

Nutzung von KI in vta MC1

Im Rahmen der Mini Challenge „Gradient Descent“ wurde Künstliche Intelligenz – insbesondere ChatGPT – als unterstützendes Werkzeug eingesetzt. Ziel war es, den Entwicklungsprozess effizienter zu gestalten, Unsicherheiten bei der Formulierung und Modellierung zu klären und typische Stolpersteine schneller zu überwinden. Die Beispiele auf den Seiten 2 - 6 dokumentieren exemplarisch, wie KI in unterschiedlichen Phasen des Projekts hilfreich eingebunden wurde.

Reflexion zur Nutzung von KI

Die Nutzung von ChatGPT erwies sich in vielen Bereichen als äußerst hilfreich. Besonders bei der Formulierung von Kommentaren und Docstrings, der Strukturierung des Notebooks sowie der Interpretation von Fehlern und Fehlermeldungen konnte Zeit gespart und Klarheit gewonnen werden. Auch die Begutachtung der Projektstruktur im Hinblick auf die Bewertungskriterien wurde durch gezielte Rückfragen effizienter.

Gleichzeitig zeigte sich, dass der Einsatz von KI immer eine kritische Nachbearbeitung erfordert. Vorschläge zu mathematischen Herleitungen oder zur Auswahl von Evaluationsmetriken waren nicht immer korrekt oder vollständig. In solchen Fällen war es notwendig, die Inhalte durch eigene Recherchen zu validieren oder zu korrigieren. Auch stilistische Vorschläge mussten mit Blick auf Kohärenz und fachlichen Anspruch geprüft werden.

Insgesamt war ChatGPT ein wertvoller Assistent, insbesondere zur Unterstützung bei Routinetätigkeiten und zur Einordnung von Konzepten – jedoch kein Ersatz für eigenes Verständnis und kritisches Denken.

Formulierung von Kommentaren und Docstrings

KI wurde verwendet um effizient Kommentare und Docstrings zu verfassen oder bereits vorhandene Kommentare zu überprüfen.

Beispiel

Me

Bitte erstelle mir für die folgende Funktion einen korrekten Docstring:

```
def train_multiclass_nn(model, X, y, epochs, learning_rate, batch_size=64,
verbose: int = None): loss_history = [] acc_history = [] num_samples =
X.shape[0]
[...]
```

ChatGPT

Klar! Hier ist ein passender Docstring im NumPy-Stil für die Funktion `train_multiclass_nn`:

```
def train_multiclass_nn(model, X, y, epochs, learning_rate, batch_size=64,
verbose: int = None):
    """
    Führt das Training eines Mehrklassen-Klassifikationsnetzwerks mit
    Mini-Batch-Stochastic Gradient Descent durch.
    
```

Args:

```
    model (MultiLayerNN): Neuronales Netzwerk mit forward(),
    backward() und update()
    X (ndarray): Eingabedaten, Shape: [n_samples, input_dim]
    y (ndarray): One-Hot-kodierte Zielwerte, Shape: [n_samples,
    num_classes]
    epochs (int): Anzahl der Trainingsepochen
    learning_rate (float): Lernrate für die Parameteraktualisierung
[...]
```

Strukturierung des Dokuments

KI wurde verwendet, um bei der logischen Struktur des Dokuments zu helfen.

Beispiel

Me

Ist der Aufbau des Kapitels zu Aufgabe 3 gut oder soll ich gewisse Abschnitte weglassen?

Aufgabe 3 – Einfaches neuronales Netzwerk zur Klassifikation einer Ziffer

- Datenumwandlung: Torch-Dataset nach NumPy + binäre Zielvariable
- Ziel: Aufbau eines einfachen neuronalen Layers
- Aktivierungsfunktionen: ReLU und Sigmoid

[...]

ChatGPT

Dein Aufbau für **Aufgabe 3** ist insgesamt **sehr gut strukturiert und vollständig** – er deckt sowohl technische Umsetzung als auch konzeptuelle Entscheidungen ab. Dennoch gibt es ein paar **kleine Optimierungsvorschläge**, um **Redundanz zu vermeiden**, die **Leseführung zu verbessern** und den Fokus zu schärfen:

 **Optimierte Gliederung mit Kommentaren**

[...]

Beispielevaluation

Wir haben unser Dokument mit Hilfe der Bewertungskriterien kontrollieren lassen.

Beispiel

Me

Ich möchte Punkt für Punkt die Bewertungskriterien durchgehen.

Betrachte das html Dokument und gib Feedback zu Bewertungskriterien 1 - 12.

ChatGPT

Super, wir gehen jetzt **Punkt für Punkt** die Bewertungskriterien 1–12 durch und bewerten sie anhand des Inhalts des HTML-Dokuments vta_mc_1.html. Ich stütze mich dabei auch auf die Aufgabenstellung und die Bewertungskriterien-PDFs

Bewertungskriterien 1–3: Dataset

1) Daten korrekt mit torchvision geladen

→ **Ja.** Das Notebook verwendet torchvision.datasets.MNIST in Kombination mit transforms.ToTensor() und lädt sowohl Trainings- als auch Testdaten korrekt.

2) Visualisierungen der Daten sind gut verständlich und repräsentativ

→ **Ja.** Es werden 15 Beispielbilder pro Ziffer (0–9) dargestellt. Zusätzlich folgt eine Balkengrafik zur Verteilung der Labels – sauber beschriftet und optisch klar.

3) Grundlegende Eigenschaften des Datasets korrekt beschrieben

→ **Ja.** Die Anzahl der Datenpunkte, Shape der Bilder, Typen der Daten und Verteilung der Labels werden textlich erläutert und auch im Output des Codes sichtbar gemacht.

[...]

Für Debugging

Für schnelle Recherche von Fehlern oder Problemen im Code.

Beispiel

Me

```
-----  
- TypeError Traceback (most recent call last)  
Cell In[25], line 141  
138 return test_acc  
140 # Train and Evaluate  
--> 141 train(model, train_loader, optimizer, criterion, test_loader,  
epochs=20, patience=5)  
  
Cell In[25], line 96, in train(model, train_loader, optimizer, criterion,  
test_loader, epochs, patience)  
93 images, labels = images.to(device), labels.to(device)  
94 optimizer.zero_grad()  
--> 96 with torch.amp.GradScaler('cuda'):  
97 outputs = model(images)  
98 loss = criterion(outputs, labels) TypeError: 'GradScaler' object does  
not support the context manager protocol
```

ChatGPT

It looks like there's a **syntax mistake** in how **mixed precision (AMP)** is being used in your training loop. Specifically, the error occurs in:

```
with torch.amp.GradScaler('cuda'):
```

🔍 What's Wrong?

- `torch.amp.GradScaler()` is **not** a context manager (with statement is incorrect).
- Instead, you should **create a GradScaler instance** before training and **use it correctly** within the loop.

[...]

Überprüfung auf Rechtschreibfehler

Bedarf hoffentlich keiner Erklärung.