# Klassifikation von Handgesten im Spiel Schere, Stein & Papier

Philipp Staufenberger (2025)
Duale Hochschule Baden-Württemberg
WWI2022F, Phlpp.staufenberger@gmail.com
Repository: https://github.com/Philippstf/RPS_MLP

***Abstract* — Dieses Paper beschreibt den Aufbau und die Evaluierung eines Convolutional Neural Networks zur Klassifikation von Handgesten im Spiel „Schere-Stein-Papier“. Ziel war es, anhand eines offenen Bilddatensatzes ein Modell zu entwickeln, das in der Lage ist, zwischen drei Gesten zu unterscheiden. Im Fokus stand eine methodisch saubere Umsetzung des CRISP-DM-Prozesses. Die Analyse zeigt, dass einfache CNN-Architekturen ausreichen um eine Klassifikationsaufgabe dieser Art zu lösen.**

***Keywords: Convolutional Neural Network; Bildklassifikation; Handgestenerkennung; CRISP-DM; Data Augmentation***

## I. Problem Understanding

Ziel dieses Projekts ist es, die Basisfunktionen eines leichten Hand-Gesten-Erkennungsmoduls zu demonstrieren, das in verschiedenen Human-Machine-Interfaces eingesetzt werden kann. Durch die Klassifikation elementarer Gesten (Schere, Stein, Papier) zeigen wir exemplarisch, wie sich ein effizientes Convolutional Neural Network (CNN) in touch-freien Steuerungssystemen, VR/AR-Anwendungen und Assistenzrobotern integrieren lässt. Ein erfolgreiches Modell könnte Handzeichen eines Spielers in Echtzeit erkennen und so beispielsweise ein klassisches „Schere-Stein-Papier“-Spiel gegen den Computer ermöglichen. Als primärer Erfolgsmaßstab dient die Klassifikationsgenauigkeit (Accuracy) auf unbekannten Testbildern. Da bei drei gleich häufigen Klassen eine naive Zufallsklassifikation nur ca. 33% Accuracy erreichen würde, sollte das CNN deutlich über diesem Zufallsniveau liegen. Weitere Gütemaße wie Präzision, Recall und die Konfusionsmatrix werden ergänzend betrachtet, um Fehlklassifikationen zwischen den Gesten sichtbar zu machen.

## II. Data Understanding

Als Datengrundlage wurde der öffentlich verfügbare Kaggle-Datensatz Rock-Paper-Scissors verwendet. Dieser umfasst insgesamt 2.188 Farbbilder von Handgesten in den drei Klassen rock (Stein), paper (Papier) und scissors (Schere) . Jedes Bild zeigt eine einzelne Hand, die eine der drei Gesten ausführt, vor unterschiedlichem Hintergrund. Die Bilder wurden ursprünglich in variabler Auflösung aufgenommen; für das Modelltraining wurden sie daher einheitlich auf ca. 300×200 Pixel skaliert (Breite×Höhe). Die Klassen sind annähernd ausgewogen vertreten, mit je ca. 700–750 Bildern pro Geste, sodass keine gravierende Datenungleicheit vorliegt.

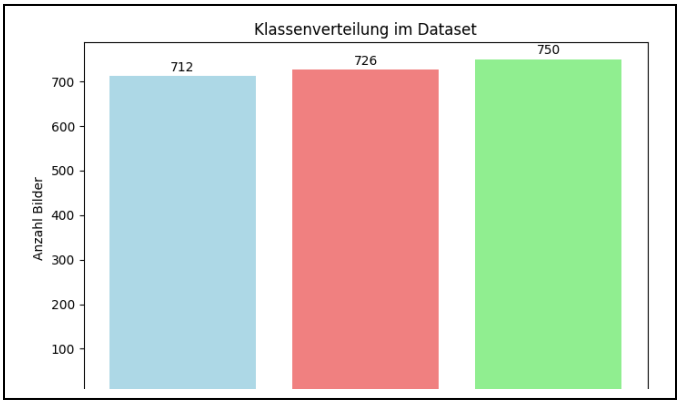


Figure 1. *Verteilung der 2.188 Bilder auf die drei Klassen. Jede Klasse ist gleich häufig vertreten, was eine Accuracy von ~33% als Zufalls-Baseline impliziert.*

Es lagen keine fehlerhaften oder unbrauchbaren Bilder vor. Eine manuelle Sichtung ergab, dass alle Dateien korrekt beschriftet und in passende Unterordner eingeteilt waren. Somit war keine aufwendige Datenbereinigung erforderlich. Wichtig für die Modellierung ist allerdings, dass trotz der klar unterscheidbaren Gesten (flache Hand für Papier vs. Faust für Stein vs. 2 gespreizte Finger für Schere) Variationen in Hautfarbe, Beleuchtung, Handposition und Sichtbarkeit der Hand die Klassifikation erschweren können.

