MNIST-Ziffernerkennung

Vergleich: Neural Network (NN) vs. Deep Neural Network (DNN)

Sheila Gómez López (81547) & Hatice Ulubas (74969)
MDP WS 2025/26
Modul: Machine Learning

Agenda

1. Kontext & Zielsetzung
2. Datenquellen: Cloud vs. Lokal
3. Datengrundlage & Datenumfang
4. Theoretische Grundlagen
5. Normalisierung
6. Modellarchitektur
7. Training
8. Analyse
9. Evaluation
10. Modellvergleich NN vs. DNN
11. Fazit
12. Quellen

Kontext & Zielsetzung

MNIST = „Modified National Institute of Standards and Technology“
Anwendungsfokus Optical Character Recognition (OCR) :

- Formularauswertung
- Postleitzahlen
- Dokumentendigitalisierung.

Modellvergleich & Kennzahlen-Interpretation für Nicht-Entwickler.

Datenquellen

Cloud

Google Storage (TensorFlow/Keras API) oder Github

```
Lade MNIST Datensatz...
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Trainingsdaten Form: (60000, 28, 28) (Bilder x Höhe x Breite)
Testdaten Form:      (10000, 28, 28)
```

Datenquellen

Lokal

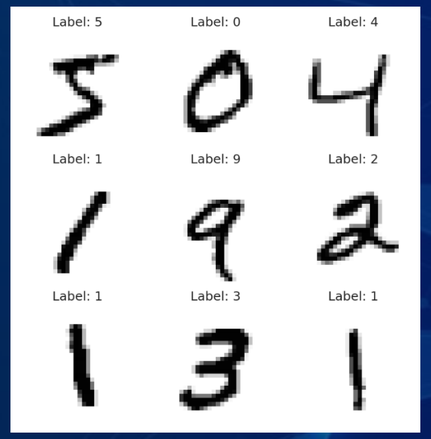
Manueller Import (ZIP/CSV)

```
zip_path = "/content/lokaler Datensatz.zip"  # Pfad zu deiner ZIP-Datei
extract_dir = "dataset"                      # Zielordner für extrahierte Dateien
```

```
Train Ordner: dataset/lokaler Datensatz/train
Test Ordner: dataset/lokaler Datensatz/test
```

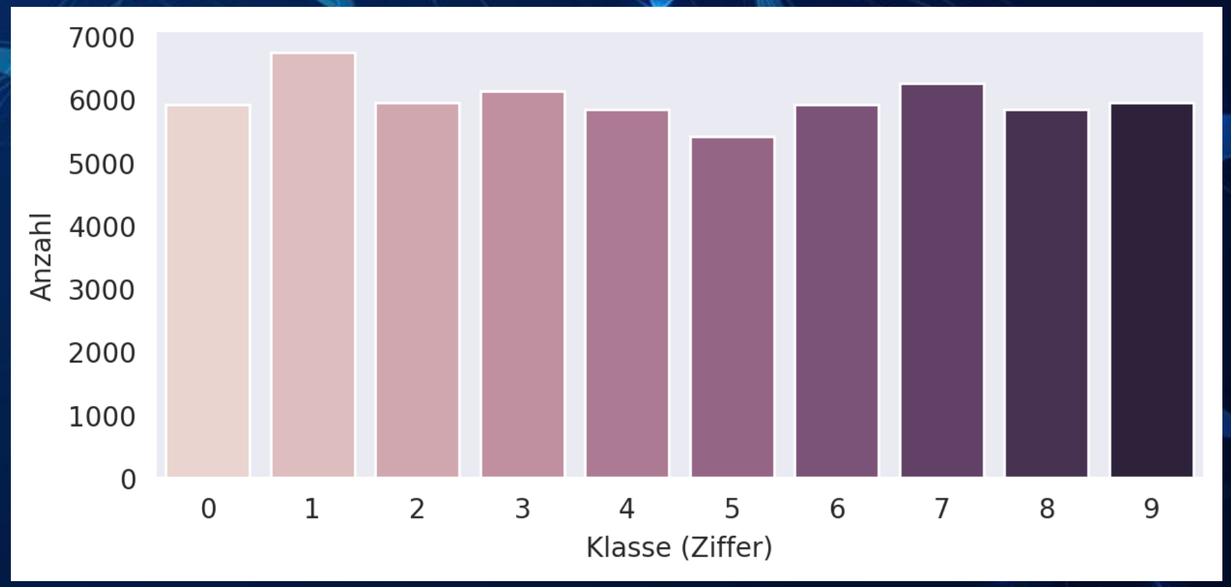
```
Train: (60000, 28, 28) (60000,)
Test : (10000, 28, 28) (10000,)
```

