erreicht. Diese Quote kann durch Verbesserungen ,wie in Abschnitt erwähnt, in der Vorverarbeitung des Datensatzes und durch eine Erweiterung des originalen Datensatzes erhöht werden. In der Vorverarbeitung können Bildverarbeitungsmethoden wie die Rotation der Bilder um beliebige Winkel und die Anpassung der Kantenschärfe im Bild angewendet werden. Der vorhandene Datensatz enthält eine unterschiedliche Anzahl von Bildern für jede Hautkrebsart, was eine wichtige Rolle bei der Klassifizierung spielt. Im Abschnitt 3.5 wird deutlich, dass je ähnlicher die Hautkrebsarten sind, desto mehr Bilder von den ähnlichen Arten benötigt werden, um eine präzise Klassifizierung zu ermöglichen. Dadurch kann das Modell die feinen Unterschiede zwischen den ähnlichen Klassen erlernen. In Tabelle 3.4 ist klar, dass größere Modelle in der Lage sind, komplexe Probleme besser zu lösen. Um gute Klassifizierungsergebnisse zu erzielen, ist es wichtig, Experten aus dem Fachbereich der Hautkrebsdiagnose, insbesondere in der Bildvorverarbeitung, einzubeziehen. Zudem sollten wichtige Informationen wie das Alter und die Position des Hautkrebses beim Trainieren ausgenutzt werden. Durch die Einbeziehung dieser Informationen könnte eine verbesserte Klassifizierungsquote wie in [20][21] erzielt werden.

# Bibliography

- [1] Deutsche Krebshilfe. Hautkrebs. URL: <https://www.krebshilfe.de/informieren/ueber-krebs/krebsarten/hautkrebs/>.
- [2] Niklas Donges. What Is Transfer Learning? Exploring the Popular Deep Learning Approach. Sep 12 2022. URL: <https://builtin.com/data-science/transfer-learning>.
- [3] Dianne Castillo. Transfer Learning for Machine Learning. Jun 29 2021. URL: <https://www.seldon.io/transfer-learning>.
- [4] Peter Stone Matthew E. Taylor. “Transfer Learning for Reinforcement Learning Domains: A Survey”. In: (2009). URL: <https://www.jmlr.org/papers/volume10/taylor09a/taylor09a.pdf?ref=https://codemonkey.link>.
- [5] ISIC 2018 Challenge - Task 3: Lesion Diagnosis. 2018. URL: <https://challenge.isic-archive.com/landing/2018/47/>.
- [6] Medical University of Vienna. The HAM10000 dataset. URL: <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>.
- [7] What is overfitting? URL: <https://www.ibm.com/topics/overfitting>.
- [8] Christophe Pere. What are Loss Functions? Jun 17, 2020. URL: <https://towardsdatascience.com/what-is-loss-function-1e2605aeb904#:~:text=The%5C%20loss%5C%20function%5C%20is%5C%20the,be%5C%20categorized%5C%20into%5C%20two%5C%20groups>.
- [9] Raden Aurelius Andhika Viadinugroho. Understanding Evaluation Metrics in Classification Modeling. Apr 9, 2021. URL: <https://towardsdatascience.com/understanding-evaluation-metrics-in-classification-modeling-6cc197950f01>.
- [10] Tom Keldenich. Recall, Precision, F1 Score – Simple Metric Explanation Machine Learning. Sep 2 2021. URL: <https://inside-machinelearning.com/en/recall-precision-f1-score-simple-metric-explanation-machine-learning/>.
- [11] Sasank Chilamkurthy. What is an epoch in ML? URL: <https://deepchecks.com/glossary/epoch-in-machine-learning/>.
- [12] raghunandepu. Understanding and implementation of Residual Networks (ResNets). Oct 30 2019. URL: <https://medium.com/analytics-vidhya/understanding-and-implementation-of-residual-networks-resnets-b80f9a507b9c>.
- [13] Sasank Chilamkurthy. Transfer Learning for Computer Vision Tutorial. URL: [https://pytorch.org/tutorials/beginner/transfer\\_learning\\_tutorial.html](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html).

