ERKLÄRBARE KÜNSTLICHE INTELLIGENZ

Vorlesung 5

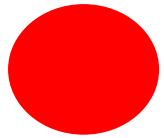
GradCam

- Grad-CAM (Gradient-weighted Class Activation Mapping) ist eine Methode zur Visualisierung und Erklärung von neuronalen Netzwerken, insbesondere von Convolutional Neural Networks (CNNs).
- Es hilft dabei, die Bereiche eines Bildes zu identifizieren, die für die Vorhersage einer bestimmten Klasse wichtig sind.
- Grad-CAM erzeugt eine "Heatmap", die zeigt, welche Bildregionen das Modell bei seiner Entscheidung berücksichtigt hat.

GradCam

- **Funktionsweise von Grad-CAM**
- **Feature Maps extrahieren**
  - Grad-CAM konzentriert sich auf die letzten **Convolutional Layer** in einem CNN, weil diese Schichten hochgradig abstrahierte Merkmale (Features) eines Bildes erfassen. Jede **Feature Map** in dieser Schicht entspricht einer bestimmten Eigenschaft, die das CNN erkannt hat, z. B. Kanten, Formen oder Texturen.
- **Gradienten berechnen**
  - Um die Bedeutung jeder **Feature Map** zu bestimmen, berechnet Grad-CAM die Gradienten des Klassenausgabe-Scores (für die Zielklasse) in Bezug auf die Aktivierungen der Feature Maps:
  - Die Gradienten geben an, wie empfindlich der Klassenausgabe-Score auf Änderungen in einer bestimmten Feature Map reagiert.

GradCam



- **3. Gewichte berechnen**
  - Die Gradienten werden verwendet, um Gewichtungen für jede Feature Map zu berechnen.
- **4. Heatmap erstellen**
  - Die gewichteten Feature Maps werden kombiniert, um eine **Klassen-spezifische Heatmap** zu erstellen:
- **5. Heatmap auf das Eingabebild projizieren**
  - Die resultierende Heatmap wird auf die Größe des Eingabebildes hochskaliert und auf dieses überlagert. So entsteht eine intuitive Visualisierung, die zeigt, welche Bildregionen für die Vorhersage entscheidend waren.

GradCam

- Vorteile von Grad-CAM
  - Kompatibilität: Funktioniert mit beliebigen CNN-basierten Modellen (z. B. ResNet, VGG).
  - Klassenabhängig: Liefert spezifische Visualisierungen für jede Zielklasse.Einfache Integration: Nutzt vorhandene Gradienten und Feature Maps, ohne Änderungen am Modell.
  - EinschränkungenBegrenzte Präzision: Liefert weniger feingranulare Ergebnisse im Vergleich zu pixelbasierten Methoden.
  - Fokussierung auf Convolutional Layers: Funktioniert nicht für vollständig vernetzte Schichten oder nicht-konvolutionale Architekturen.

Übung Grad CAM

- Andere Bilder testen: Tausche das Bild gegen ein anderes aus.
- Andere Schichten analysieren: Ändere die Zielschicht (layer4) und beobachte die Unterschiede.
  - for name, module in model.named_modules(): print(name) → Gibt die schichten aus
  - Durch die Änderung der Zielschicht kannst du untersuchen, welche Informationen in den verschiedenen Schichten enthalten sind. Niedrigere Schichten zeigen allgemeinere Merkmale, während höhere Schichten spezifischere Informationen enthalten.
    - layer1: Niedrigere Merkmale wie Kanten und Texturen
    - Layer2: layer3: Mittlere Merkmale wie Formen.
    - Layer4: Höhere Merkmale wie Objekte.