Datengrundlage & Datenumfang



DATENSPEZIFIKATION

Input: 28x28 Pixel (Graustufen)
Feature Vektor: 784 Dimensionen
Training Set: 60.000 Samples
Test Set: 10.000 Samples
Quelle: LeCun et al., 1998



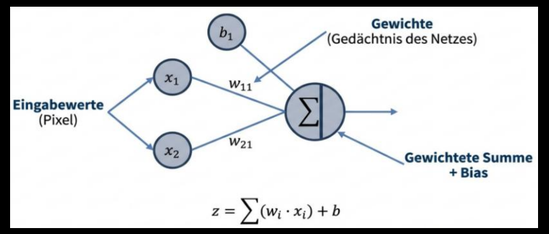
Theoretische Grundlagen

Grundgleichungen

Ein Neuron berechnet zuerst eine gewichtete Summe und wendet dann eine Aktivierungsfunktion an.

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$a = f(z)$$



Aktivierungsfunktionen (Formeln + Bedeutung)

Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

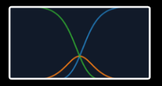
Wertebereich (0,1). Gut für binäre Ausgaben. Kann vanishing gradients verstärken.

ReLU

$$f(z) = \max(0, z)$$

Standard in Hidden-Layern. Schnell, reduziert vanishing gradients (aber „dead ReLU“ möglich).

Softmax



$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Output als Wahrscheinlichkeitsverteilung (Ziffern 0–9).

Loss-Funktion (Klassifikation)

Für Ziffernklassifikation wird meist Cross-Entropy verwendet.

$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$$

Optimizer: SGD vs. Adam

SGD

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

Einfach & gut erklärbar. Häufig mehr Tuning nötig (Learning Rate, Momentum).

Adam

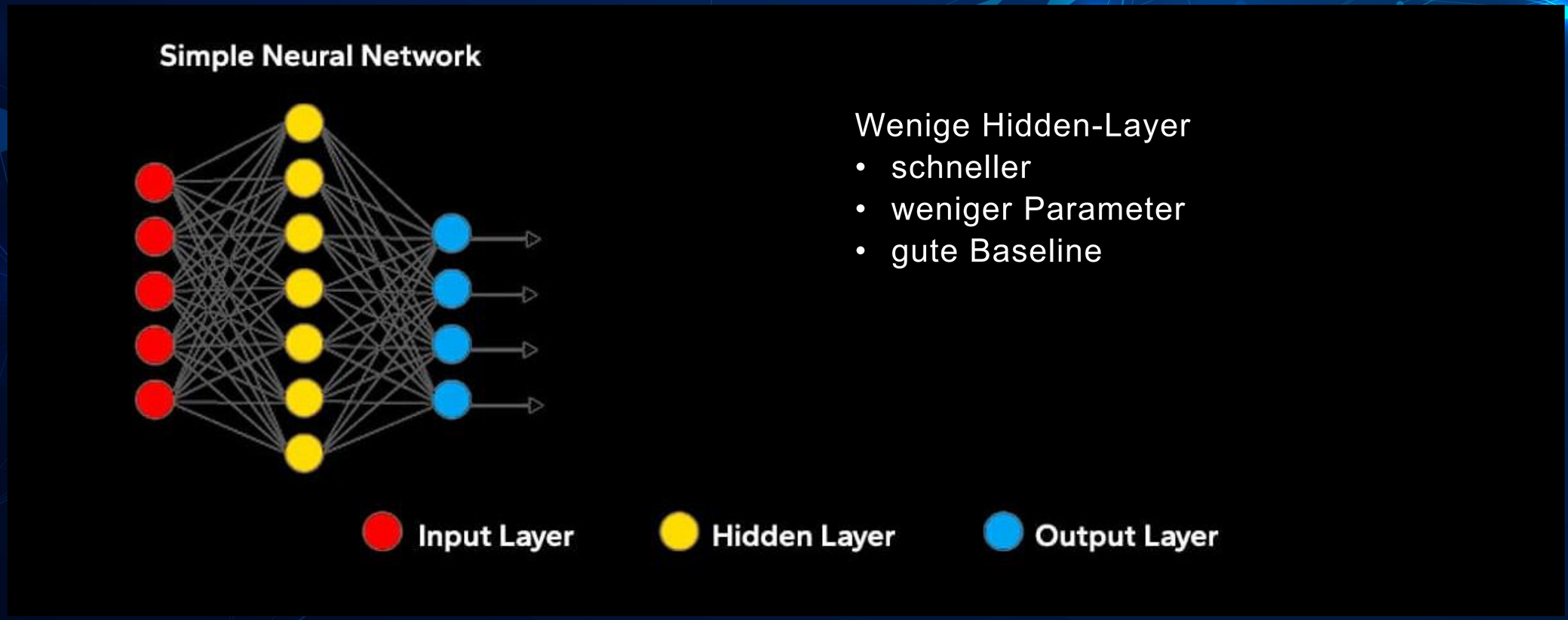
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Meist stabiler Start, oft schneller gute Ergebnisse → ideal für Vergleichsexperimente.

