DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Entwerfen und Testen eines neuronalen Netzes mithilfe des MNIST-Datensatzes

Zielsetzung: Verbesserung der Accuracy

Projektarbeit des Moduls Digitale Bildverarbeitung

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Valentin Berisha

Abgabedatum:	4. Januar 2024
Bearbeitungszeitraum:	25.10.2023 - 05.01.2024
Matrikelnummer:	2014335
Kurs:	TFE21-2

Erklärung

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017 in der Fassung vom 25.07.2018.

Ich versichere hiermit, dass ich meine Bachelorarbeit (bzw. Projektarbeit oder Studienarbeit bzw. Hausarbeit) mit dem Thema:

> *Entwerfen und Testen eines neuronalen Netzes mithilfe des MNIST-Datensatzes - Zielsetzung: Verbesserung der Accuracy*

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Friedrichshafen, den 4. Januar 2024

_____

Valentin Berisha

Inhaltsverzeichnis

1 Ausgangslage und Zielsetzung 1

2 Umsetzung 3

3 Ergebnis und Auswertung 5

4 Zusammenfassung und Fazit 7

Literatur 8

Abbildungsverzeichnis 9

Anhang 10

1 Ausgangslage und Zielsetzung

Die Projektarbeit widmet sich dem Entwurf, Training und Testen eines neuronales Netzes zur Erkennung und Vorhersagen von handgeschriebenen Ziffern zwischen 0 bis 9 aus dem MNIST-Datensatz, mit abschließender Analyse und Evaluation. Dieser Satz wird mithilfe der keras-Bibliothek in Verbindung mit dem Tensorflow-Framework geladen. Dabei soll das Netz auf 60.000 Daten trainiert und anschließend auf 10.000 Daten validiert werden. Die Evaluierung erfolgt anhand von Genauigkeit (Accuracy) und Abweichung (Loss) im Vergleich zu den Soll-Daten.
Die grundlegende Art ein Netzwerk beruhend auf den MNIST-Datensatz zu konzipieren, ist als Single-Layer-Perceptron (SLP), das One-Hot-Encoding verwendet. Das Input-Layer umfasst 28 mal 28 Neuronen, die jeden einzelnen Pixel repräsentieren. Diese Neuronen sind direkt, ohne Hidden-Layer, mit dem Output-Layer verknüpft, welches wiederum 10 Neuronen enthält und für die verschiedenen Klassen (in diesem Fall Ziffern von 0 bis 9) stehen. Das Neuron, das die zutreffende Ziffer repräsentiert, wird auf 1, während alle anderen auf 0 gesetzt werden. SLPs können für das Lernen von einfachen Mustern angewendet werden. Für tiefere, detaillierte Mustererkennung und somit für bessere Genauigkeit bzw. besserer Vorhersage werden komplexere, mehrschichtige Netzwerke benötigt, wie z.B. Convolutional Neural Network (CNN). Das bereits vorhandene ‚marvin'-Netz umfasst ein Input-Flatten-Layer, ein Hidden-Dense-Layer mit 128 Neuronen und ein Output-Dense-Layer mit 10 Neuronen. Daneben enthält dieses auch ein Dropout-Layer, welches eingefügt wird, um während des Trainings zufällig bestimmte Neuronen auszusetzen oder zu deaktivieren. Dieses Netz weist eine, bei einer Batch-Size von 8 und 10 Epochen eine Accuracy von 97,78% auf die Testdaten. Auffällig ist, dass die Accuracy auf die Trainingsdaten höher liegt (98,29%) und es somit zur Overfitting kommt. Das Modell passt während des Trainings zu gut auf die Trainingsdaten, sodass es Schwierigkeiten aufweist, auf neue, bisher ungesehenen Daten gut zu