Übung Grad CAM

- Grad-CAM für mehrere Klassen
  - Ziel: Visualisiere die relevanten Bildbereiche für unterschiedliche Klassen und vergleiche die Heatmaps.
  - Schritte: Wähle zwei Klassen (z. B. "German Shepherd" und "Labrador Retriever").
  - Erstelle Grad-CAM-Heatmaps für beide Klassen.
  - Vergleiche, welche Bildbereiche für jede Klasse wichtig sind.
  - Diskussionsfragen:
    - Überlappen sich die relevanten Bildbereiche der Klassen?
    - Warum oder warum nicht?Wie könnte Grad-CAM helfen, wenn ein Bild mehreren Klassen zugeordnet werden kann?
  - Code-Änderung: Erzwinge die Klasse für die Grad-CAM-Berechnung: (s. Nächste Folie)

Übung Grad CAM

```
class_indices = [207, 208]  # Class IDs for German Shepherd and Labrador Retriever

for class_idx in class_indices:
    print(f"Class Index: {class_idx}")
    heatmap = grad_cam(model, input_tensor, target_layer="layer4", class_idx=class_idx)
    overlayed_image = overlay_heatmap(original_image, heatmap)

    plt.figure(figsize=(10, 5))
    plt.suptitle(f"Results for Class {class_idx}")

    plt.subplot(1, 2, 1)
    plt.title("Grad-CAM Heatmap")
    plt.imshow(heatmap, cmap='jet')
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.title("Overlayed Image")
    plt.imshow(overlayed_image)
    plt.axis("off")

    plt.show()
```

Übung Grad CAM

- Verwendung des Bildes aus Saliency Map
- Vergleiche Grad-CAM mit einer anderen Erklärungsmethode (z. B. Saliency Maps).

Counterfactual Explanations im Bildbereich

- Eine **Counterfactual Explanation** (kontrafaktische Erklärung) im Bildbereich ist eine Methode, um zu zeigen, welche Änderungen an einem Bild erforderlich wären, damit ein Modell seine Klassifikation zu einer anderen, gewünschten Klasse ändert. Es ist ein Ansatz der Explainable AI (XAI), der hilft, die Entscheidungsprozesse eines Modells besser zu verstehen, indem er aufzeigt, wie das Bild verändert werden müsste, um eine alternative Vorhersage zu erhalten.
- Stellen wir uns vor, ein neuronales Netzwerk klassifiziert ein Bild als „Katze“. Eine kontrafaktische Erklärung würde nun ein ähnliches Bild erzeugen, das so modifiziert ist, dass das Modell es als eine andere, vorgegebene Zielklasse (z.B. „Hund“) klassifiziert. Dieses modifizierte Bild wird als **kontrafaktisches Bild** bezeichnet.
- **Vorgehensweise bei der Erstellung kontrafaktischer Bilder**

1. **Modellvorhersage und Zielklassendefinition:**
    1. Zuerst wird die aktuelle Vorhersage des Modells für das Originalbild ermittelt.
    2. Eine alternative Zielklasse wird festgelegt, z. B. „Hund“ anstelle der ursprünglichen Vorhersage „Katze“.

Counterfactual Explanations im Bildbereich

- **Vorgehensweise bei der Erstellung kontrafaktischer Bilder**

2. **Optimierung zur Erzeugung des kontrafaktischen Bildes:**
    1. Das Originalbild wird als Eingabe verwendet und in kleinen Schritten verändert, bis das Modell es als die Zielklasse erkennt.
    2. Die Optimierung minimiert die „Distanz“ zur Zielklasse und erzeugt eine Bildversion, die möglichst nah am Original ist, jedoch für das Modell so verändert wurde, dass es die Zielklasse statt der Originalklasse erkennt.
3. **Visualisierung der Änderungen:**
    1. Das kontrafaktische Bild zeigt, welche Merkmale das Modell als typisch für die Zielklasse betrachtet. Es zeigt also, welche Eigenschaften im Bild angepasst wurden, um die Klassifikation zu ändern.

Counterfactual Explanations im Bildbereich

- **Aufgabe: Kontrafaktische Erklärung – Bild einer Katze in einen Hund umwandeln**
  - In dieser Übung werden Sie ein Bild einer Katze mithilfe eines vortrainierten Modells und TensorFlow so modifizieren, dass das Modell das Bild letztendlich als Hund klassifiziert. Hierbei werden Sie schrittweise Änderungen am Bild vornehmen und dabei die kleinsten Anpassungen finden, die das Modell benötigen könnte, um die Klassifikation von „Katze“ zu „Hund“ zu ändern.
- **Ziel der Aufgabe**
  - Das Ziel ist, zu verstehen, wie man mit kontrafaktischen Erklärungen arbeitet, indem man das Bild durch Optimierung so verändert, dass die Bildinhalte für das Modell eher einem Hund entsprechen. Dabei nutzen Sie das VGG16-Modell, das auf dem ImageNet-Datensatz vortrainiert ist.

