Imbalanced Dataset - Methoden zur Verbesserung des Trainings

Konrad Schweiger *k.schweigeroth-aw.de*

21. Februar 2023

1 Einleitung

Selten sind in einem realen Datensatz für Deep Learning alle Klassen gleichverteilt. In dieser Arbeit sollen Methoden verglichen werden, um dieser Ungleichheit entgegenzuwirken. Dabei werden verschiedene Sampling-Strategien, Verlustfunktionen, Optimizer und Metriken verwendet und deren Erfolg untersucht. Das Ziel dabei ist es verschiedene Ansätze zu vergleichen und deren Einfluss auf das Training von ungleich verteilten Klassen auszuwerten.

2 Materials

2.1 Datensatz

Für die Untersuchung von Imbalanced Datasets (Datensätze mit sehr ungleich verteilten Klassenhäufigkeiten) wird der HAM10000 Datensatz [1] verwendet. Dieser enthält Bilder von verschiedenen Hautpartien, welche unterschiedliche Arten von Hautkrebs aufweisen. Der Datensatz enthält 10000 Bilder mit insgesamt sieben verschiedenen Hautkrebstypen. Da nun die Häufigkeit dieser Krebsarten nicht gleich-verteilt ist, sondern sehr starke Unterschiede aufweist, handelt es sich um einen geeigneten Datensatz für diese Versuche.

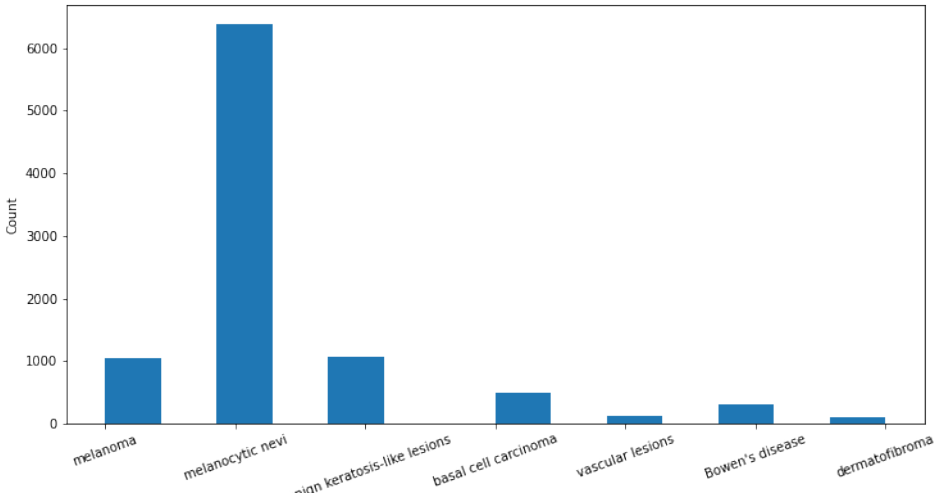

Abbildung 1: Verteilung der Klassen im HAM10000 [1] Datensatz

Aufgrund dieser Verteilung stehen Klassifikatoren vor der folgenden Hürde. Da ein Modell in knapp 50% der Fälle die Klasse `melanocytic nevi` sieht, wird er sehr schnell lernen diese Klasse deutlich häufiger vorherzusagen, als andere. Dadurch leidet die Generalisierung enorm. Wenn man nur die `Accuracy` des Modells betrachtet, ist ein solcher, schlecht angepasster Klassifikator, womöglich sogar sehr gut. Denn selbst wenn er ausschließlich diese Klasse vorhersagt, erreicht er eine Genauigkeit von mindestens 50%. Diesem Verhalten soll im folgenden entgegengewirkt werden. Ein Auszug aus den verwendeten Bildern zusammen mit deren Klassenzuordnung ist in Abbildung 2 zu sehen.