Theoretische Grundlagen



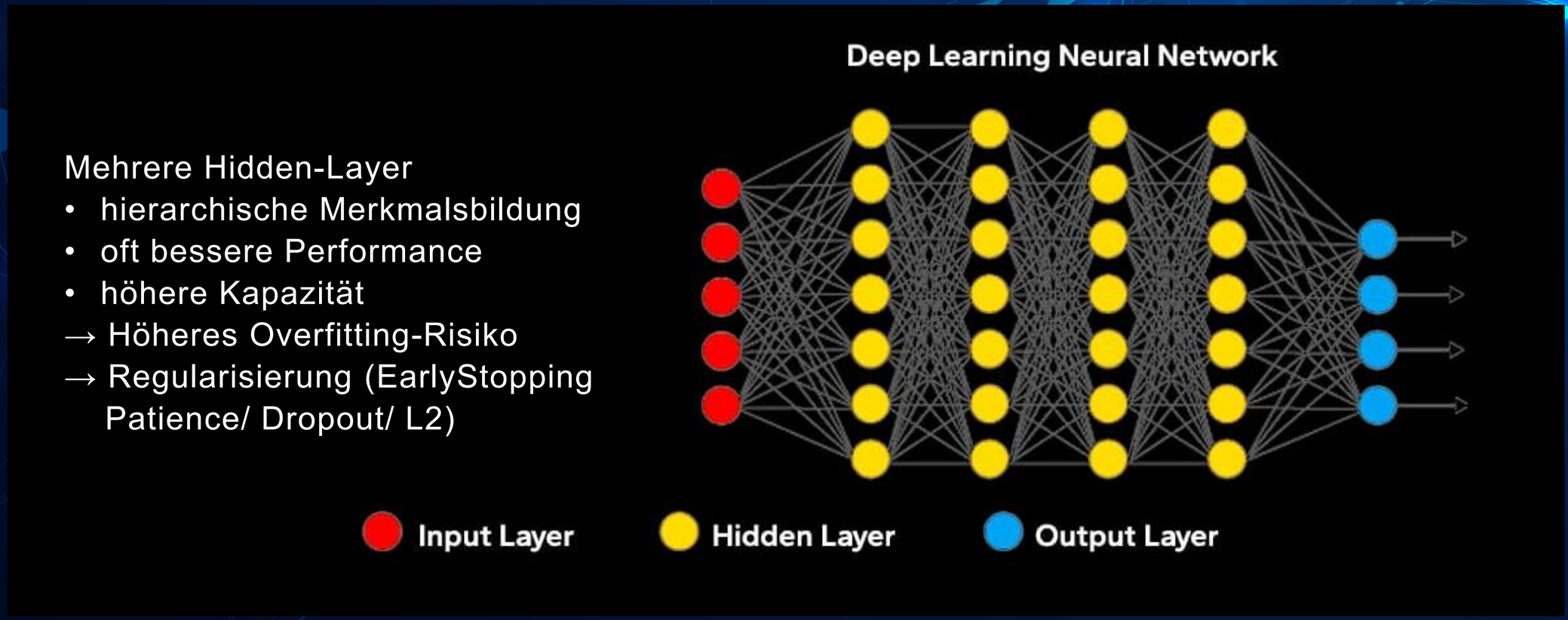
Wenige Hidden-Layer

- schneller
- weniger Parameter
- gute Baseline

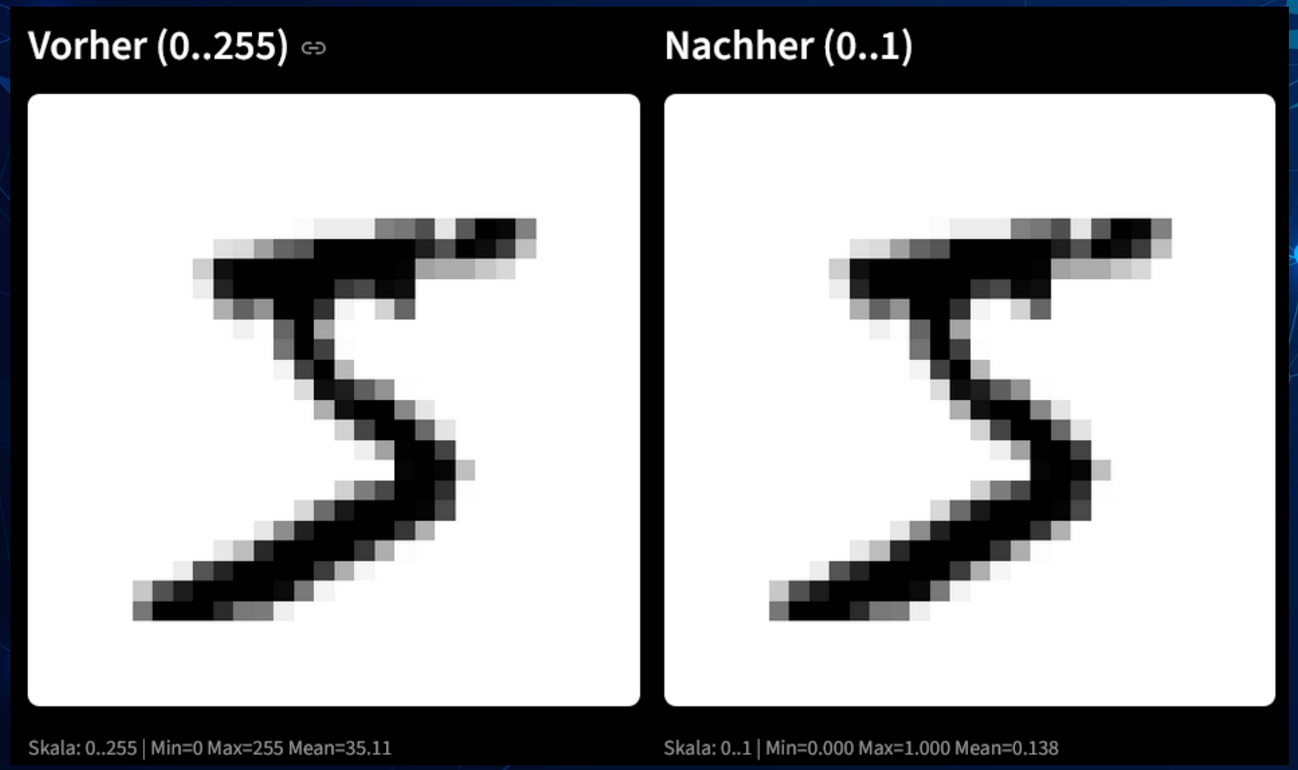
Theoretische Grundlagen

Mehrere Hidden-Layer
- hierarchische Merkmalsbildung
- oft bessere Performance
- höhere Kapazität

→ Höheres Overfitting-Risiko
→ Regularisierung (EarlyStopping Patience/ Dropout/ L2)

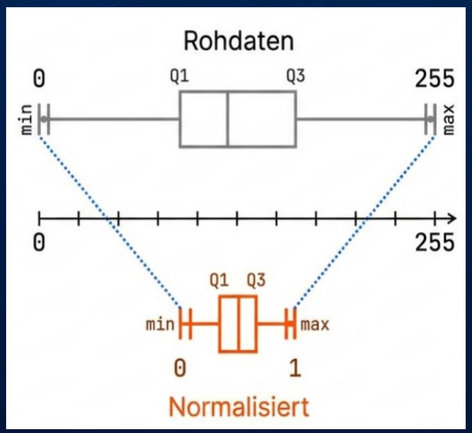


Normalisierung



Operation: `X / 255.0`
Typ: `Integer → Float32`

Inter JetBrains Mono
Operation: `X / 255.0`
Typ: `Integer → Float32`
Warum? Große Input-Werte destabilisieren den Gradienten.



Modellarchitektur

Simple NN (Baseline)