Übung zu Counterfactual Explanations im Bildbereich

- Führen Sie den Code aus und beobachten was passiert.
- Schafft es der Algorithmus das Bild zu ändern?
  - Falls, ja auf welche Klasse?

Vision Large Language Modelle

- Das **CLIP-Modell** (Contrastive Language–Image Pretraining) ist ein von OpenAI entwickeltes, leistungsstarkes Modell, das **Texte** und **Bilder** miteinander verknüpfen kann.
- Es wurde speziell darauf trainiert, Beziehungen zwischen Bildern und deren textuellen Beschreibungen zu verstehen, und stellt damit eine bedeutende Innovation im Bereich von Vision-Language-Modellen dar.

Vision Large Language Modelle

- **Wie funktioniert CLIP?**

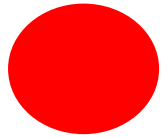
1. **Architektur:**
    1. Besteht aus zwei separaten Modellen:
        1. Ein **Bildencoder** (z. B. ein ViT – Vision Transformer), der Bilder verarbeitet.
        2. Ein **Textencoder** (z. B. ein Transformer), der Texte verarbeitet.
    2. Beide Encoder projizieren ihre Eingaben in denselben Embedding-Space.
2. **Trainingsdaten:**
    1. CLIP wurde mit **400 Millionen Bild-Text-Paaren** aus dem Internet trainiert.
    2. Diese riesige Menge an Daten ermöglicht es dem Modell, allgemeines Wissen über Sprache und Bilder zu lernen.
3. **Training:**
    1. Während des Trainings werden die Embeddings von passenden Bild-Text-Paaren so angepasst, dass sie nahe beieinander im Embedding-Space liegen.
    2. Unpassende Paare werden so verändert, dass ihre Embeddings weit voneinander entfernt liegen.

Vision Large Language Modelle

- **Wofür wird CLIP verwendet?**

1. **Bildklassifikation:**
    1. CLIP kann Bilder klassifizieren, indem es die Ähnlichkeit zwischen einem Bild und einer Liste von Textbeschreibungen berechnet.
    2. Beispiel: Ein Bild wird einer Kategorie wie "Hund" oder "Katze" zugeordnet.
2. **Zero-Shot-Learning:**
    1. CLIP kann neue Aufgaben ohne zusätzliche Feinabstimmung bewältigen, da es während des Trainings allgemeines Wissen gelernt hat.
3. **Text-Bild-Suche:**
    1. Suche nach Bildern basierend auf Textbeschreibungen oder umgekehrt (z. B. "Finde Bilder von Hunden").
4. **Kreative Anwendungen:**
    1. Kann in Kunst und Design genutzt werden, z. B. zur Generierung oder Filterung von Bildern basierend auf Beschreibungen.

Erklärbarkeit bei Vision Large Language Modelle



- Die **Visualisierung der Token-Wichtigkeit** zeigt, wie relevant jedes einzelne Token (Wort oder Teil eines Wortes) im Text für die Vorhersage des Modells ist. Diese Relevanz basiert auf den **Aufmerksamkeitswerten** (Attention Scores), die vom CLIP-Modell berechnet werden.
- Was ist Token-Wichtigkeit? Token: Ein Token ist ein einzelnes Wort oder ein Teil eines Wortes (bei Subword-Tokenization).
- Beispiel: Der Satz "a cute dog in a park" könnte in Tokens zerlegt werden wie:
- ['a', 'cute', 'dog', 'in', 'a', 'park’]
- Wichtigkeit: Zeigt, wie stark das Modell bei der Berechnung der Textrepräsentation auf ein bestimmtes Token "achtet".
- Tokens, die besonders relevant sind, wie "cute" oder "dog", erhalten einen höheren Relevanzwert.Weniger wichtige Tokens, wie Artikel oder Präpositionen ("a", "in"), erhalten niedrigere Relevanzwerte.