generalisieren. Das Netz soll nicht nur gut auswendig lernen, sondern die grundlegenden Muster wirklich verstehen. Die Problemstellung und die Zielsetzung bzw. der Fokus der Arbeit liegt darin, durch geeignete Methoden die Genauigkeit des neuronalen Netzes bei gleichzeitigem Verringern des Losses zu erhöhen. Dafür soll das bereits vorhandene Netz erweitert werden und anhand von Gegenüberstellungen die Methoden erklärt, sowie Vor- und Nachteile erläutert werden. Außerdem sollen geeignete Gegenmaßnahmen definiert werden, um Overfitting zu verringern und die Generalisierungsfähigkeit des Modells zu erhöhen. Das Modell heißt ‚bambi‘, dessen Testergebnisse in 3 mit den des ‚marvin‘ Modells ausgewertet werden.

2 Umsetzung

Im Folgenden sollen die Methoden dargestellt und erklärt werden, die zu einer Erhöhung der Accuracy und Generalisierungsfähigkeit führen. Zuerst wird festgelegt, dass ein CNN-Netz, auch ConvNet genannt, verwendet wird. Diese Anpassung der Netzart ist ausschlaggebend für die Erhöhung der Accuracy des neuronalen Netzes [Sco23]. Das Netzwerk umfasst Convolutional-Layer zu Mustererkennung, Pooling-Layer zur Dimensionalitätsreduktion, Flatten-Layer zur Umwandlung in eindimensionale Daten und Dense-Layer für die Erfassung der Daten. Während Dense-Layer nur globale Muster aller Pixel lernen können, haben Convolutional-Layer die Möglichkeit lokale Muster aller Pixel zu lernen. Durch Einfügen des Convolutional-Layers (Faltungsschichts) lernt das neuronale Netz nicht die Gewichte/Parameter aller Verbindungen zwischen Neuronen, sondern konzentriert sich auf die Gewichte eines sogenannten Kernel-Filters. Diese Strategie hat mehrere Vorteile, die dazu beitragen, das Training schneller und effizienter zu gestalten. Der Kernel-Filter wird über die Eingangsdaten geschoben und es wird ein entsprechender lokaler Wert mithilfe Faltung berechnet (Feautere Map). Der selbe Filter wird somit auf verschiedene Bereiche des Bildes angewendet, wodurch die Anzahl der zu lernenden Parameter erheblich reduziert wird. Pooling-Layer werden in Kombination mit den Convolutional-Layer verwendet, um die Eingabe zu verkleinern und somit die Dimensionalität zu verringern. Im vorliegenden Fall werden Max-Pooling-Layer verwendet. Insgesamt werden 3 Convolution- und 3 Pooling-Layer verwendet, wobei die richtige Auswahl der Faktoren Filterzahl und Kernel-Size ebenso zur Steigerung der Effizienz des Modells beitragen. Eine große Filterzahl, hier 64, benötigt zwar eine hohe Rechenzeit, jedoch bietet sie dem Modell eine höhere Flexibilität in Bezug auf die Repräsentation der Daten und eine bessere Generalisierung. Die Kernel-Size wird auf 5 (5x5) gesetzt.
Als weitere Maßnahme zur Verbesserung der Accuracy eignet sich durch das Hinzufügen von mehreren Dense-Layern. Während im ‚marvin'-Netz nur ein Hidden-Layer mit 128