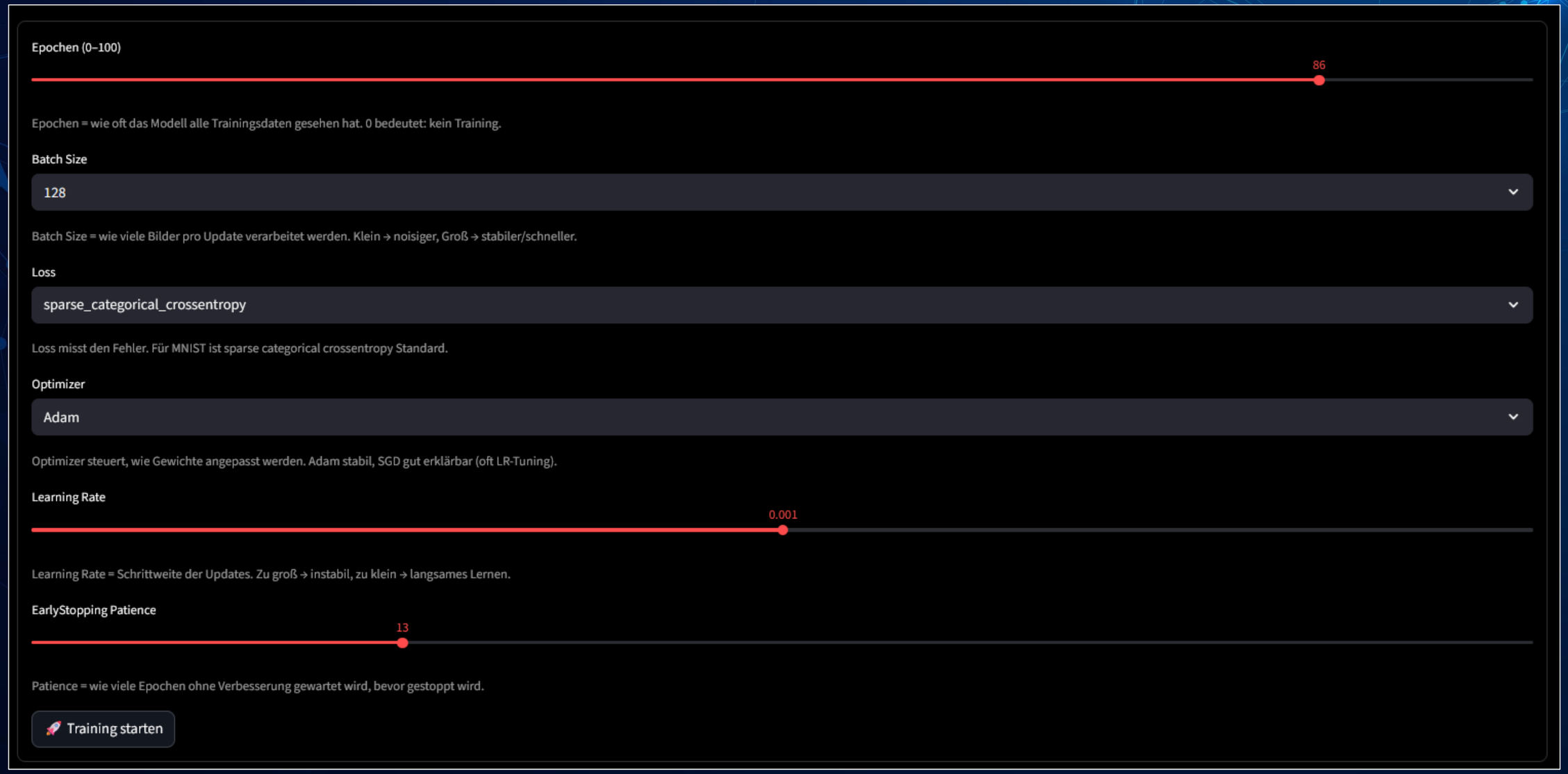
- Flatten
- Dense(128, ReLU)
- Dense(10, Softmax)

Deep NN (DNN)

- Flatten
- Dense(512, ReLU) → Dense(256, ReLU) → Dense(128, ReLU)
- Dense(10, Softmax)

- Flatten: wandelt 2D-Bildmatrix (28x28) in 1D-Vektor um
- Dense: Voll-verbundene Schichten; lernen Gewichtungen zwischen allen Neuronen)
- DNN hat deutlich mehr lernbare Parameter
  ->Höhere Accuracy erfordert aber stärkere Regularisierung (Overfitting-Schutz)

Hyperparameter & Early Stopping



Training

Simple NN Training

NN: Epoche 37/100 | acc=0.9998 val_acc=0.9772 | loss=0.0055
val_loss=0.0884 | ETA ~ 131.8s

NN Training fertig (inkl. EarlyStopping wenn ausgelernt).

Trainingsdauer (NN)

78.4 s

Tatsächlich trainierte Epochen (durch EarlyStopping): 37

Deep NN Training

DNN: Epoche 25/100 | acc=1.0000 val_acc=0.9829 | loss=0.0000
val_loss=0.1068 | ETA ~ 353.0s

DNN Training fertig (inkl. EarlyStopping wenn ausgelernt).