Bedeutung des Attention Scores

1. **Relevanz eines Tokens:**
    1. Ein höherer Attention Score zeigt an, dass das Modell diesem Token mehr Bedeutung beigemessen hat, weil es für die Vorhersage der Ähnlichkeit zwischen Bild und Text entscheidend ist.
    2. Zum Beispiel könnte das Wort "*carrot*" bei der Beschreibung eines Bildes mit einer Karotte einen hohen Score erhalten, weil es direkt relevant ist.
2. **Kontextabhängigkeit:**
    1. Transformer-Modelle wie CLIP arbeiten mit Selbstaufmerksamkeit, d. h., sie gewichten Tokens basierend auf ihrem Kontext.
    2. Ein Token kann einen hohen Attention Score erhalten, wenn es in Bezug auf das Bild besonders informativ oder einzigartig ist.
3. **Interpretierbarkeit:**
    1. Die Scores helfen, das Verhalten des Modells zu erklären:
        1. Warum wurde ein bestimmtes Bild als passend oder unpassend zum Text eingestuft?
        2. Welche Wörter beeinflussen die Entscheidung des Modells am meisten?

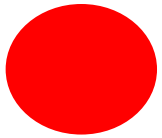
Übung VLLM

- Aufgabe
  - Schritte: Lade ein anderes Bild von einer URL (z. B. eine Katze oder eine Landschaft).
  - Ändere die Beschreibung (text) entsprechend.
- Führe den Code aus und notiere die Ähnlichkeit.
- Wie passt der Attention Score zusammen?
- Fragen:
  - Wie verändert sich die Ähnlichkeit, wenn der Text präziser oder ungenauer wird?
  - Was passiert, wenn der Text etwas beschreibt, das nicht im Bild ist?

Übung VLLM

- Kombiniere mehrere Beschreibungen
- Ziel: Teste, wie CLIP mit mehreren Textbeschreibungen arbeitet.
- Schreibe verschiedene Beschreibungen:
  - Beschreibung 1: "a cute dog in a park„
  - Beschreibung 2: "a large German Shepherd„
  - Beschreibung 3: "a small kitten on a chair„
  - Berechne die Ähnlichkeit für jede Beschreibung mit demselben Bild.
  - Fragen:
    - Welche Beschreibung erzielt die höchste Ähnlichkeit?
    - Warum? Wie beeinflussen die Unterschiede in den Beschreibungen die Relevanz der Tokens?

Partial Dependence Plots



- Ein **Partial Dependence Plot (PDP)** ist ein Visualisierungswerkzeug, das in der erklärbaren künstlichen Intelligenz (XAI) verwendet wird.
- Es zeigt, wie eine oder mehrere Eingabevariablen die Vorhersagen eines maschinellen Lernmodells beeinflussen, indem der mittlere Effekt dieser Variablen auf die Zielgröße dargestellt wird.
- Der PDP gibt an:
  - Wie sich die Zielvariable (Vorhersage) im Durchschnitt ändert, wenn eine Eingabevariable variiert wird, während alle anderen Variablen konstant gehalten werden.
  - Dies ermöglicht es, die Beziehung zwischen einer Eingabevariable und der Zielvorhersage zu interpretieren, auch in komplexen Modellen wie Entscheidungsbäumen, Random Forests oder neuronalen Netzen.

Partial Dependence Plots

- **Funktionsweise**
  1. **Fixierung anderer Variablen:**
     - Für jede Instanz im Datensatz werden die Werte der zu untersuchenden Eingabevariablen geändert, während alle anderen Eingabevariablen konstant gehalten werden.
  2. **Durchschnittliche Vorhersage berechnen:**
     - Für jeden Wert der interessierenden Variable wird die durchschnittliche Modellvorhersage berechnet.
  3. **Plot erstellen:**
     - Die Ergebnisse werden als Linie (bei einer Variablen) oder als 3D-Oberfläche (bei zwei Variablen) dargestellt

Übung PDP

- Führen Sie das Notebook aus
- Aufgabe 1:
  - Erstelle einen Partial Dependence Plot für ein anderes Feature aus dem Boston Housing Dataset, z. B. LSTAT (Anteil der Bevölkerung mit niedrigem sozialen Status).
  - feature_to_analyze = "LSTAT"
  - Interpretiere die Form des Plots.