Neuronen verwendet wird, umfasst das ‚bambi'-Netz 3 Hidden-Dense-Layer. Die Anzahl der Hidden-Layer und die damit verbundene Anzahl an Neuronen sind Hyperparameter, welche die Leistung des neuronalen Netzes stark beeinflussen. Netzwerke mit mehreren versteckten Schichten haben die Fähigkeit komplexere Muster in den Daten zu erlernen und Vorhersagen mit höherer Genauigkeit zu treffen. Dies liegt daran, dass im Gegensatz zu einfachen Single-Perceptron-Netzen, haben Hidden-Layer in mehrschichtigen Netzwerken die Eigenschaft auch nichtlineare Beziehungen zwischen den Daten zu erkennen, was somit zu einer höheren Varianz des Modells führt und eine bessere Anpassung an komplexe Zusammenhänge in den Daten ermöglicht. Die Integration der Nichtlinearitäten in das Netz wird durch Aktivierungsfunktionen erreicht. Das ‚bambi'-Netz verwendet überwiegend die ReLu-Funktion. Sie bietet den Vorteil, dass sie alle negative Werte auf 0 setzt und positive Werte unverändert lässt. Für die Output-Schicht wird softmax verwendet, da diese Funktion die Ausgaben so skaliert, dass sie als Wahrscheinlichkeiten interpretiert werden können und die Summe aller Wahrscheinlichkeiten 1 ergibt.
Ebenso wichtig, wie die Leistungsfähigkeit des Netzes zu verbessern, ist Overfitting zu vermeiden. Eine Überanpassung an die Trainingsdaten hat zur Folge, dass das Netz Schwierigkeiten hat neue, bisher ungesehene Daten gut zu generalisieren. Als Gegenmaßnahme werden Dropout-Layer an sämtlichen Stellen des Modells eingefügt. Während bestimmter Forward- und Backpropagation (innerhalb einer Epoche) wird eine gewisse Anzahl an Neuronen ausgesetzt. Dadurch wird verhindert, dass das Modell zu sehr auf bestimmte Gewichte stützt. Das Netz wird gezwungen mit unvollständigen Daten zu arbeiten, was eine höhere Robustheit und eine bessere Generalisierungsfähigkeit zufolge hat. Dem ‚bambi'-Modell werden Dropout-Layer mit einer Auslassrate von 0,3 eingefügt, was bedeutet, dass während des Trainings 30% der Neuronen zufällig deaktiviert werden. Zusätzlich werden Batch-Normalization-Layer hinzugefügt, um das Training des neuronalen Netzes, durch Normalisierung der Aktivierungsfunktionen, zu verbessern und zu beschleunigen [Iof15]. Auch sie verringern die Chance des Overfittings. Durch Anpassung der Hyperparameter, wie Epochenzahl und Batch-Size lässt sich ebenfalls die Leistungsfähigkeit erhöhen. Eine zu kleine Epochenzahl kann dazu führen, dass das Netz nicht ausreichend trainiert wird, wohingegen eine zo große Epochenzahl zu Overfitting führen kann. Bei einer kleinen Batch-Size wird das Training verlangsamt, zu hohe Batch-Size führt zu Ungenauigkeiten. Im vorgegebenen Netz wird eine Epochenzahl von 10 und eine Batch-Size von 8 verwendet. Dies wird im ‚bambi'-Netz auf 25 und 32 angepasst.

3 Ergebnis und Auswertung

In diesem Kapitel sollen sowohl die Ergebnisse des ‚bambi'-Netzes als auch die des ‚marvin'-Modells anhand von Diagrammen dargestellt und erläutert werden. Angemerkt sei, dass weder die Struktur noch die Hyperparameter von ‚marvin' abgeändert wurden.

