Bericht: Optimierungen des CNN-Classifiers für den RoboCup-Wettbewerb

Gruppe 01: Jonas Jakob, Mischan Malek und Leon Wagner

Während der Übung haben wir (Gruppe 01) zahlreiche Optimierungen an unserem CNN-Classifier durchgeführt, um die Leistung des Modells bei der Objekterkennung zu steigern. In diesem Bericht werden die wichtigsten Änderungen besprochen und ihre Auswirkung beschrieben.

Datenaugmentation:

Wir haben verschiedene Techniken zur Datenaugmentation implementiert, um die Vielfalt unserer Trainingsdaten zu erhöhen und die Generalisierungsfähigkeit unseres Modells zu verbessern. Dazu gehören:

- Horizontales Spiegeln (RandomFlip)
- Zufällige Rotation (RandomRotation)
- Zufälliges Zoomen (RandomZoom)
- Zufällige Helligkeitsanpassung (RandomBrightness)
- Zufällige Kontrastanpassung (RandomContrast)

Diese Techniken sollten dem Modell mitteilen, dass es robust gegenüber kleinen Variationen der Eingangsdaten wird und die Gefahr des Overfittings reduziert.

Erweiterung der Netzwerkarchitektur:

Wir haben die Architektur unseres CNN erweitert, indem wir einen zusätzlichen Convolutional-BatchNorm-ReLU-Pool-Block hinzugefügt haben. Die Anzahl der Filter wurde schrittweise erhöht (32, 64, 128, 256), was dem Netzwerk ermöglicht, komplexere Merkmale zu erfassen.

Batch Normalization:

Nach jeder Convolutional-Schicht haben wir Batch Normalization eingeführt. Diese Technik normalisiert die Aktivierungen, was zu einer schnelleren Konvergenz während des Trainings und einer verbesserten Generalisierung führt.

Regularisierung:

Zur Verhinderung von Overfitting haben wir L2-Regularisierung in der Dense-Schicht implementiert. Zusätzlich wurde eine Dropout-Schicht mit einer Rate von 0.5 hinzugefügt, um die Abhängigkeit von einzelnen Neuronen zu reduzieren.

Optimierter Lernratenplan:

Wir haben einen Cosine Decay Lernratenplan implementiert, der die Lernrate im Laufe des Trainings anpasst. Dies ermöglicht eine effektivere Exploration des Parameterraums zu Beginn des Trainings und eine feinere Anpassung gegen Ende.

Gradient Clipping:

Um das Problem explodierender Gradienten zu vermeiden, haben wir Gradient Clipping mit einem Schwellenwert von 1.0 eingeführt.

Focal Loss:

Anstelle der herkömmlichen Categorical Crossentropy haben wir Focal Loss als Verlustfunktion implementiert. Focal Loss legt mehr Gewicht auf schwierig zu klassifizierende Beispiele, was besonders nützlich ist, wenn die Klassen unausgewogen sind.

Early Stopping:

Wir haben Early Stopping mit einer Geduld von 20 Epochen implementiert, um das Training zu beenden, wenn keine Verbesserung der Validierungsgenauigkeit mehr festgestellt wird. Dies hilft, Overfitting zu vermeiden und die Trainingszeit zu optimieren.

Ergebnisse:

Unsere Optimierungen führten zu sehr guten Ergebnissen sowohl in der lokalen Validierung als auch im Kaggle-Wettbewerb:

Lokale Validierungsergebnisse:

	precision	recall	f1-score	support
ball	0.98	0.99	0.99	146
field_line	0.99	0.99	0.99	161
penalty_mark	1.00	0.99	1.00	148
rest	0.98	0.97	0.97	169
robot_foot	0.98	0.99	0.99	176
accuracy		0.99	0.99	800
macro avg	0.99	0.99	0.99	800
weighted avg	0.99	0.99	0.99	800

Kaggle Ergebnisse:

Private Score: 0.927982

Public Score: **0.937526**