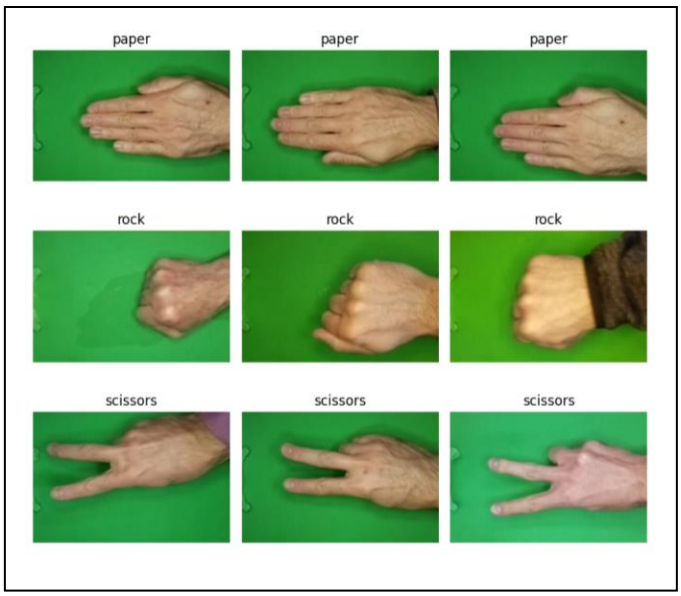


Figure 2. *Beispielbilder aus dem Datensatz für jede Klasse. Oben: Papier-Geste (flache Hand), Mitte: Stein-Geste (Faust), unten: Schere-Geste (zwei gespreizte Finger).*

Man erkennt, dass die Aufnahmen unterschiedliche Hintergründe, Hände und Lichtbedingungen aufweisen. Diese Variabilität stellt sicher, dass das Modell nicht einfach einen konstanten Hintergrund oder einzelne Personen „lernt", erhöht aber den Schwierigkeitsgrad der Erkennungsaufgabe. Der Datensatz enthielt keine vordefinierte Trennung in Trainings-/Testbilder. Diese Aufteilung wurde im Projekt selbst vorgenommen (siehe Datenvorbereitung). Dabei musste besonders auf eine saubere Trennung geachtet werden, um Datenlecks zu vermeiden (kein Bild darf gleichzeitig in Training und Test auftauchen).