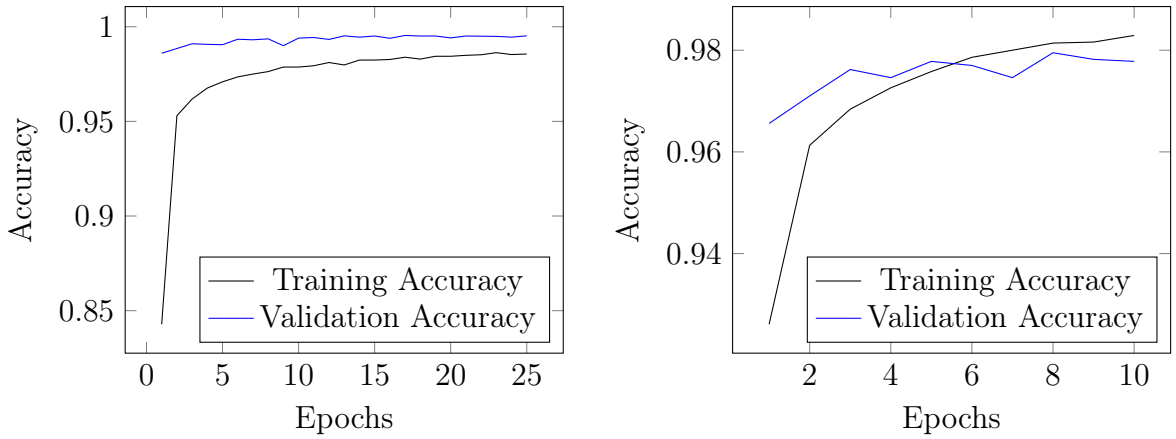

Abbildung 3.1: Accuracy ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts)

In Abbildung 3.1 wird der Accuracy-Unterschied zwischen den beiden Netzen und die Überanpassung des ‚marvin'-Modells deutlich. Während ‚bambi' bereits nach 10 Epochen eine Validation-Accuracy von ca. 99,5% erreicht (99,52% nach 25 Epochen), kommt ‚marvin' auf gerade mal 97,8%. Außerdem ist es ersichtlich, dass das ‚marvin'-Netzwerk overfitted, da die Accuracy der Trainingsdaten höher liegt als die der noch ungesehenen Validation-Daten, wohingegen die Traingsaccuracy des ‚bambi'-Netzes sich der Validationaccuracy mit zunehmender Epochenzahl annähert, diese jedoch nie überschreitet. Dies liegt daran, dass das ‚bambi' Modell deutlich mehr Dropout-Layer und andere Maßnahmen gegen Overfitting verwendet. Mit diesen Maßnahmen geht allerdings auch wie in 2 erklärt einher, dass das Training am Anfang deutlich erschwert wird, weshalb, wie in

Abbildung 3.1 zu erkennen, die Accuracy auf die Trainingsdaten zu Beginn niedriger ist als die des ‚marvin'-Netzes. Außerdem ist noch zu erwähnen, dass die Hyperparameter Batch-Size und Epochenzahl für das ‚bambi'-Modell gut gewählt wurde, denn es gibt keine großen Schwankungen und die Validation-Accuracy hält sich annähernd konstant. Ähnliche Überlegungen gelten auch für die Verlustdiagramme. Auch hier (in 3.2 rechts) lässt sich die Überanpassung erkennen, zudem konnte der Validation-Loss durch das neue Modell im Vergleich zum bereits vorhandenen Modell verringert werden (5,1%).

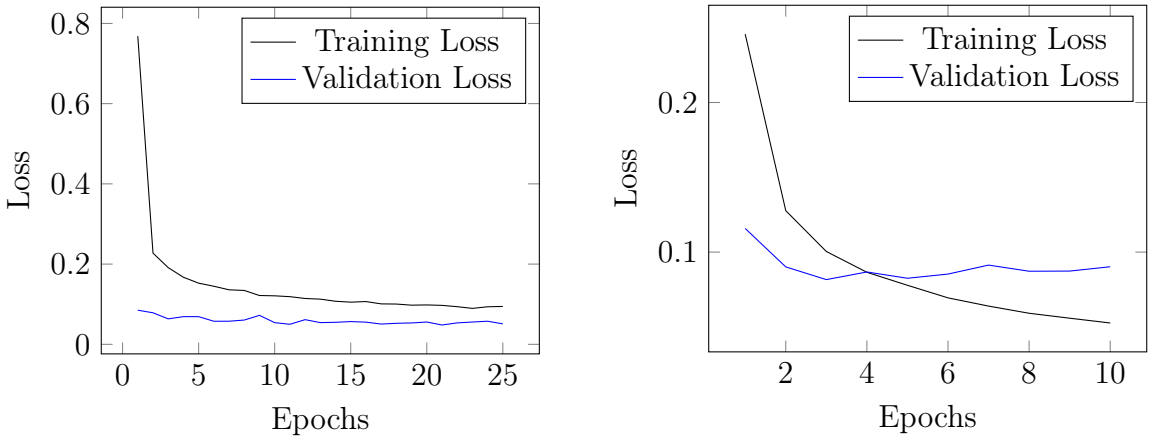

Abbildung 3.2: Loss ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts)