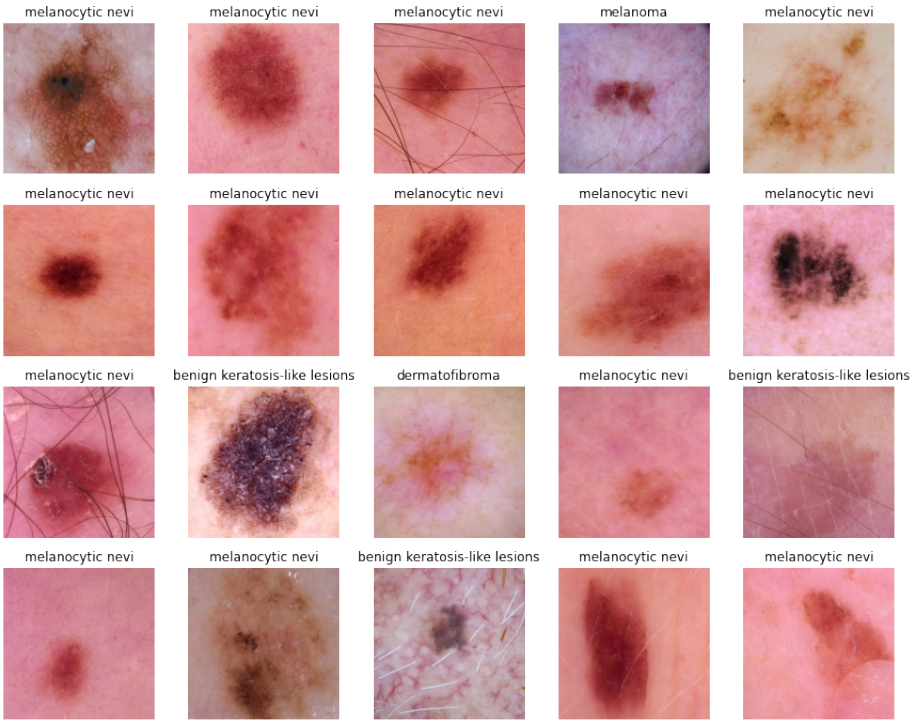

Abbildung 2: Ein Auszug aus den Bildern des Datensatzes zusammen mit den Labels

2.2 Architektur

Das neuronale Netz, welches trainiert werden soll, ist ein *resnet18*, welches auf dem Imagenet Datensatz vortrainiert wurde. Um das Netz an den neuen Datensatz anzupassen wird für 5 Epochen ein Transfer-Learning vorgeschoben. Dieses Netz wird anschließend für jeden Versuch je 40 Epochen trainiert. Für die Learningrate des Netzes wird jeweils der von *fastai* vorgestellte `Learningrate-Finder` verwendet. Dieser findet für jede Einstellung die passende Learningrate. Eine festgelegte Learningrate würde dafür sorgen, dass manche Modelle ein schlechteres Training absolvieren und damit auch die Ergebnisse verfälschen.

3 Methoden

Um die Klassenungleichheit auszugleichen werden hier verschiedene Methoden vorgestellt und erklärt.

3.1 Metriken

Zunächst muss eine geeignete Metrik gefunden werden, um bewerten zu können, wie gut der Klassifikator auch die seltenen Klassen erkennt. Denn vor allem die Accuracy bewertet jede korrekte Vorhersage des Netzes gleich. Das benachteiligt die weniger häufigen Klassen. Eine gute Metrik für unausgeglichene Datensätze sollte eine Gewichtung für die seltenen Klassen vorsehen. Die hier verwendeten Metriken werden sind: *F1-Score*, *Cohen-Kappa*, *Recall*, *Precision*, *ROC-AUC-Score*, sowie *Accuracy* als Baseline. Diese Metriken werden in allen Trainings verwendet, um die Netze zu vergleichen. Der F1-Score wurde mit einer *macro*-Gewichtung verwendet, wobei die schwächeren Klassen stärker gewichtet werden. Der F1-Score wird für jede Klasse nach folgender Formel berechnet:

$$f1\ score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{1}$$

Diese Pro-Klasse-Werte müssen noch gemittelt werden. Das geschieht nach der *macro*-Variante, indem die F1-Scores der Klassen summiert werden und durch die Anzahl der Klassen geteilt werden.
Die Precision und der Recall berechnen sich so:

