

CIFAR10 mit PyTorch klassifizieren

Neuronale Netze in der Bildverarbeitung

Jan Hoegen Nico Weber

27. Oktober 2025

Hochschule Karlsruhe
University of Applied Sciences

1. Allgemeines zu PyTorch
2. Eigenes Modell zu CIFAR 10
3. Training und Hyperparameter
4. Hyperparameter mit Bayesian Search

Allgemeines zu PyTorch

Was ist PyTorch?

- Python-Bibliothek für Deep Learning
- Stark verbreitet in Forschung und Lehre [1]
- Unterstützt dynamische Berechnungsgraphen („Define-by-Run“)

Warum PyTorch?

- Einfache und flexible Modellimplementierung
- Direkte Nutzung von GPU-Beschleunigung
- Große Community, viele Tutorials und Ressourcen

Autograd Berechnet Gradienten automatisch für Backpropagation

nn.Module Basis für selbstdefinierte Modelle, enthält vordefinierte Layers

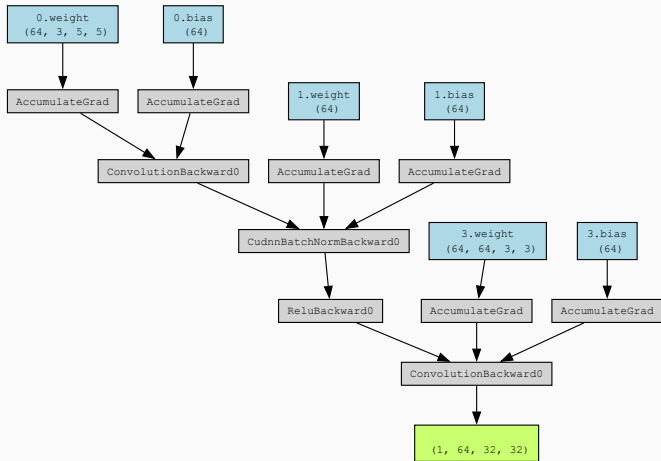
DataLoader Einfaches Laden, Batchen und **Parallelisieren** von Datensätzen

Optimizer Vorgefertigte Optimierer, z.B. **Adam**

Eigenes Modell zu CIFAR 10

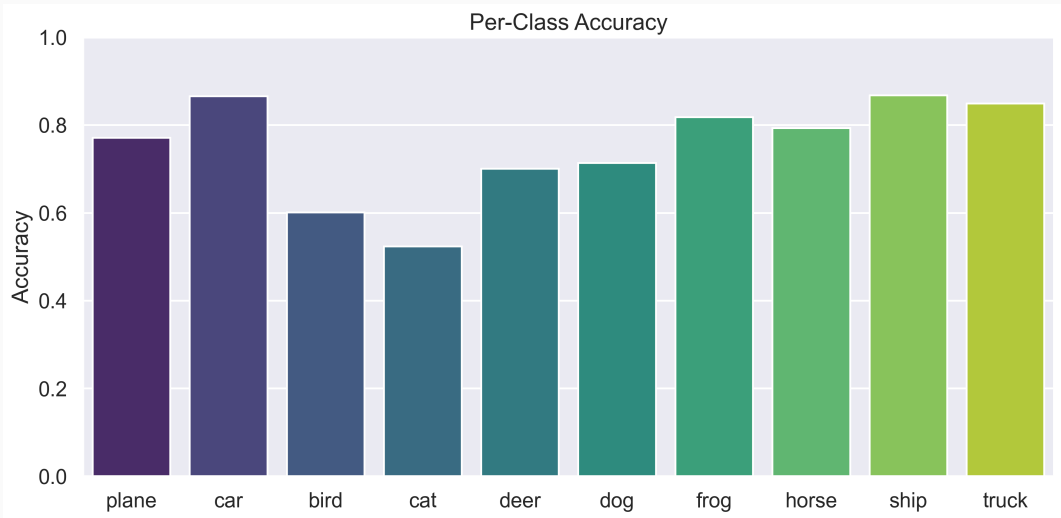
- **Aufgabe:** Klassifikation von CIFAR-10 Bildern
- **Datensatz:**
 - 10 Klassen
 - 60.000 Bilder, Größe $32 \times 32 \times 3$
 - Trainingsset: 49.000 Bilder
 - Validierungsset: 1.000 Bilder
 - Testset: 10.000 Bilder
- **Ziel:** Modell in *maximal 10 Epochen* trainieren, um Bilder korrekt zu klassifizieren