Auch in der Heatmap werden die Unterschiede zwischen den zwei neuronalen Netzen deutlich. Auf der Hauptdiagonale lässt sich erkennen, dass jede Ziffer besser vorhergesagt werden kann. Des Weiteren sind die Ausreißer deutlich niedriger.

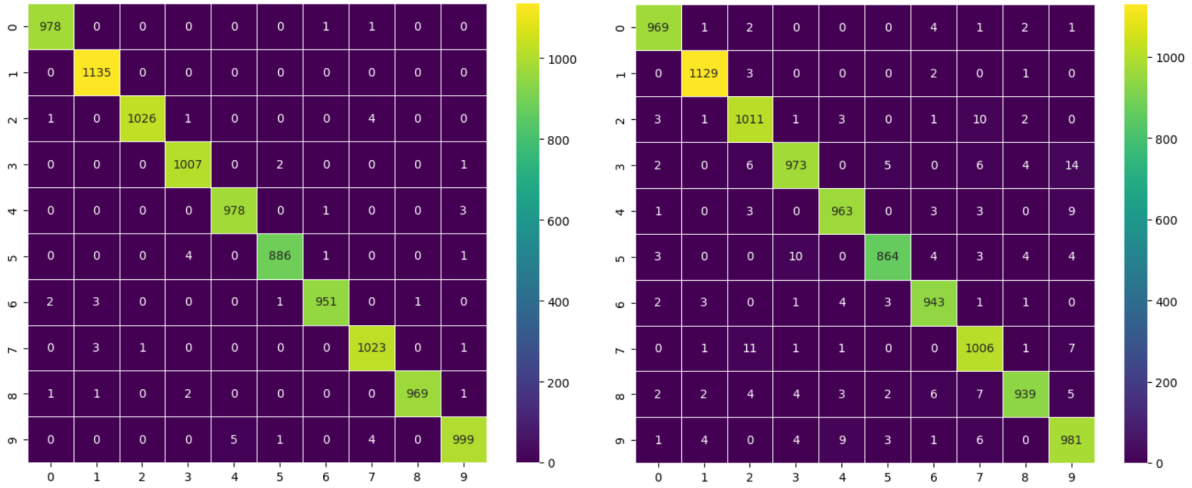

Abbildung 3.3: Heatmap ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts)

4 Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass ein neuronales Netz entwickelt wurde, welches der Zielsetzung gerecht wird und dass die Problemstellung gut umgesetzt wurde. Durch den Aufbau des Netzes als ConvNet und durch Implementierung von Gegenmaßnahmen, kann Overfitting vermieden und eine gute Generalisierungsfähigkeit gewährleistet werden. Demzufolge liefert das Netz eine Accuracy von etwas über 99,5% und die Fähigkeit Ziffer sicher vorherzusagen. Neben der Anpassung der Netzart und der Layeranzahl, tragen die Anpassungen der Hyperparameter, wie Batch-Size und Epochenzahl, ebenso zur Stabilität und Leistungsfähigkeit des Modells bei. Insgesamt konnten alle negativen Aspekte des bereits vorhandenen Netzes ausgebessert werden. Aufgrund der Fokus der Arbeit auf hohe Genauigkeit, wurden die anderen Metriken Parameter- und Labelzahl nicht berücksichtigt.

Literatur

[Iof15] Ioffe, Sergey and Szegedy, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. URL: https://arxiv.org/abs/1502.03167 (besucht am 03. 01. 2024).