- [14] Sebastian Ruder. [An overview of gradient descent optimization algorithms](https://www.ruder.io/optimizing-gradient-descent/). Jan 19 2016. URL: <https://www.ruder.io/optimizing-gradient-descent/>.
- [15] Jimmy Ba Diederik P. Kingma. [Adam: A Method for Stochastic Optimization](https://arxiv.org/abs/1412.6980). 30 Jan 2017. URL: <https://arxiv.org/abs/1412.6980>.
- [16] [CS231n Convolutional Neural Networks for Visual Recognition](https://cs231n.github.io/optimization-1/?ref=ruder.io). URL: <https://cs231n.github.io/optimization-1/?ref=ruder.io>.
- [17] Evan Lutins. [Ensemble Methods in Machine Learning: What are They and Why Use Them?](https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f) Aug 2 2017. URL: <https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f>.
- [18] Vandit Jain. [Everything you need to know about “Activation Functions” in Deep learning models](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253). Dec 30 2019. URL: <https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253>.
- [19] Prashant Gupta. [Cross-Validation in Machine Learning](https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f). Jun 5 2017. URL: <https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>.
- [20] Bo. [1st place solution](https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175412). URL: <https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175412>.
- [21] Ian Pan. [\[2nd place\] Solution Overview](https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175324). URL: <https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175324>.
- [22] Masdevallia. [3rd place solution overview](https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175633). URL: <https://www.kaggle.com/competitions/siim-isic-melanoma-classification/discussion/175633>.
- [23] gesund.bund.de. [Schwarzer Hautkrebs](https://gesund.bund.de/schwarzer-hautkrebs). URL: <https://gesund.bund.de/schwarzer-hautkrebs>.
- [24] Geoffrey Rolls. [An Introduction to Specimen Preparation](https://www.leicabiosystems.com/en-fr/knowledge-pathway/an-introduction-to-specimen-preparation/). URL: <https://www.leicabiosystems.com/en-fr/knowledge-pathway/an-introduction-to-specimen-preparation/>.

# Code Documentation

- [23] *sklearn.metrics.recall-score* Verfügbar unter : [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html)
- [24] *sklearn.metrics.f1-score* Verfügbar unter : [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\_score.html#sklearn.metrics.f1\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score) [25] *panda documentation* Verfügbar unter : [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide)
- [26] *os documentation* Verfügbar unter : <https://docs.python.org/3/library/os.html>
- [27] *os methoden benutzt* Verfügbar unter : <https://www.geeksforgeeks.org/python-os-path-exists-method/>
- [28] *panda dataframe documentation* Verfügbar unter : <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>
- [29] *Confusion matrix and curve* Verfügbar unter : <https://stackoverflow.com/questions/53290306/confusion-matrix-and-test-accuracy-for-pytorch-transfer-learning-tutorial>
- [30] *matplotlib.pyplot.annotate* Verfügbar unter : [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.annotate.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.annotate.html)
- [31] *Adam optimizer* Verfügbar unter : <https://keras.io/api/optimizers/adam/>
- [32] *SGD optimizer* Verfügbar unter : <https://keras.io/api/optimizers/sgd/>
- [33] *Metrics and scoring: quantifying the quality of predictions* Verfügbar unter : [https://scikit-learn.org/stable/modules/model\\_evaluation.html](https://scikit-learn.org/stable/modules/model_evaluation.html)
- [34] *How to use Confusion Matrix in Scikit-Learn (with Python Example)*. 9 JUNE 2023 . Jean-Christophe Chouinard . Verfügbar unter :  
<https://www.jcchouinard.com/confusion-matrix-in-scikit-learn/>
- [35] *pandas.DataFrame* Verfügbar unter : <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>
- [36] *documentation seaborn.heatmap* Verfügbar unter : <https://seaborn.pydata.org/generated/seaborn.heatmap.html>
- [37] *documentation seaborn.heatmap* Verfügbar unter : <https://seaborn.pydata.org/generated/seaborn.heatmap.html>
- [38] *doc openCV* Verfügbar unter : <https://docs.opencv.org/4.7.0/>
- [39] *doc shutil* Verfügbar unter : <https://docs.python.org/3/library/shutil.html>

[40] *transfer learning - fine tuning* Verfügbar unter : [https://keras.io/guides/transfer\\_learning/](https://keras.io/guides/transfer_learning/)

[41] *gitHub link* Verfügbar unter : <https://github.com/thomasliebgott/PML>