## III. Data preparation

Für eine korrekte Validierung des Modells wurde der Datensatz in getrennte Teilmengen für Training, Validierung und Test aufgeteilt. Um sicherzustellen, dass kein Daten-Leck auftritt, erfolgte der Split dateibasiert und stratifiziert nach Klasse. Konkret wurde mit Keras' Utility-Funktion *image_dataset_from_directory* gearbeitet, welche den gesamten Datensatz (in den Ordnern paper, rock, scissors) zunächst in ein Trainingsset (hier 70%) und ein kombiniertes Validierungs/Test-Set (30%) aufteilte. Anschließend wurde das Validierungs/Test-Set zur Hälfte weiter unterteilt (jeweils 15%), sodass 1.532 Trainingsbilder, 320 Validierungsbilder und 336 Testbilder entstanden. Dieses Vorgehen stellt sicher, dass alle drei Splits ähnlich verteilt sind und keinerlei Überschneidungen enthalten. Das resultierende Trainingsset (ca. 70% der Daten) wurde ausschließlich zum Trainieren des CNN genutzt, das Validierungsset (15%) zur Hyperparameterabstimmung während des Trainings, und das unabhängige Testset (15%) erst für die finale Evaluierung herangezogen. Somit wurde gewährleistet, dass das Testset bis zum Schluss „unsichtbar" für das Modell blieb, was essenziell für eine objektive Messung der Generalisierungsleistung. Vor dem Training wurden mehrere Vorverarbeitungsschritte durchgeführt. Alle Bilder wurden auf eine einheitliche Größe von 300×200 Pixel gebracht und anschließend auf den 2 Wertebereich [0,1] normalisiert (Division durch 255). Diese Normalisierung ist entscheidend, um unterschiedliche Belichtungssituationen zu relativieren und numerische Stabilität beim Training zu gewährleisten. (Im vorigen Entwicklungszyklus war hier ein Fehler aufgetreten, da Bilder uneinheitlich normalisiert wurden, was in der finalen Pipeline korrigiert wurde.) Außerdem kam Datenaugmentation zum Einsatz, um die effektive Trainingsdatenmenge zu erhöhen und Overfitting vorzubeugen. Es wurde *on-the-fly* (während des Trainings in Echtzeit) bei jedem Trainingsbatch eine zufällige Transformation auf die Trainingsbilder angewandt, konkret: horizontales Spiegeln, Rotation bis ±10° und Zoom bis ±5%. Diese Augmentation mit moderater Intensität erhöht die Robustheit des Modells gegenüber verschiedenen Handorientierungen und Abständen zur Kamera. Die Validierungs- und Testbilder wurden nicht augmentiert, um deren Verteilung nicht zu verändern. Da die Klassen im Ausgangsdatensatz leicht unterschiedlich häufig waren (750 Bilder in der Spitze vs. 712 Bilder im minimum), wurden anfänglich Klassengewichte berechnet, um eine mögliche Bevorzugung der häufigsten Klasse zu vermeiden. In der Praxis zeigte sich jedoch, dass die minimale Unbalancierung keine nennenswerte Verzerrung verursachte. Das Modell lernte alle Klassen fast gleich gut (siehe Evaluierung), sodass die Gewichtung vernachlässigbar war. Abschließend wurde die Input-Pipeline mit Leistungsfeatures versehen: Die Daten wurden in Batches der Größe 32 geladen und soweit möglich zwischengespeichert und asynchron vorab geladen (Prefetching), um die GPU-Auslastung während des Trainings zu maximieren. Nach der Datenvorbereitung standen ein konsistentes Trainingsset, ein Validierungsset zur Hyperparameter-Tuning und ein völlig unabhängiges Testset bereit. Durch die strikte Trennung und identische Vorverarbeitung aller Splits ist sichergestellt, dass das Modell keinen unerlaubten Informationsvorsprung erhält und die finale Evaluation die echte Generalisierungsfähigkeit widerspiegelt.

## IV. Modelling

Als Modell wurde ein Convolutional Neural Network (CNN) entworfen, das üblich für Bildklassifikationsaufgaben wie diese ist. Die Architektur besteht aus drei aufeinanderfolgenden Faltungs-Pooling-Blöcken gefolgt von vollvernetzten Schichten. Im Einzelnen: Dem Eingabebild (200×300 Pixel mit 3 Farbkanälen) folgen drei 2D-Convolution-Layer mit 32, 64

bzw. 128 Filtern (Kernelgröße 3×3, ReLU-Aktivierung), jeweils gefolgt von einer 2×2 Max-Pooling-Schicht zur Dimensionsreduktion. Nach der dritten Faltung wird mittels Global Average Pooling auf einen Feature-Vektor der Länge 128 komprimiert. Darauf folgen eine Dropout-Schicht (Rate 50%) zur Regularisierung und schließlich zwei Dense-Layer: eine versteckte Schicht mit 64 Neuronen (ReLU) und der Output-Layer mit 3 Neuronen (Softmax) für die drei Klassen rock, paper, scissors. Diese schlanke Architektur umfasst insgesamt nur rund 101.700 trainierbare Parameter, was bewusst gewählt wurde, um trotz begrenzter Datenmenge eine gute Generalisierung zu ermöglichen. Auf eine mögliche Erweiterung durch Batch Normalization wurde verzichtet, da das Modell sich bereits als ausreichend leistungsfähig erwies, sodass zusätzliche Regularisierung nicht notwendig war. Trainiert wurde das CNN mit dem *Adam-Optimierer* (Lernrate 0,001) und *Sparse Categorical Crossentropy* als Verlustfunktion (die Ziel-Labels liegen als Integer-Klassenindices vor). Als Metrik zur Leistungsbeobachtung diente die Accuracy auf Trainings- und Validierungsdaten. Neben der Augmentation und der Klassenwichtung wurden weitere Maßnahmen gegen Overfitting ergriffen: Ein EarlyStopping-Mechanismus überwachte die Validierungsverlustfunktion ( val_loss ) und stoppte das Training, sobald über mehrere Epochen keine Verbesserung mehr eintrat. Parallel dazu speicherte *ModelCheckpoint* das beste Modell (basierend auf höchster Validierungs-Accuracy) unter dem Pfad models/rps_best.h5 zur späteren Verwendung in der Anwendung. Insgesamt verlief das Training sehr erfolgreich: Das Netzwerk lernte die Gestenerkennung innerhalb weniger Epochen nahezu perfekt.