[Sco23] Scodellaro, Riccardo und Schroeter, Matthias. *Training Convolutional Neural Networks with the Forward-Forward Algorithm*. Hrsg. von Journal of Machine Learning Research. 2023. URL: https://arxiv.org/abs/2312.14924 (besucht am 03. 01. 2024).

Abbildungsverzeichnis

3.1 Accuracy ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts) 5
3.2 Loss ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts) 6
3.3 Heatmap ‚bambi'-Netzwerk (links) und ‚marvin'-Netzwerk (rechts) 6

A.1 Accuracy und Loss ‚marvin'-Netzwerk 10
A.2 Accuracy und Loss ‚bambi'-Netzwerk 11

Anhang

```
Epoch 1/10
7500/7500 [==============================] - 127s 16ms/step - loss: 0.2457 - accuracy: 0.9261 - val_loss: 0.1158 - val_accuracy: 0.9656
Epoch 2/10
7500/7500 [==============================] - 117s 16ms/step - loss: 0.1277 - accuracy: 0.9613 - val_loss: 0.0901 - val_accuracy: 0.9710
Epoch 3/10
7500/7500 [==============================] - 113s 15ms/step - loss: 0.1004 - accuracy: 0.9684 - val_loss: 0.0816 - val_accuracy: 0.9762
Epoch 4/10
7500/7500 [==============================] - 118s 16ms/step - loss: 0.0865 - accuracy: 0.9726 - val_loss: 0.0867 - val_accuracy: 0.9746
Epoch 5/10
7500/7500 [==============================] - 114s 15ms/step - loss: 0.0778 - accuracy: 0.9758 - val_loss: 0.0825 - val_accuracy: 0.9778
Epoch 6/10
7500/7500 [==============================] - 121s 16ms/step - loss: 0.0694 - accuracy: 0.9786 - val_loss: 0.0853 - val_accuracy: 0.9770
Epoch 7/10
7500/7500 [==============================] - 123s 16ms/step - loss: 0.0639 - accuracy: 0.9800 - val_loss: 0.0913 - val_accuracy: 0.9746
Epoch 8/10
7500/7500 [==============================] - 106s 14ms/step - loss: 0.0591 - accuracy: 0.9814 - val_loss: 0.0872 - val_accuracy: 0.9795
Epoch 9/10
7500/7500 [==============================] - 94s 13ms/step - loss: 0.0558 - accuracy: 0.9816 - val_loss: 0.0873 - val_accuracy: 0.9782
Epoch 10/10
7500/7500 [==============================] - 100s 13ms/step - loss: 0.0526 - accuracy: 0.9829 - val_loss: 0.0902 - val_accuracy: 0.9778
```