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}$$

Der Cohen-Kappa Score berechnet sich folgendermaßen:

$$k = (p_0 - p_e) * (1 - p_e) \tag{4}$$

p_0: Anteil der richtigen Klassifikationen
p_e: Anteil der zufälligen Klassifikationen
Der *ROC-AUC*-Score lässt sich aus der Kurve von True Positive Rate(TPR, Recall) über False Positive Rate(FPR) ablesen.

$$FPR = \frac{FalsePositive}{TrueNegative + FalsePositive} \tag{5}$$

3.2 Sampling

Für diesen Versuch werden verschiedene Sampling-Strategien verwendet. Den Anfang macht das *undersampling*. Hierfür werden alle Klassen die häufiger als ein bestimmter Wert vorkommen seltener verwendet, es wird also eine Teilmenge aus diesen Bildern verwendet. Dies geschieht in drei Stufen. Im ersten Datensatz werden alle Klassen genau 500-mal verwendet, im zweiten 1000-mal und im dritten 3000-mal. Das geschieht, indem von häufigeren Klassen eine Teilmenge genommen wird und kleinere Klassen so oft kopiert werden, bis dieser Wert erreicht ist. Als nächstes folgt das *Oversampling*. Hierfür werden alle Bilder die nicht zur größten Klasse gehören so oft kopiert, dass sie dieselbe Anzahl, wie die maximale Klasse, erreichen. Der dritte Versuch ist die Mitte, zwischen diesen Extremen. Hierfür wird die durchschnittliche Größe der Klassen berechnet und ähnlich zu den vorherigen Methoden die Klassengrößen angepasst. Des weiteren wird ein kreativer Ansatz verwendet, um die Bilder zu vermehren. Dafür wird jedes der Bilder zerlegt in die roten, grünen und blauen Farbkanäle. Diese werden dann als Graustufenbilder abgespeichert. Dadurch erhält man Kopien des selben Bildes, die sich dennoch etwas unterscheiden. Zudem wird eine weitere Kopie des Bildes durch einen künstlichen Filter unscharf gemacht. Dieser Ansatz wird ähnlich wiederholt, indem die Zerlegung der Bilder in die Farbkanäle auf allen Bilder aller Klassen angewendet wird und im Nachhinein auf die mittlere Größe der Klasse gesampled wird.

3.3 Loss

Da die Verlustfunktion (engl.:loss function) die Klassen auch unterschiedlich gewichten kann, wird hier untersucht, wie stark dieser Einfluss ist. Getestet werden der *Cross-Entropy Loss*, *Label Smoothing Loss (flat)*, *Focal Loss (gamma=1, gamma=2)*. Dabei dient der *Cross-Entropy Loss* als Baseline. Vor allem der *Focal Loss* ist besonders für diese Aufgabe geeignet, da dieser einen größeren Wert annimmt, solange eine Klasse nicht mit sehr hoher Wahrscheinlichkeit vorhergesagt wird. Dadurch wird das Netz nicht zu selbstsicher und generalisiert besser. Der Verlauf des Cross-Entropy Loss(CE), sowie des Focal Loss samt Formel ist in der Grafik 3 zu sehen.

Der Label Smoothing Loss basiert ebenfalls auf dem Cross-Entropy-Loss. Dieser wird jedoch leicht verändert indem der Zielvektor leicht verändert wird. Dabei soll nicht länger 1 für die Zielklasse und 0 für die restlichen Klassen vorhergesagt werden. Es wird der Smoothing-Parameter ϵ eingeführt, welcher den Zielvektor verändert. Es soll nun für die korrekte Klasse $1 - \epsilon$, sowie für alle anderen Klassen ϵ vorhergesagt werden.