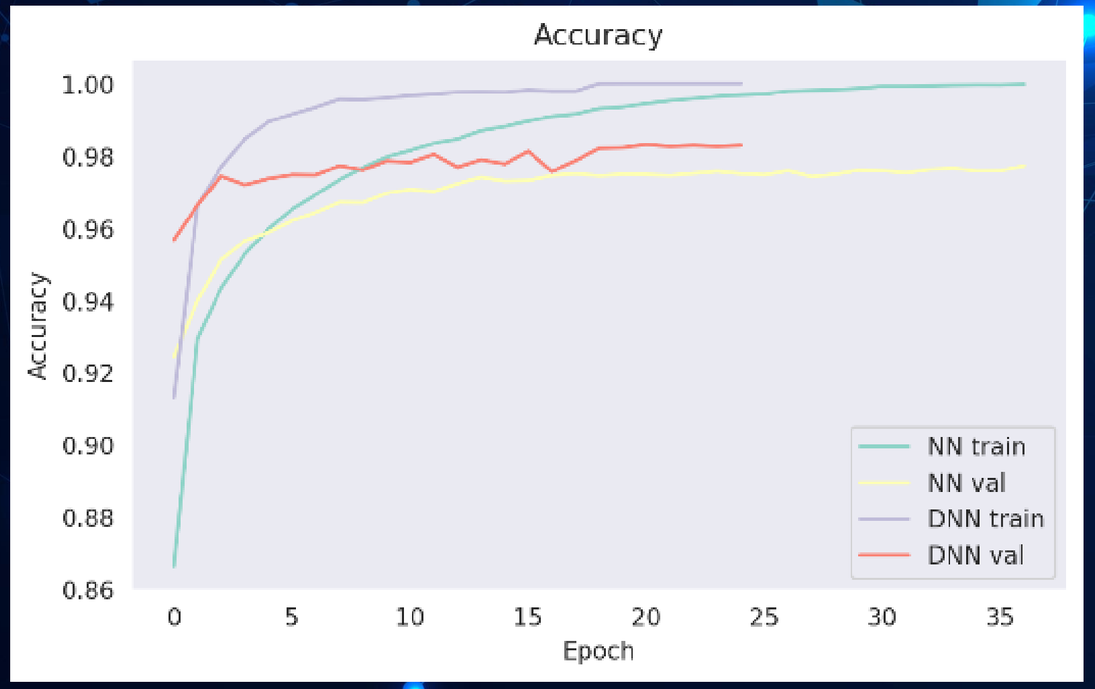
Trainingsdauer (DNN)