Architektur des Modells

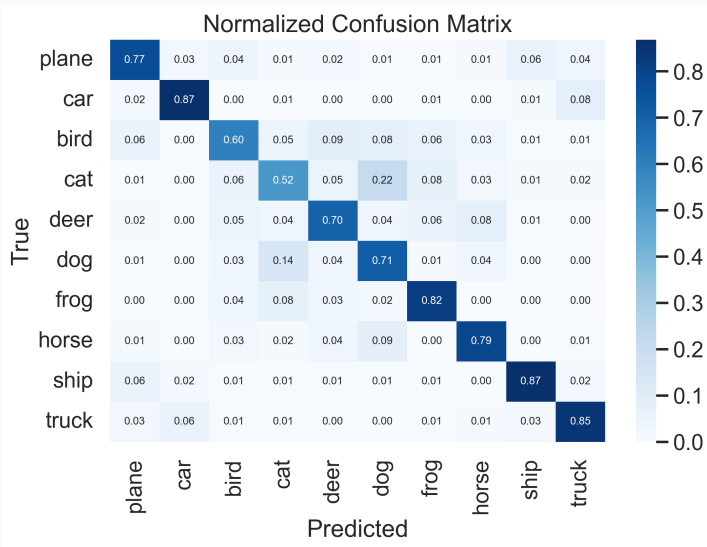


- Gewählte Hyperparameter:
 - $ch1 = 64$, $ch2 = 64$, $ch3 = 64$
 - Lernrate = 0.00086
 - Weight Decay = $3.81e-6$
- Best Validation Accuracy: 75.70%
- Trainings-Accuracy: 73%
- Test Accuracy: 75.05%

Accuracy für verschiedene Klassen

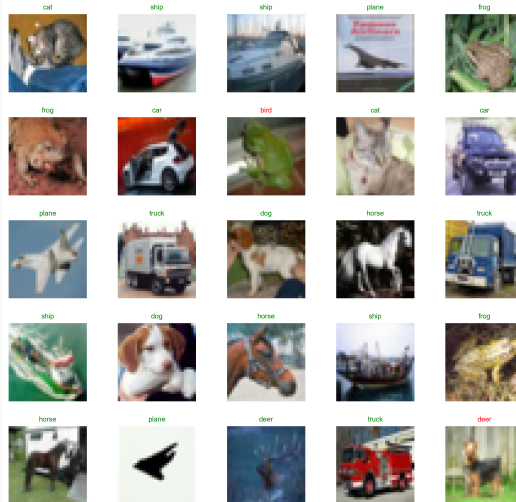


Confusion Matrix



Beispielvorhersagen

25 Model Predictions (green=correct, red=wrong)



Training und Hyperparameter

Einstellungen für das Training

Trainingsdaten:

- 49.000 Trainingsbilder, Größe $32 \times 32 \times 3$
- Batchgröße: 256
- DataLoader-Worker: 12 (hohe Parallelisierung)

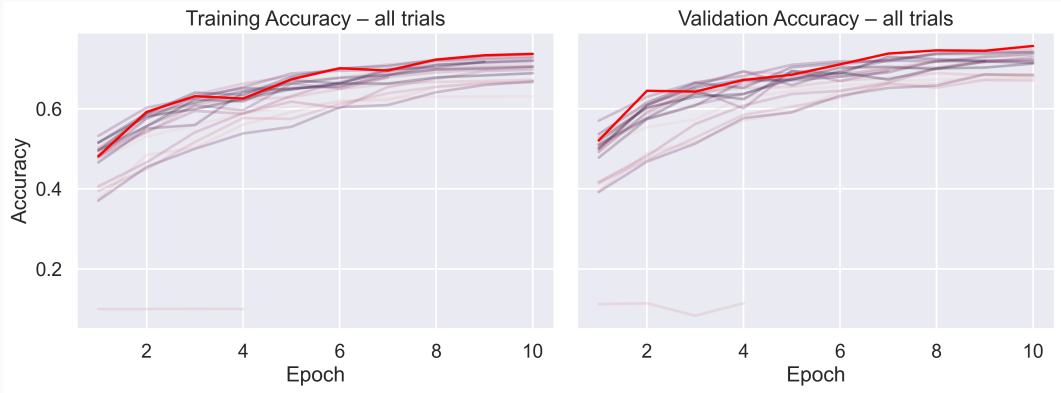
Training:

- Trainierbare Parameter: 604.810
- Iterationen pro Epoche: $\lceil 49,000/256 \rceil = 192$
- Epochen pro Trial: 10

Hyperparameter:

- Bayesian Search mit 20 Trials
- Optimierte Parameter: ch1, ch2, ch3, Lernrate, Weight Decay

genauigkeit während des Trainings



Early Stopping

Motivation:

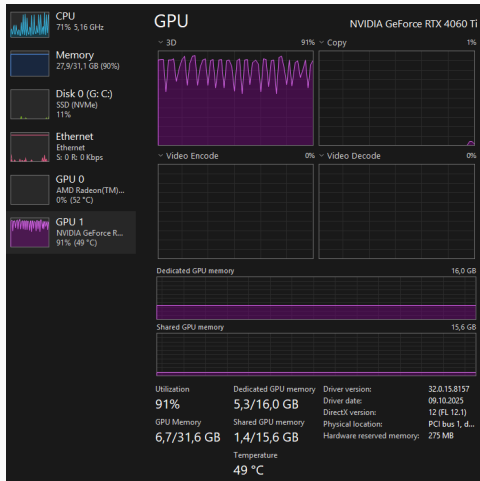
- Verhindert Overfitting
- Stoppt Training, wenn Validierungsgenauigkeit über mehrere Epochen nicht steigt