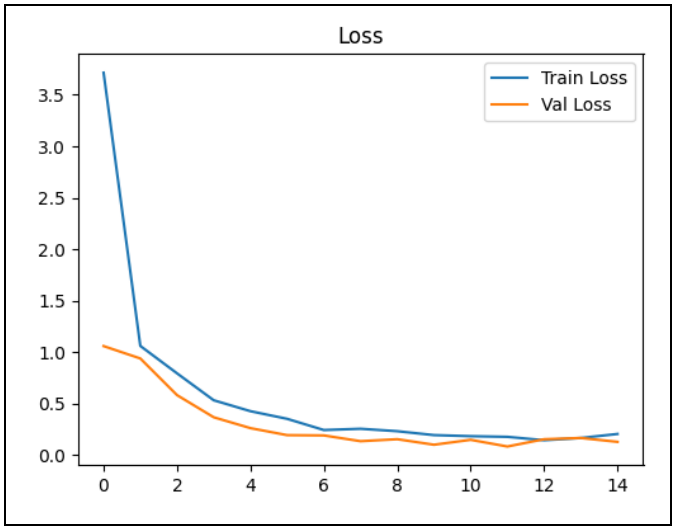


Figure 3. Loss-Werte des CNN

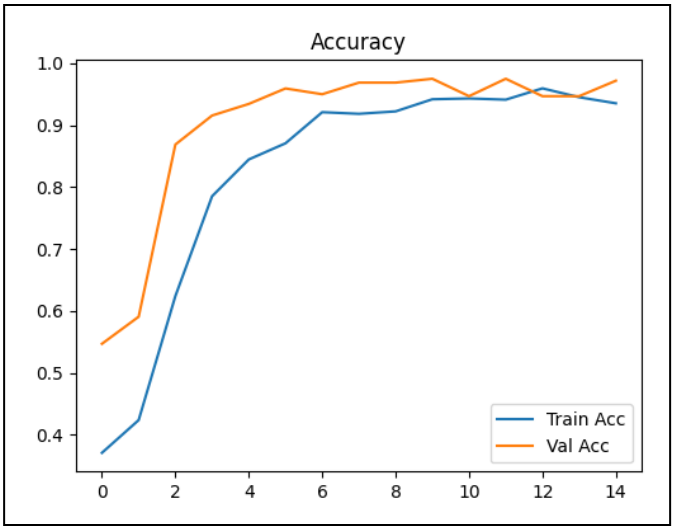


Figure 4. Accuracy-Werte des CNN

Man erkennt, dass bereits nach ~5 Epochen eine Accuracy >85% erreicht wird. Beide Kurven steigen kontinuierlich und konvergieren gegen Ende nahe 100%, ohne nennenswerte Divergenz zwischen Training und Validierung. Dieses Verhalten deutet darauf hin, dass weder starkes Overfitting noch Underfitting vorliegt. Das Modell besitzt genügend Kapazität, um die komplexen Merkmale zu lernen, bleibt aber durch Regularisierung und Augmentation generalisierungsfähig. Der finale Validierungs-Accuracy-Wert lag bei etwa 98–99%, was auf eine exzellente Modellleistung hindeutet.

## V. Evaluation

Die finale Bewertung erfolgte auf dem unabhängigen Testset mit 336 zuvor ungesehenen Bildern. Erwartungsgemäß bestätigte sich das im Training beobachtete Bild: Das trainierte CNN erreicht auf dem Testset eine Accuracy von 98,8%, was weit über dem Zufallsniveau von 33% liegt und das anfängliche Projektziel einer hohen Treffsicherheit deutlich übertrifft. Das Netzwerk erkennt die Klasse der Handgeste nahezu fehlerfrei. Wichtig ist dabei, dass die Testgenauigkeit konsistent mit der Validierungsgenauigkeit ist. Dies belegt, dass durch die gewählte saubere Pipeline keinerlei Datenüberschneidungen zwischen Trainings- und Testdaten stattgefunden haben und die gemessene Leistung wirklich die generalisierte Performance widerspiegelt. Ein detaillierter Blick auf die Konfusionsmatrix des Modells (Abb. 5) offenbart die Verteilung der Vorhersagen auf die tatsächlichen Klassen.