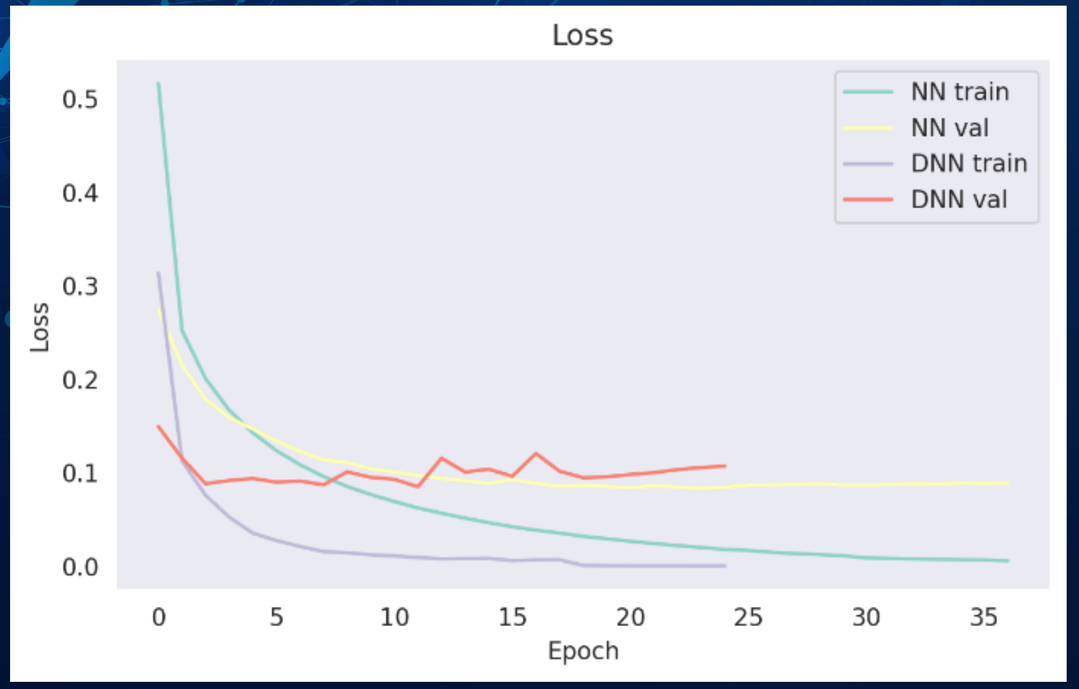
118.0 s

Tatsächlich trainierte Epochen (durch EarlyStopping): 25

Training

Lernkurven





Analyse

Analyse (NN): Stärken, Schwächen, Verbesserungen

- Hinweis auf Overfitting: Val-Loss deutlich höher als Train-Loss → EarlyStopping/Dropout/L2-Regularisierung nutzen.
- Schwächste Klassen (F1): 9 (F1=0.97), 7 (F1=0.97), 8 (F1=0.97) → Fokus auf diese Verwechslungen (mehr Training, LR-Tuning, ggf. DNN/Regularisierung).
- Häufigste Verwechslungen: 9→4 (12x), 4→9 (10x), 7→2 (10x) → Daten/Preprocessing prüfen, ggf. stärkere Architektur oder Regularisierung.

Fit-Diagnose (NN): **overfitting** | Gap(train-val acc)=0.0226

Train/Val driften auseinander (Gap=0.0226). Zusätzlich: val_loss_trend_up=True, train_loss_trend_down=True.

Analyse

Analyse (DNN): Stärken, Schwächen, Verbesserungen

- Hinweis auf Overfitting: Val-Loss deutlich höher als Train-Loss → EarlyStopping/Dropout/L2-Regularisierung nutzen.
- Schwächste Klassen (F1): 9 (F1=0.97), 7 (F1=0.98), 5 (F1=0.98) → Fokus auf diese Verwechslungen (mehr Training, LR-Tuning, ggf. DNN/Regularisierung).
- Häufigste Verwechslungen: 5→3 (9x), 7→9 (9x), 4→9 (8x) → Daten/Preprocessing prüfen, ggf. stärkere Architektur oder Regularisierung.

Fit-Diagnose (DNN): **optimal** | Gap(train-val acc)=0.0171

Train/Val nahe beieinander (Gap=0.0171), val_loss stabil.

Evaluation

Bedeutung der Kennzahlen

- **Accuracy:** Anteil korrekt klassifizierter Beispiele.
- **Precision:** Wie „sauber“ sind positive Vorhersagen? (wenig False Positives)
- **Recall:** Wie viele echte Beispiele wurden gefunden? (wenig False Negatives)
- **F1-Score:** Kompromiss aus Precision & Recall.
- **Support:** Anzahl Beispiele pro Klasse.
- **Confusion Matrix:** Zeigt, welche Ziffern verwechselt werden → Ansatzpunkt für Optimierung.

Evaluation

Ergebnisse: NN

Test Accuracy (NN)

97.56%

	precision	recall	f1-score	support
0	0.971	0.9898	0.9803	980
1	0.9903	0.9885	0.9894	1135
2	0.9775	0.969	0.9732	1032
3	0.9734	0.9782	0.9758	1010
4	0.9736	0.9766	0.9751	982
5	0.9764	0.9753	0.9759	892
6	0.9831	0.9687	0.9758	958
7	0.9644	0.9757	0.97	1028

Evaluation

Ergebnisse: DNN

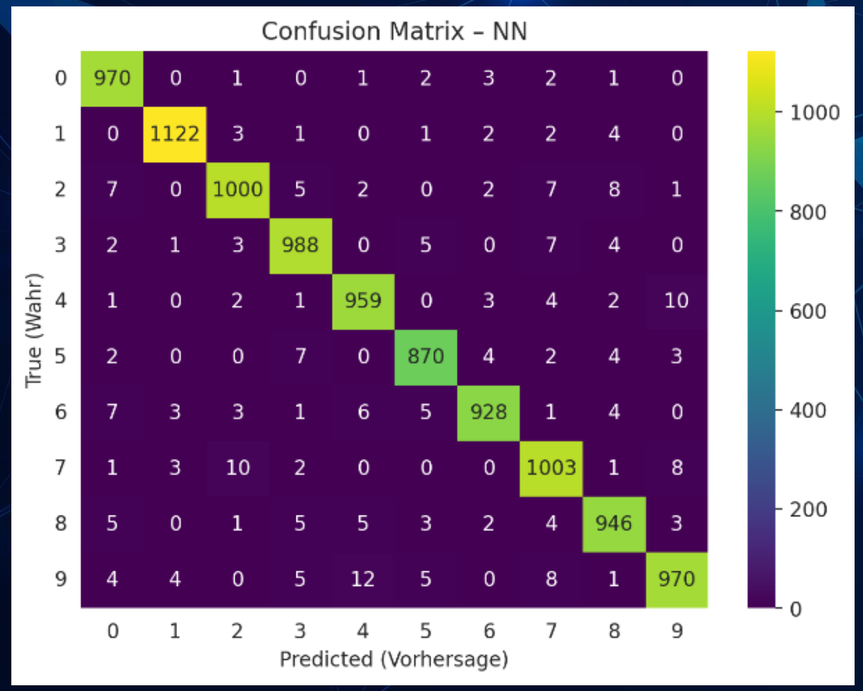
Test Accuracy (DNN)

