KLASSIFIKATION VON GEBÄRDENSPRACHE DURCH EIN CONVOLUTIONAL NEURAL NETWORK: EINE STUDIE AUF DEM SIGN LANGUAGE MNIST-DATENSATZ

Felix Widmann
Duale Hochschule Baden-Württemberg
Stuttgart
wi21156@lehre.dhbw-stuttgart.de

ABSTRACT

Die korrekte Klassifikation von Handgesten stellt einen vielversprechenden Ansatz dar, um die Sprachbarriere zwischen hörenden und gehörlosen Menschen zu überwinden. In diesem Projekt wird daher die Verwendung von Convolutional Neural Networks (CNNs) zur Klassifizierung von Handgesten untersucht. Der zugrunde liegende Datensatz, Sign Language MNIST, stammt von Kaggle und besteht aus 34.627 Bildern von Handgesten, die alle Buchstaben der amerikanischen Gebärdensprache (ASL) abbilden. Durch systematisches Hyperparameter-Tuning mit Weights and Bias Sweeps konnte dabei ein weighted average F1-Score von 94.9% auf dem Datensatz erreicht werden. Die Ergebnisse zeigen, dass CNNs die Merkmale einzelner Handgesten erfassen können und eine vielversprechende Lösung für die automatische Bildklassifikation bieten. Der Code ist verfügbar auf Github.

***K*eywords** Bildklassifikation · Convolutional Neural Network · Gebärdensprache

1 Einleitung

Handgesten sind seit Anbeginn der menschlichen Zivilisation ein fundamentales Mittel der Kommunikation. Dennoch besteht eine erhebliche Kommunikationsbarriere zwischen der gehörlosen Gemeinschaft und der hörenden Mehrheit, da weltweit lediglich etwa 70 Millionen Menschen Gebärdensprache sprechen oder verstehen können. Ein vielversprechender Ansatz zur Überwindung dieser Barriere ist die Bildklassifikation mittels maschinellen Lernens.

In diesem Projekt wird ein CNN genutzt, um Gebärdensprache aus Bildern zu klassifizieren. Der zugrunde liegende Datensatz ist der Sign Language MNIST Datensatz, welcher Graubilder von Handgesten der einzelnen Buchstaben des amerikanischen Alphabets enthält. Ziel dieses Projekts ist es, ein CNN zu trainieren, das die korrekte Klassifikation der Handgesten ermöglicht und dadurch Gebärdensprache in Text überführt.

Die Resultate zeigen, dass das CNN in der Lage ist, die Feinheiten einzelner Zeichen der Gebärdensprache richtig zu klassifizieren, wobei ein weighted F1-Score von 94.9% auf dem Test-Datensatz erreicht wurde.

2 Experiment

2.1 Sign Language MNIST

Der zugrunde liegende Datensatz ist der Sign Language MNIST-Datensatz von Kaggle [1]. Jedes Trainings- und Testbeispiel besteht aus einem Label und 784 Pixelwerten, die ein einzelnes 28x28-Pixel-Bild mit Graustufenwerten zwischen 0 und 255 darstellen. Die Sign Language MNIST-Daten entstanden durch eine erhebliche Erweiterung der kleinen Anzahl (1704) von Farbbildern, die ursprünglich nicht auf den Handbereich zugeschnitten waren. Um neue Daten zu erstellen, wurden eine 5% zufällige Pixelation, +/- 15% Helligkeit/Kontrast und schließlich eine Rotation um 3 Grad genutzt. Jedes Trainings- und Testbeispiel stellt ein Label im Bereich von 0 bis 25 dar. Dies ist eine

Figure 1: Beispiele aus dem Sign Language MNIST-Datensatz in Reihenfolge. Von Links nach Rechts: Darstellung der Buchstaben A bis Z in Gebärdensprache, ausgenommen J und Z

Eins-zu-eins-Zuordnung für jeden Buchstaben von A bis Z (A=0, B=1,..., Z=25). Allerdings besitzt der Datensatz keine Fälle für 9=J und 25=Z, da diese Buchstaben Handbewegungen beinhalten, die in einem Bild nicht darstellbar sind. Der Datensatz ist bereits in Trainings- und Testdaten unterteilt. Die Trainingsdaten umfassen 27.455 Beispiele und die Testdaten 7.172 Beispiele, was einem Trainings-/Test-Split von etwa 80/20 entspricht. Eine grafische Darstellung der Einträge ist in Abbildung 1 gegeben.