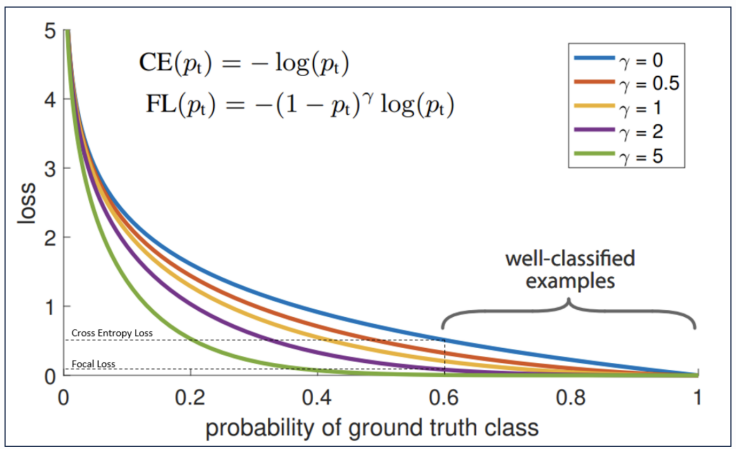

Abbildung 3: Vergleich von Cross-Entropy Loss und Focal Loss bei verschiedenen gamma Werten. Quelle:[2]

3.4 Optimizer

In diesem Training soll untersucht werden, ob die Auswahl der *Optimizer* einen Einfluss auf die Generalisierung der schlechteren Klasse hat. Dabei werden die folgenden Optimizer untersucht:

- Adam
- QHAdam
- RAdam
- SGD
- Lamb
- RMSProp
- Ranger

Die Eigenschaften dieser Optimizer können in der Dokumentation von fastai [3] entnommen werden. Besonders Interessant ist die Kombination *ranger* aus dem *Rectified Adam (RAdam)* und *LookAhead*. RAdam hat eine *warm-up*-Phase implementiert, sodass die ersten Anpassungen der Hyperparameter nicht zufällig, sondern auf bereits initialisierten Werten erfolgt. LookAhead sorgt nun dafür, dass ständig eine Kopie der Gewichte vergangener Epochen verfügbar ist und somit eine Interpolation der Gewichte mit einem alten Stand möglich ist, sollte die Anpassung der Gewichte zu keinem besseren Ergebnis führen. Dieses Vorgehen ist weiter auf dieser Website [4] beschrieben.

4 Ergebnisse und Diskussion

Die in Kapitel 3 vorgestellten Methoden wurden für 40 Epochen sowie 5 Epochen Tranfer-Learning trainiert. In diesem Abschnitt sollen nun die Ergebnisse vorgestellt werden und der Erfolg der Methoden bewertet werden.

4.1 Metriken

In diesem Training wurde untersucht, wie sich der Verlauf verschiedener Metriken unterscheidet. Um diesen Verlauf zu analysieren wurde er in der Abbildung 4 dargestellt