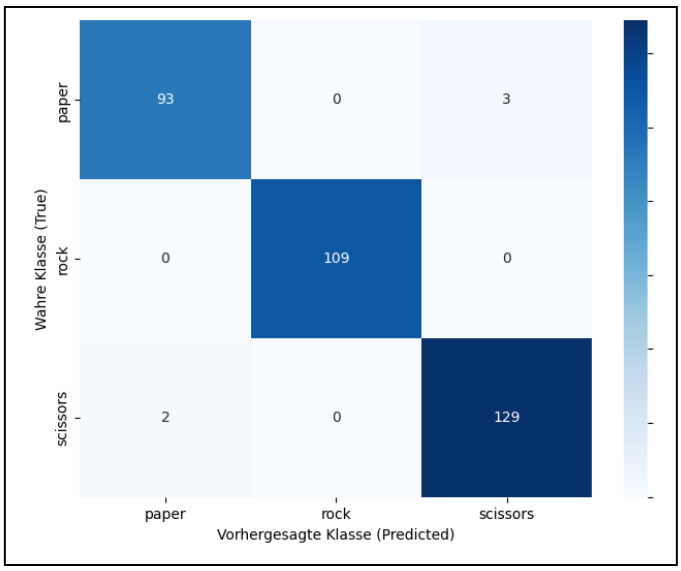


Figure 5. Konfusionsmatrix des finalen CNN auf dem Testset

Jede Zeile entspricht der wahren Klasse (True Label), jede Spalte der vom Modell vorhergesagten Klasse. Man sieht eine nahezu perfekte Zuordnung entlang der Diagonalen: Von 109 tatsächlichen rock-Bildern wurden alle 109 korrekt als rock erkannt. Bei paper wurden 93 von 96 Bildern richtig klassifiziert (3 irrtümlich als scissors erkannt), und bei scissors erkannte das Modell 129 von 131 Bildern korrekt (2 fälschlich als paper klassifiziert).

Kein einziges Bild wurde fälschlich der Klasse rock zugeordnet, was zeigt, dass das CNN die Faust-Geste eindeutig von den offenen Handgesten unterscheiden kann. Die wenigen Fehler treten ausschließlich zwischen paper und scissors auf: Diese Verwechslungen sind verständlich, da beide Gesten eine ausgestreckte Hand zeigen (fünf Finger vs. zwei Finger) und sich in manchen Aufnahmesituationen ähneln. Dennoch sind die Fehlerquoten gering, paper und scissors erreichen jeweils Precision- und Recall-Werte um 98–99%, und für rock beträgt sowohl Precision als auch Recall sogar 100%. Die Makro-Avg über alle Klassen liegt bei 0,98–0,99, was die ausgezeichnete Modellgüte unterstreicht. Insgesamt können alle drei Gesten mit hoher Sicherheit erkannt werden. Das robuste CNN-Modell erfüllt damit die Anforderungen für den praktischen Einsatz: Im Spielkontext würden die wenigen Fehlklassifikationen das Ergebnis nur in Ausnahmefällen beeinflussen.

## VI. Deployment

Abschließend wurde das trainierte Modell in eine Streamlit-Webanwendung integriert, um die Funktion in einer realistischen Benutzerschnittstelle zu demonstrieren. Die Anwendung ermöglicht es, ein beliebiges Handgesten-Bild hochzuladen und vom CNN klassifizieren zu lassen. Im Hintergrund wird dazu das beste trainierte Modell *(rps_best.h5*) geladen und auf das Eingabebild angewendet. Damit die Vorhersage konsistent mit dem Training erfolgt, wird das hochgeladene Bild zunächst vorverarbeitet: Es wird mit PIL auf 300×200 Pixel skaliert, in ein numpy-Array konvertiert und genauso wie im Training normalisiert (Division durch 255.0). Anschließend prognostiziert das Modell die Wahrscheinlichkeiten für jede der drei Klassen. Die App gibt das erkannte Symbol zusammen mit einem Konfidenzwert in Prozent aus, der angibt, wie sicher sich das Modell bei seiner Entscheidung ist. Zusätzlich wird je nach Konfidenzbereich eine kurze textuelle Rückmeldung angezeigt (z. B. „Ich bin mir sehr sicher!“ bei >90% oder „Ich bin mir nicht ganz sicher…“ bei niedrigeren Werten). Der Nutzer kann nach einer Vorhersage ein neues Bild laden und das Modell erneut testen. Die Benutzeroberfläche wurde ansprechend gestaltet (inkl. Sidebar mit Anleitung) und das Modul läuft in Echtzeit, da das CNN mit ~100 k Parametern sehr klein und performant ist. Dieses Deployment demonstriert, wie das entwickelte Gestenerkennungs-Modell praktisch eingesetzt werden kann, z.B. als Bestandteil einer interaktiven Anwendung, die kamerabasierte Handgesten als Eingabe für Spiele oder andere Human-ComputerInteraction-Szenarien nutzt.