2.2 Datenbereinigung und Vorverarbeitung

Grundsätzlich muss berücksichtigt werden, dass es sich bei dem Sign Language MNIST-Datensatz um einen populären Datensatz handelt, der dementsprechend bereits gut kuratiert und vorverarbeitet ist. Demnach ist eine exzessive Datenaufbereitung nicht notwendig.

Die Betrachtung der Verteilung der Beispiele je Label in den Test- und Trainingsdaten zeigt, dass diese ungefähr ausgeglichen auf die einzelnen Labels verteilt sind. Die Verteilung ist in Abbildung 2 dargestellt. Der Trainingsdatensatz umfasst etwa 1000 Beispiele pro Label und der Testdatensatz etwa 300 pro Label.

Aufgrund der ausbalancierten Datensätze sind weitere Augmentations- oder Up/Down-Sampling-Strategien nicht notwendig. Des Weiteren wurden, wie beschrieben, bei der Erstellung des Datensatzes bereits Methoden der Datenaugmentation genutzt, wie Pixelation, Kontrastveränderung und Rotation.

Datenkonvertierung Die Trainings- und Testdaten sind ursprünglich im CSV-Format, wobei ein Bild durch eine Zeile mit 785 Einträgen dargestellt wird, die dem Label und den 784 Pixeln entsprechen. Diese müssen für die Bildklassifikation in das 28x28-Bildformat konvertiert werden. Dazu werden die Daten in das Format (Anzahl der Bilder, 1 Kanal, 28x28 Pixel) umgeformt, da das CNN Eingaben in dieser Form erwartet. Die Anzahl der Kanäle ist 1, da es sich hier um Graubilder handelt, die die räumlichen Dimensionen 28 auf 28 Pixel umfassen.

Die Labels (Zielwerte) werden aus der ersten Spalte des Datensatzes extrahiert und in LongTensor-Objekte umgewandelt. Da die Labels bereits numerische Werte im Bereich von 0 bis 25 darstellen, können sie direkt in einen LongTensor umgewandelt werden. In PyTorch ist LongTensor das übliche Format für Klassifikationslabels und stellt sicher, dass sie korrekt von der Verlustfunktion Cross Entropy Loss und während des Modelltrainings verarbeitet werden.

Normalisierung Die Pixelwerte der einzelnen Graubilder, die zwischen 0 und 255 liegen, wurden mithilfe des Robust Scalers normalisiert. Diese Normalisierungsmethode skaliert die Pixelwerte auf einen standardisierten Bereich, was zur

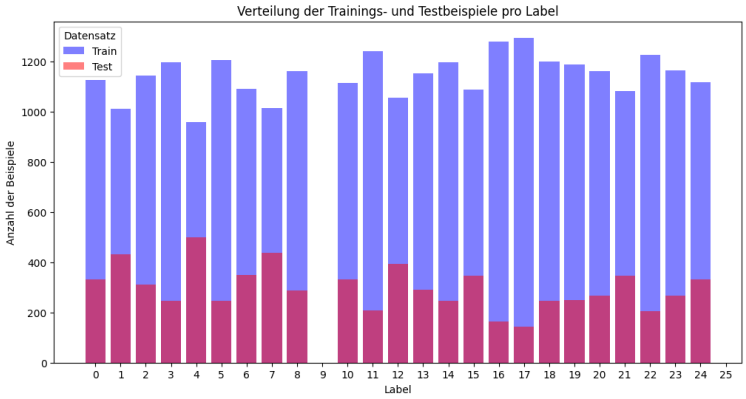

Figure 2: Anzahl der Beispiele je Label in dem Trainings-und Testdatensatz. Das Label entspricht dem Index des jeweiligen Buchstaben, wobei 0=A und 25=Z.