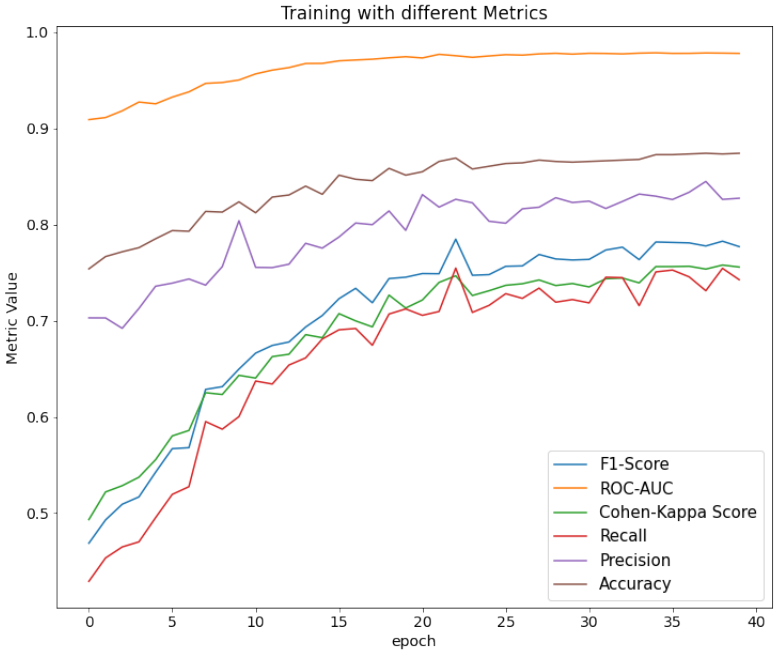

Abbildung 4: Verlauf der verschiedenen Metriken über 40 Epochen

Es ist klar zu erkennen, dass es einen großen Unterschied in der Bewertung eines Modells machen kann, welche Metrik verwendet wird. Denn einige der Metriken erreichen sehr gute Werte in diesem Training. Allerdings zeigt die Confusions-Matrix in Abbildung 5, wie viele Fehler der Klassifikator bei den kleineren Klassen macht. Denn dieser Klassifikator hat durch die ungleiche Verteilung der Klassen gelernt, immer nur die größte Klasse vorherzusagen

und dadurch eine gute Accuracy zu erreichen. Deshalb werden die drei Metriken mit den schlechtesten Werten ausgewählt die nachfolgenden Experimente zu validieren. Das sind der *F1-Score,* der *Recall* und der *Cohen-Kappa Score.* Um die Darstellung ordentlich zu halten werden im Folgenden ausschließlich die Grafiken des F1-Scores abgebildet. Es soll vor allem um den Verlauf während des Trainings gehen, welcher bei allen drei Metriken ähnlich ist.

Da dieses Training ohne spezielle Anpassungen an den unausgeglichenen Datensatz durchgeführt wurde, stellt dieser Versuch gleichzeitig die Baseline für künftige Anpassungen. Die Ergebnisse davon sind in der folgenden Tabelle zu sehen.

epoch	**train-loss**	**valid-loss**	**F1-Score**	**Recall**	**Cohen-Kappa**
40	0.518	0.366	0.777	0.874	0.827

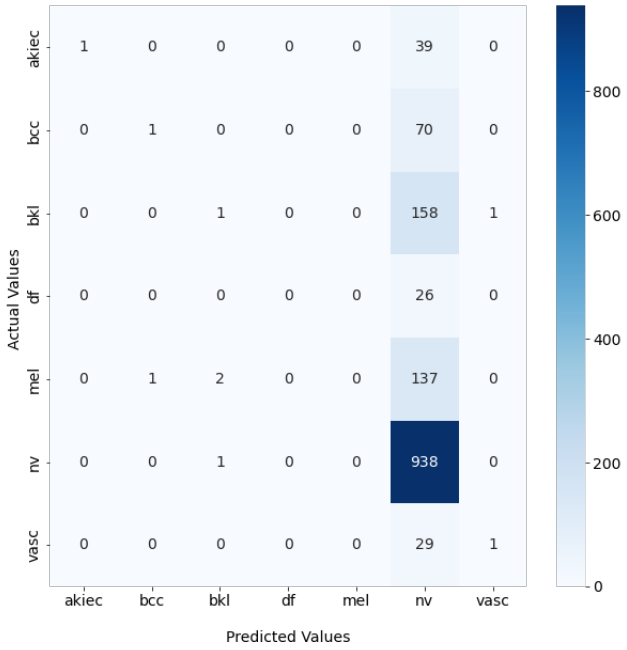

Abbildung 5: Confusionsmatrix des Baseline Modells

4.2 Sampling

Die Ergebnisse der verschiedenen Sampling Strategien sind in der Abbildung 6 zu sehen. Die Ergebnisse bestätigen die Intuition, dass es sinnvoll ist dem Klassifikator seltene Klassen öfter zu zeigen. Denn die Kurve für das `oversampled df` hat die besten Werte erziehlt. Gleichzeitig ist es auch klar warum das Training mit je 500 Bildern pro Klasse (`sampled to 500 df`)