Abbildung A.1: Accuracy und Loss ‚marvin'-Netzwerk

```
1875/1875 [==============================] - 250s 132ms/step - loss: 0.7535 - accuracy: 0.8475 - val_loss: 0.0898 - val_accuracy: 0.9839
Epoch 2/25
1875/1875 [==============================] - 243s 130ms/step - loss: 0.2212 - accuracy: 0.9533 - val_loss: 0.0825 - val_accuracy: 0.9867
Epoch 3/25
1875/1875 [==============================] - 239s 128ms/step - loss: 0.1848 - accuracy: 0.9623 - val_loss: 0.0802 - val_accuracy: 0.9878
Epoch 4/25
1875/1875 [==============================] - 252s 135ms/step - loss: 0.1678 - accuracy: 0.9674 - val_loss: 0.0664 - val_accuracy: 0.9910
Epoch 5/25
1875/1875 [==============================] - 240s 128ms/step - loss: 0.1514 - accuracy: 0.9711 - val_loss: 0.0654 - val_accuracy: 0.9914
Epoch 6/25
1875/1875 [==============================] - 271s 144ms/step - loss: 0.1436 - accuracy: 0.9738 - val_loss: 0.0651 - val_accuracy: 0.9918
Epoch 7/25
1875/1875 [==============================] - 222s 118ms/step - loss: 0.1353 - accuracy: 0.9758 - val_loss: 0.0514 - val_accuracy: 0.9936
Epoch 8/25
1875/1875 [==============================] - 221s 118ms/step - loss: 0.1227 - accuracy: 0.9781 - val_loss: 0.0580 - val_accuracy: 0.9924
Epoch 9/25
1875/1875 [==============================] - 208s 111ms/step - loss: 0.1231 - accuracy: 0.9780 - val_loss: 0.0533 - val_accuracy: 0.9936
Epoch 10/25
1875/1875 [==============================] - 206s 110ms/step - loss: 0.1232 - accuracy: 0.9789 - val_loss: 0.0516 - val_accuracy: 0.9951
Epoch 11/25
1875/1875 [==============================] - 208s 111ms/step - loss: 0.1151 - accuracy: 0.9810 - val_loss: 0.0563 - val_accuracy: 0.9944
Epoch 12/25
1875/1875 [==============================] - 210s 112ms/step - loss: 0.1107 - accuracy: 0.9810 - val_loss: 0.0542 - val_accuracy: 0.9942
Epoch 13/25
1875/1875 [==============================] - 209s 111ms/step - loss: 0.1089 - accuracy: 0.9815 - val_loss: 0.0493 - val_accuracy: 0.9946
Epoch 14/25
1875/1875 [==============================] - 208s 111ms/step - loss: 0.1082 - accuracy: 0.9818 - val_loss: 0.0551 - val_accuracy: 0.9933
Epoch 15/25
1875/1875 [==============================] - 208s 111ms/step - loss: 0.1033 - accuracy: 0.9829 - val_loss: 0.0572 - val_accuracy: 0.9947
Epoch 16/25
1875/1875 [==============================] - 206s 110ms/step - loss: 0.1071 - accuracy: 0.9827 - val_loss: 0.0549 - val_accuracy: 0.9945
Epoch 17/25
1875/1875 [==============================] - 209s 112ms/step - loss: 0.1013 - accuracy: 0.9836 - val_loss: 0.0545 - val_accuracy: 0.9941
Epoch 18/25
1875/1875 [==============================] - 209s 112ms/step - loss: 0.1000 - accuracy: 0.9845 - val_loss: 0.0593 - val_accuracy: 0.9942
Epoch 19/25
1875/1875 [==============================] - 223s 119ms/step - loss: 0.0979 - accuracy: 0.9848 - val_loss: 0.0501 - val_accuracy: 0.9943
Epoch 20/25
1875/1875 [==============================] - 213s 114ms/step - loss: 0.0953 - accuracy: 0.9846 - val_loss: 0.0549 - val_accuracy: 0.9944
Epoch 21/25
1875/1875 [==============================] - 211s 112ms/step - loss: 0.0952 - accuracy: 0.9848 - val_loss: 0.0575 - val_accuracy: 0.9944
Epoch 22/25
1875/1875 [==============================] - 207s 111ms/step - loss: 0.0948 - accuracy: 0.9855 - val_loss: 0.0610 - val_accuracy: 0.9945
Epoch 23/25
1875/1875 [==============================] - 211s 113ms/step - loss: 0.0902 - accuracy: 0.9857 - val_loss: 0.0535 - val_accuracy: 0.9945
Epoch 24/25
1875/1875 [==============================] - 244s 130ms/step - loss: 0.0919 - accuracy: 0.9855 - val_loss: 0.0546 - val_accuracy: 0.9952
Epoch 25/25
1875/1875 [==============================] - 241s 129ms/step - loss: 0.0870 - accuracy: 0.9865 - val_loss: 0.0576 - val_accuracy: 0.9951
```

Abbildung A.2: Accuracy und Loss ‚bambi'-Netzwerk