Stabilisierung und Beschleunigung des Trainingsprozesses beiträgt. Die Normalisierung erfolgte nach der folgenden Formel:

$$x' = \frac{x - \text{Median}(x)}{\text{IQR}(x)}$$

wobei x der ursprüngliche Pixelwert ist, Median(x) der Median der Pixelwerte des Datensatzes und IQR(x) der Interquartilbereich der Pixelwerte ist. Durch die Normalisierung werden die Daten so transformiert, dass Ausreißer weniger Einfluss haben, was zu einer robusteren und genaueren Modellleistung führt. Indem die Eingabewerte auf eine ähnliche Skala gebracht werden, kann das Modell schneller konvergieren und eine bessere Leistung erzielen, da Probleme im Zusammenhang mit unterschiedlichen Größenordnungen der Eingabewerte vermieden werden.

Um die Daten für das Training und Testen des CNNs effizient zu laden, werden DataLoader-Objekte für den Trainings- und Testdatensatz erstellt. Der DataLoader ermöglicht das effiziente Laden und Shufflen der Daten in Batches, was für das Training und Testen von neuronalen Netzwerken entscheidend ist.

2.3 Auswahl der Zielmetrik

Ausgehend von der Beschaffenheit des Trainingsdatensatzes wird der Weighted Average F1-Score als Zielmetrik für die Bildklassifikation mithilfe des Convolutional Neural Networks genutzt. Der gewichtete F1-Score berechnet den F1-Score für jede Klasse unabhängig, verwendet jedoch beim Durchschnittsbilden ein Gewicht, das von der Anzahl der wahren Instanzen jeder Klasse abhängt. Auch wenn der Datensatz relativ ausgewogen ist, bietet die Wahl des gewichteten F1-Scores zusätzliche Sicherheit und berücksichtigt Ungleichgewichte in den Klassenverteilungen. So wird auch die größte Differenz in der Anzahl an Trainingsbeispielen je Label zwischen Label 4 (E) und 17 (R) von etwa 300 berücksichtigt. Dadurch wird verhindert, dass Klassen mit wenigen Instanzen unverhältnismäßig großen Einfluss auf die Gesamtleistung des Modells haben.

2.4 Model Architektur

Die verwendete CNN-Architektur wurde von [2] übernommen, die das Modell ursprünglich für den Digits MNIST-Datensatz verwendet haben. Grundsätzlich besteht das CNN aus drei Convolutional Layern, die ähnlich aufgebaut sind.

Convolutional Layer 1 Der erste Convolutional Layer nimmt das Graubild mit den Dimensionen 28 auf 28 als Eingabe. Dieser Layer wendet 64 Convolutional Filter, jeder von der Größe 5x5, an. Der Stride ist auf 1 und das Padding auf 2 definiert, wodurch die Feature Map die gleichen räumlichen Abmessungen wie die Eingabe hat. Dies führt zu einer Feature-Map mit den Abmessungen 28x28x64. Nach der Faltungsoperation wird die ReLU-Aktivierungsfunktion (Rectified Linear Unit) auf die Feature-Map angewendet, um Nichtlinearität einzuführen, die dem Netz hilft, komplexere Muster in den Daten zu lernen. Nach der ReLU-Aktivierung wird eine Max-Pooling-Operation mit einer Kernelgröße von 2x2 und einem Stride von 2 durchgeführt. Diese Pooling-Operation reduziert die räumlichen Abmessungen der

Feature-Map von 28x28x64 auf 14x14x64, wodurch die Feature-Map effektiv verkleinert und die Rechenlast für die nachfolgenden Schichten verringert wird. Die resultierende Anzahl der trainierbaren Parameter ist 5564 = 1600.

Convolutional Layer 2 Der zweite Convolutional Layer nimmt die 64 Feature-Maps aus der vorherigen Schicht und wendet einen weiteren Satz von 64 Convolutional-Filtern der Dimension 5x5x64 (da die Input-Dimension 64 beträgt) mit einem Stride von 1 und Padding von 2 an, gefolgt von einer ReLU-Aktivierungsfunktion und einer Max-Pooling-Schicht mit einer Kernelgröße von 2x2. Dies reduziert die Dimension der resultierenden Feature-Map auf 7x7x64. Die Gesamtzahl der trainierbaren Parameter dieses Layers beträgt 64*(5564) = 102400.