keine guten Ergebnisse erreicht, denn die Datenmenge hat einen sehr großen Einfluss auf die Qualität des Klassifikators. Sehr interessant dabei ist es allerdings, dass die Ergebnisse sich kaum verbessern, wenn 10000 Bilder pro Klasse statt 3000 Bilder pro Klasse(*sampled to 3000*) verwendet werden. Dies könnte daran liegen, dass das Netz nicht nur sehr viele Bilder, sondern vor allem sehr viele unterschiedliche Bilder benötigt. Im allgemeinen liefern die Oversampling-Strategien in diesem Versuch jedoch die besten Ergebnisse.

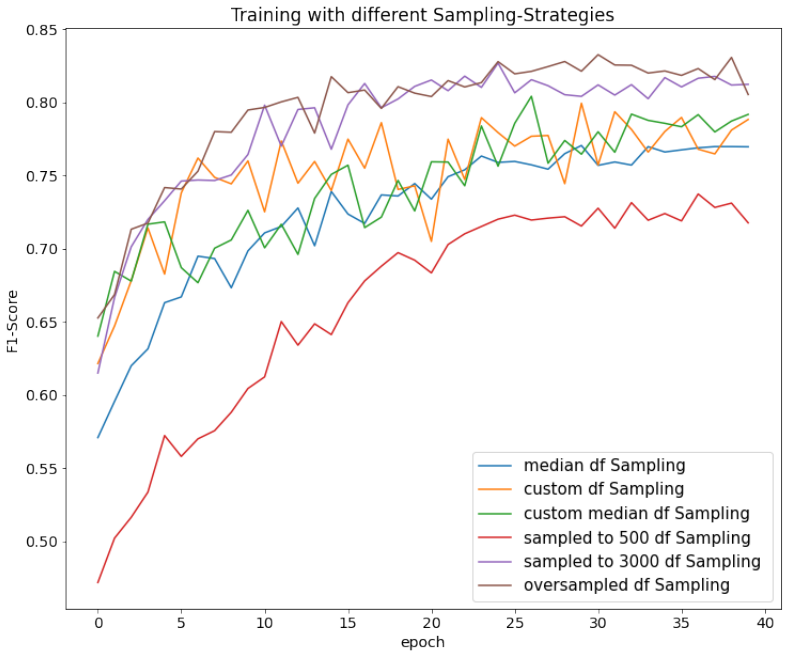

Abbildung 6: Verlauf des F1-Scores bei verschiedenen Sampling-Strategien über 40 Epochen

Die Aufteilung der Bilder in die einzelnen Farbkanäle, wie in den Kurven `custom df` und `custom median df` zu sehen, hatte weniger Erfolg (ähnlich zu *median-sampling*). Daraus könnte man schließen, dass die Farbgebung der Bilder auch Information enthält und Grauton-Bilder nicht gut geeignet sind für Hautkrebserkennung.

4.3 Loss

Der Verlauf der Loss-Funktionen im Training gibt Auskunft über die Anpassungen des Modells (der Gewichte). Bei einer Fehlklassifikation von den seltenen Klassen würde eine einfache/schlechte Loss-Funktion dennoch kleine Werte Annehmen, solange der Großteil der Labelzuordungen richtig ist. Dieses Verhalten ist in diesem Versuch auch zu erkennen. Denn einige der ausgewählten Funktionen erreichen sehr niedrige Werte, was zeigt, dass das Modell keine großen Änderungen an den Gewichten vornimmt, also zufrieden mit den Ergebnissen ist. Doch vor allem der `LabelSmoothingCrossEntropyLoss` (-Flat) zeigt dieses Verhalten nicht. Diese Verlustfunktion behält sehr große Werte. In der nachfolgenden Grafik 7 ist zu erkennen, dass sich die Loss-Werte über die Epochen sehr stark unterscheiden. Besonders hoch sind dabei die *LabelSmoothing*-Losses. Sehr niedrig sind die *FocalLosses*. Das bedeutet, dass diese Loss-Funktionen die Gewichtung nicht ausreichend auf die Klassen verteilen und daher mit der Klassifikation zufriedener sind, als die anderen Funktionen.