Prinzip:

- Wähle Grenzwert: `patience`
- Nach jeder Epoche: Behalte den besten Validierungswert
- Zähle Epochen ohne Verbesserung: `epochs_no_improvement`
- Stoppe, wenn `epochs_no_improvement \geq patience`

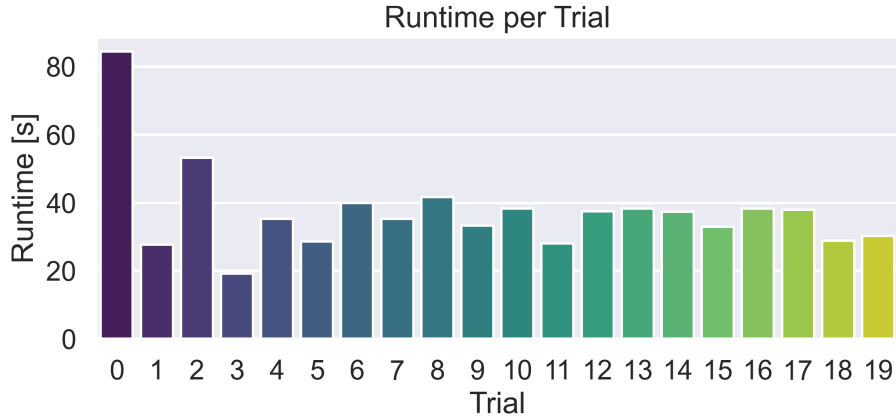
Hardware:

- GPU: RTX 4060 Ti, 16 GB VRAM
- CPU: Ryzen 5 7600X, 12 Kerne
- RAM: 32 GB DDR5



Laufzeiten pro Trial

Gesamtdauer: ca. 15 Minuten



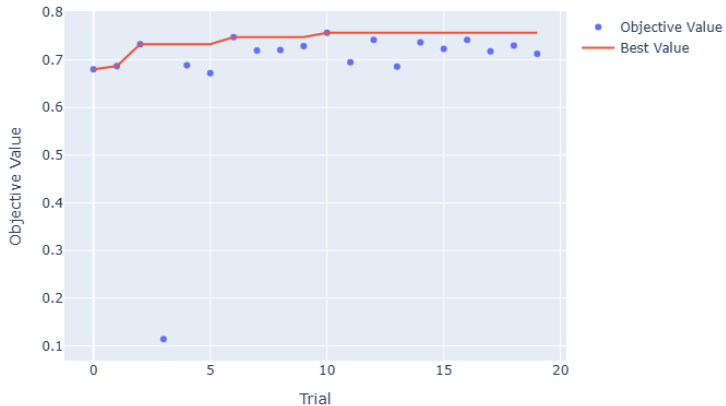
Hyperparameter mit Bayesian Search

Hier GIF einfügen.

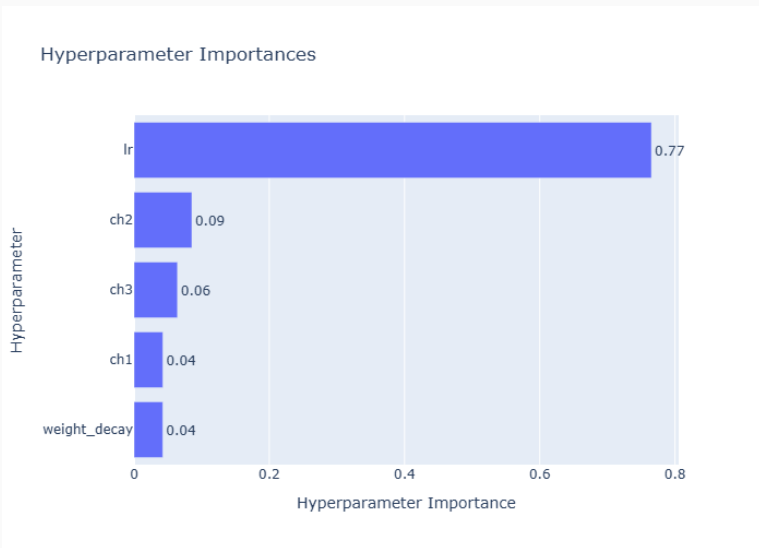
Wie wurde es auf unserer model angewendet?

Bayesian Search Ergebnisse

Optimization History Plot



Parameter des Bayesian Search



Fragen?

- [1] J. Bauer, *ComputerVision2: Neuronale Netze in der Bildverarbeitung*. 18. Juni 2025.
- [2] S. Subramanian, S. Juarez, C. Breviu, D. Soshnikov und A. Bornstein, „**PyTorch Tutorials: Beginner Basics**“, besucht am 26. Okt. 2025. Adresse:
<https://docs.pytorch.org/tutorials/beginner/basics/intro.html>