Convolutional Layer 3 Der letzte Convolutional Layer ist gleich aufgebaut wie der zweite Convolutional Layer. Dieser nimmt ebenfalls die 64 Feature-Maps der vorherigen Schicht als Input und wendet 64 Convolutional Filter mit den Dimensionen 5x5x64 an. Anschließend wird die ReLU-Aktivierungsfunktion angewendet, gefolgt von einer Max-Pooling-Schicht mit einer Kernelgröße von 2x2. Dies reduziert die Dimension der resultierenden Feature-Map auf 4x4x64. Die Gesamtzahl der trainierbaren Parameter dieses Layers beträgt ebenfalls 64*(5564) = 102400.

Fully-connected Layer 1 Diese Schicht nimmt die abgeflachte Feature-Map des letzten Convolutional Layers mit den Dimensionen 4x4x64 als Eingangsvektor, was in einem Eingangsvektor der Dimension 1024 resultiert. Die Ausgangsdimension ist auf 120 festgelegt. Zusätzlich wird ein Dropout von 0.5 angewendet, um Overfitting während des Trainings zu vermeiden.

Fully-connected Layer 2 Das letzte Fully-connected-Network nimmt den Ausgangsvektor des vorherigen Netzwerks mit der Dimension 120 als Eingangsvektor. Die finale Schicht besteht aus 25 Neuronen, welche die 24 zu klassifizierenden Klassen und J=9 umfasst. Diese Schicht liefert die finalen Klassifikationswerte der einzelnen Klassen. Da Cross-Entropy als Verlustfunktion verwendet wird, welche die reinen Logits des Ausgangslayers zur Berechnung nutzt und implizit eine Softmax-Operation durchführt, wird kein zusätzlicher Softmax-Layer benötigt.

2.5 Training

Training Da dies ein Multiklassen-Klassifikationsproblem darstellt, wird der Cross Entropy Loss als Verlustfunktion genutzt, die wie folgt definiert ist:

$$L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Für das Training des Modells wurde der Adam Optimizer genutzt, da er effiziente Konvergenz ermöglicht und gut mit spärlichen Gradienten umgeht. Die Lernrate wurde während des Trainings nicht angepasst.

Hyperparameter Für das Training und effektive Hyperparameter-Tuning wurde Weights & Biases als ML-Lebenszyklus-Programm zur Dokumentation der Trainingsmetriken und Hyperparameter verwendet. Da dieses Experiment als grundlegende Demonstration eines vollständigen Deep-Learning-Prozesses dient, wurde die Anzahl der einstellbaren Hyperparameter minimal gehalten. Die Epochenzahl in der Menge 10, 20, 30, die Batchsize in der Menge 16, 32, 64, 96 und die Lernrate in der Menge 0.1, 0.001, 0.0001 wurden mithilfe einer Zufallssuche evaluiert.

3 Ergebnisse

Das finale Modell wurde auf dem Test-Datensatz evaluiert und hat einen weighted F1-Score von 94.9% erzielt. Diese Ergebnisse wurden mit den Hyperparametern: Batchsize=32, Lernrate=0.001 und einer Epochenzahl von 30 erreicht. Eine Confusion-Matrix der erzielten Ergebnisse auf dem Test-Datensatz ist in Abbildung 3 gegeben. Dabei ist erkennbar, dass das Modell häufig die gleichen Klassifikationsfehler macht. Dies ist durch ähnliche Handstellungen in der Gebärdensprache zu erklären, insbesondere bei den Buchstaben e und s, welche beide eine Faust als Grundlage haben, jedoch durch unterschiedliche Fingerpositionen erkennbar sind (siehe Abbildung 1).

Die in den Abbildungen 4 und 5 dargestellten Graphen des weighted F1-Scores sowie der Verlustkurven für Training und Test zeigen, dass das Modell tatsächlich gelernt hat, die richtigen Buchstaben anhand der dazugehörigen Zeichen in Gebärdensprache aus den Bildern zu klassifizieren.