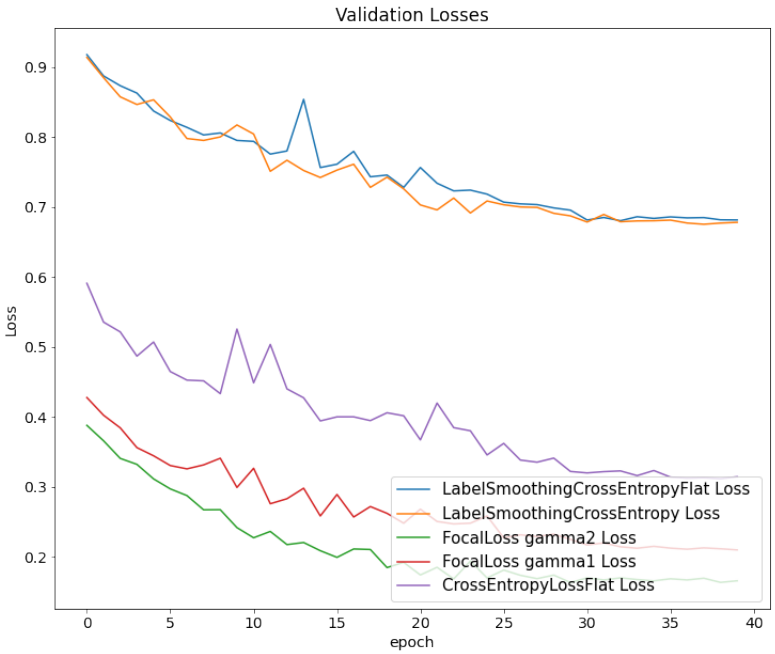

Abbildung 7: Verlauf des Validation-Losswertes bei verschiedenen Loss-Funktionen über 40 Epochen

Label Smoothing ist eine Regularisierungsmethode, welche eingesetzt werden kann, um *overconfidence* und *overfitting* zu unterdrücken. Gerade bei diesem Datensatz erlernen Modelle schnell das ausschließliche Vorhersagen der größten Klasse und das kann dieser Loss etwas unterdrücken.[5]

4.4 Optimizer

Die Ergebnisse der Verschiedenen Optimizer-Konfigurationen zeigen, wie geeignet die Auswahl für das gegebene Problem ist. In der nachfolgenden Grafik 8 ist ersichtlich, dass sich die Ergebnisse sehr stark unterscheiden. Während *Stochasik Gradient Descent* die schlechtesten Werte erreicht, hat der *ranger*-Optimizer durchgehend die besten Scores. Dieser ist dicht gefolgt von *Adam* und *RAdam*. Dies ist nicht erstaunlich, da *ranger* auf diesen aufbaut.