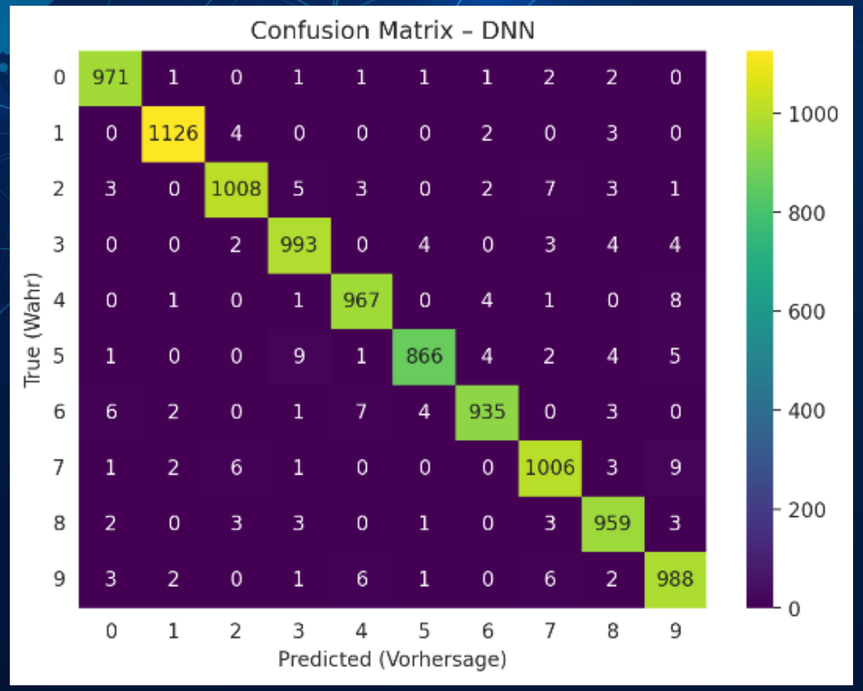
98.19%

	precision	recall	f1-score	support
0	0.9838	0.9908	0.9873	980
1	0.9929	0.9921	0.9925	1135
2	0.9853	0.9767	0.981	1032
3	0.9783	0.9832	0.9807	1010
4	0.9817	0.9847	0.9832	982
5	0.9875	0.9709	0.9791	892
6	0.9863	0.976	0.9811	958
7	0.9767	0.9786	0.9776	1028
8	0.9756	0.9846	0.9801	974
9	0.9705	0.9792	0.9748	1009

Evaluation

Confusion Matrix & Fehlklassifikationen





Modellvergleich NN vs. DNN

Bewertung & Vergleich (NN vs. DNN)

Accuracy-Differenz (DNN - NN): +0.63%

Sehr ähnlich → Hyperparameter/Regularisierung könnten entscheidend sein.

Welche Architektur ist besser geeignet (NN oder DNN)?

	Modell	Test Accuracy	Fit	Gap(train-val acc)	Parameter	Trainingszeit (s)	Epochen (tatsächlich)
0	NN	0.9756	overfitting	0.0226	101770	78.4109	37
1	DNN	0.9819	optimal	0.0171	567434	117.9508	25

Empfehlung: NN – Accuracy ist sehr ähnlich (ΔAcc=+0.63%). NN ist meist die bessere Wahl (weniger Parameter, schneller, leichter erklärbar).

Fazit

- Testreihe mit unterschiedlichen Parametern ergab immer Overfitting bei DNN
- Das einfache NN gewinnt durch Effizienz
  ->fast identische Accuracy bei deutlich weniger Ressourcen und Risiko

Künftige Optimierungen sollten eher auf Regularisierung setzen (z. B. Dropout, L2)

Quellen

- https://ieeexplore.ieee.org/abstract/document/10074302?casa_token=3YFf8744FvoAAAAA:Oy1n4MfcKdlqGY5gKRIoczGpIAigQnYFem3h6Bc_bFosoI93map8F_M_badQCij4Z-IiMKv0h9w1
- https://link.springer.com/chapter/10.1007/978-981-13-3441-2_11
- https://arxiv.org/abs/1805.06822
.
- https://link.springer.com/chapter/10.1007/978-981-13-1810-8_24
- https://link.springer.com/chapter/10.1007/978-3-662-68668-3_6
- https://link.springer.com/chapter/10.1007/978-3-658-25137-6_4

Vielen Dank!

Noch Fragen?