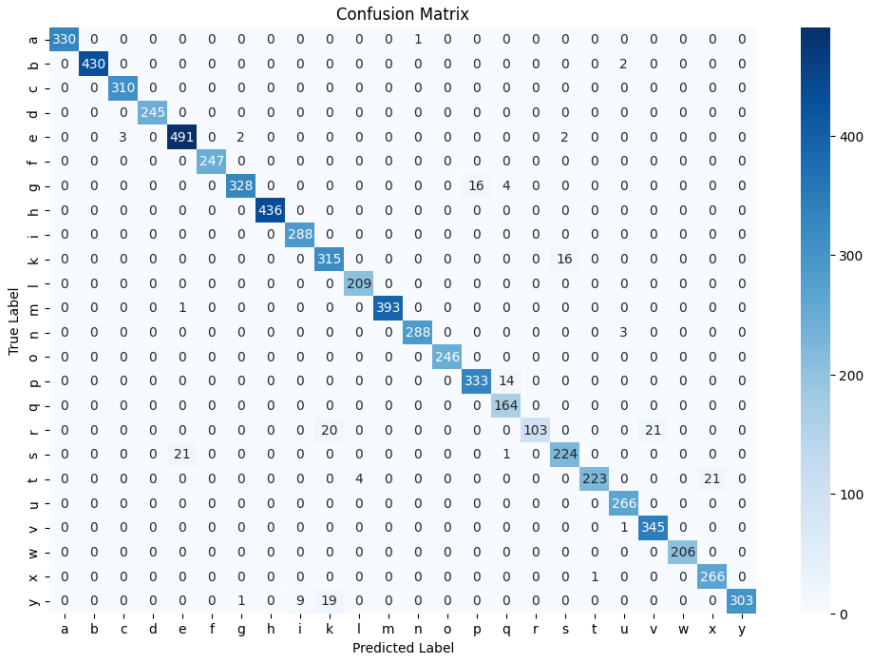

Figure 3: Confusion-Matrix

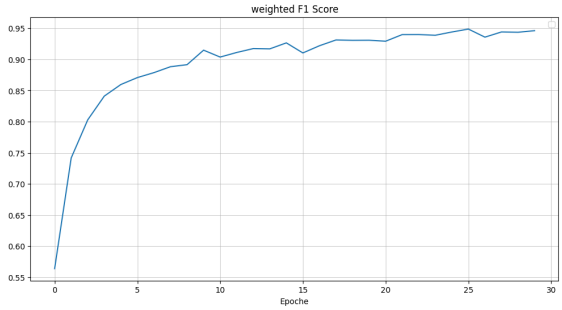

Figure 4: weighted F1 Score je Epoche

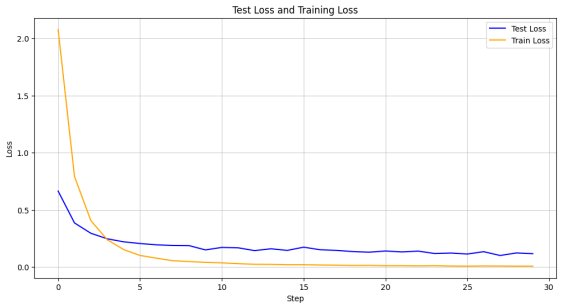

Figure 5: Test- und Trainingsverlust je Epoche

4 Schlussbetrachtung

In diesem Projekt wurde erfolgreich gezeigt, dass Convolutional Neural Networks (CNNs) eine effektive Methode zur Klassifikation von Handgesten aus Bildern sind, wie dies am Beispiel des Sign Language MNIST-Datensatzes demonstriert wurde. Das finale Modell erreichte einen weighted F1-Score von 94.9%, was die Fähigkeit des Modells unterstreicht, verschiedene Buchstaben der amerikanischen Gebärdensprache (ASL) zuverlässig zu erkennen.

Die Experimentergebnisse verdeutlichen, dass durch den Einsatz von CNNs komplexe Muster in den Handgestenbildern erfasst werden können, wodurch eine hohe Klassifikationsgenauigkeit erzielt wird. Dies wurde durch systematisches Hyperparameter-Tuning erreicht, bei dem die optimalen Parameter für die Batchgröße, Lernrate und Epochenzahl ermittelt wurden.

Insgesamt zeigt diese Studie das Potenzial von CNNs zur automatischen Erkennung und Klassifikation von Gebärdensprache und leistet einen Beitrag zur Überwindung der Kommunikationsbarriere zwischen hörenden und gehörlosen Menschen. Weiterführende Arbeiten könnten die hier erzielten Ergebnisse erweitern, indem sie größere und vielfältigere Datensätze verwenden oder fortschrittlichere Deep-Learning-Techniken anwenden.

References

[1] o. V. Sign Language MNIST: Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks, 2024.

[2] Tianmei Guo, Jiwen Dong, Henjian Li, and Yunxing Gao. Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724, 2017.