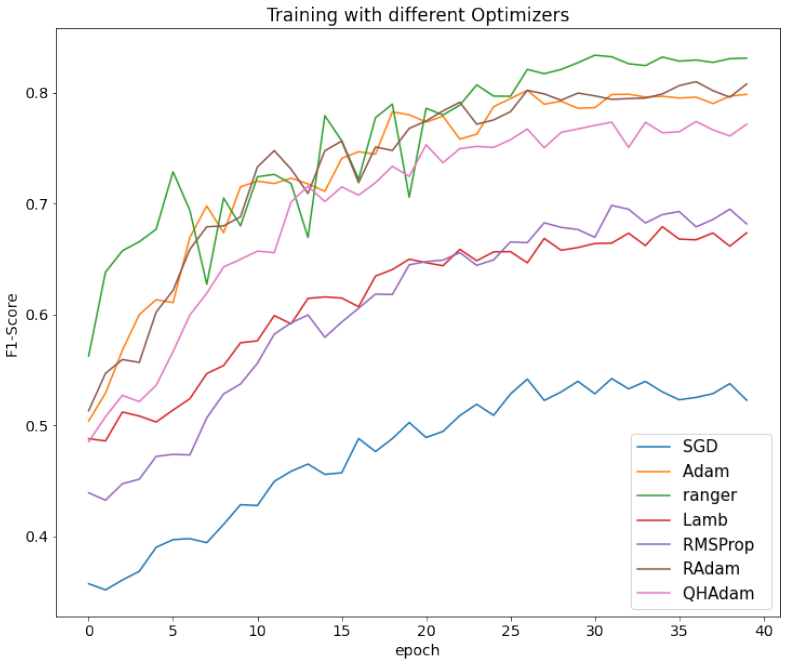

Abbildung 8: Verlauf des F1-Scores bei verschiedenen Optimizern über 40 Epochen

Daraus lässt sich schließen, dass die Wahl des Optimizers bei einem *imbalanced Dataset* einen großen Einfluss auf die Güte des Modells haben kann und daher in jedem Anwendungsfall untersucht werden sollte.

4.5 Abschlusstraining

Am Ende dieser Arbeit wurde noch ein Training durchgeführt, welches aus den Erkenntnissen der vorherigen Versuchen gespeist wird. Deshalb werden die erfolgreichsten Methoden daraus verwendet. Dazu zählen neben den Metriken *F1-Score*, *Recall* und *Cohen-Kappa* Score der Optimizer *ranger* und die Loss-Funktion *LabelSmoothing*. Dieses Training hat nach 40 Epochen die folgenden Ergebnisse erreicht:

epoch	**train-loss**	**valid-loss**	**F1-Score**	**Recall**	**Cohen-Kappa**
40	0.595	0.117	0.948	0.944	0.957

Diese Ergebnisse sind im F1-Score deutlich besser mit einem Wert von 0.948, statt 0.777 in der Baseline (siehe Abbildung 4, Kapitel 3.1).

5 Ausblick

Dieser Experimente zeigen, dass die Unterschiede im Verlauf des Trainings durch sehr viele Verschiedenen Parameter beeinflusst werden können. Gleichzeitig hat sich auch gezeigt, dass diese voneinenander anhängen. Beispielsweise kann eine Loss Funktion besonders geeignet sein für unausgeglichene Datasets, wenn jedoch Oversampling eingesetzt wird, kann eine alternative Verlustfunktion besser geeignet sein. Daher handelt es sich um ein sehr komplexes Arrangement von Einstellungen, die man nicht ausschließlich gesondert evaluieren sollte, sondern auch die unzähligen Kombinationen die sich daraus ergeben. Jedoch kann man sagen, dass durch eine Untersuchung dieser Methoden die Ergebnisse deutlich verbessern kann und somit ein sehr großes Potential für eine Klassifikationsaufgabe hat.

Literatur

[1] P. Tschandl, *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions*, Version V3, 2018. DOI: 10.7910/DVN/DBW86T. Adresse: https://doi.org/10.7910/DVN/DBW86T.

[2] A. Arora. (2020). "What is Focal Loss and when should you use it?" Accessed on 08.06.2022, Adresse: https://amaarora.github.io/2020/06/29/FocalLoss.html (besucht am 29. 06. 2020).

[3] *Fastai Python Library v2 Documentation*, https://docs.fast.ai/losses.html, Accessed on 08.06.2022, 2019.

[4] L. Wright. (2019). "New Deep Learning Optimizer, Ranger: Synergistic combination of RAdam + LookAhead for the best of both." Accessed on 13.06.2022, Adresse: https://lessw.medium.com/new-deep-learning-optimizer-ranger-synergistic-combination-of-radam-lookahead-for-the-best-of-2dc83f79a48d (besucht am 20. 08. 2019).

[5] W. Wong. (2019). "What is Label Smoothing?" Accessed on 13.06.2022, Adresse: https://towardsdatascience.com/what-is-label-smoothing-108debd7ef06 (besucht am 17. 12. 2019).