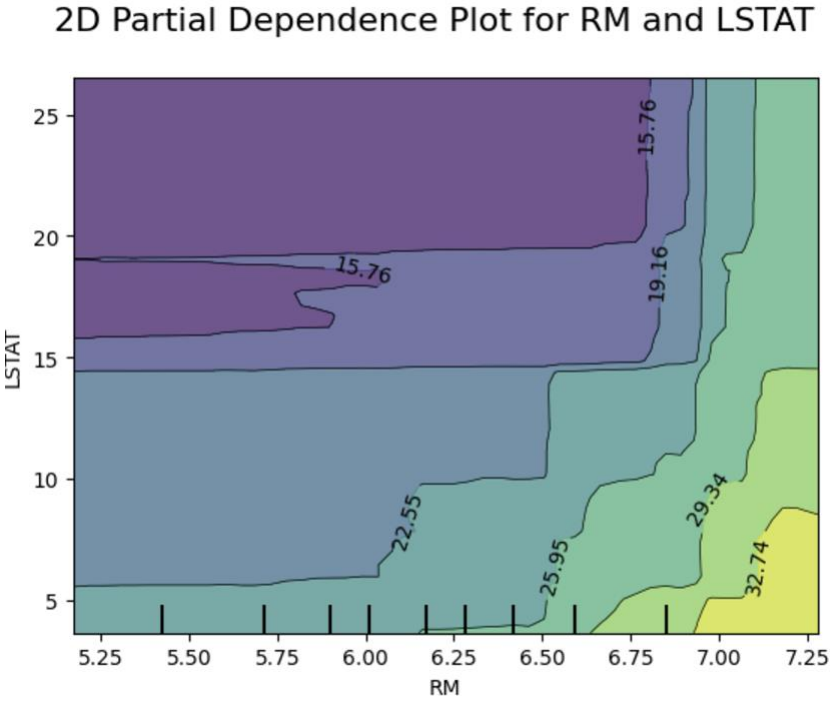
Übung PDP

- Aufgabe 2: Generiere einen 2D-Partial Dependence Plot für die Interaktion zwischen RM (Anzahl der Zimmer) und LSTAT (Bevölkerungsstatus).
- Untersuche, wie die Interaktion zwischen den beiden Features die Vorhersagen beeinflusst.
- Code:
- # Generiere 2D Partial Dependence Plot für "RM" und "LSTAT"

```
PartialDependenceDisplay.from_estimator(
model, X_test, [("RM", "LSTAT")], feature_names=X_test.columns, grid_resolution=50)
# Titel hinzufügen
plt.suptitle("2D Partial Dependence Plot for RM and LSTAT", fontsize=16)
plt.show()
```

Übung PDP

- Regionen im Plot und ihre Bedeutung:
- 1. Hohe Vorhersagen (Gelb/Grün):Ort: Rechts unten, bei hohen RM (über 7) und niedrigen LSTAT (unter 10).
- Interpretation: Große Häuser in Gegenden mit wohlhabender Bevölkerung erzielen die höchsten Hauspreise. Der Preis liegt hier über 32.74 (Isolinie).
- Grund: In wohlhabenden Gegenden treiben große Häuser die Nachfrage und somit die Preise in die Höhe.
- 2. Mittlere Vorhersagen (Blau/Grün):Ort: Mittlerer Bereich, RM zwischen 6 und 7 und LSTAT zwischen 10 und 20.
- Interpretation: Der Preis ist moderat (zwischen 22.55 und 29.34) und wird durch eine Kombination aus mittelgroßen Häusern und einem durchschnittlichen sozialen Status beeinflusst.
- Grund: Diese Gegenden repräsentieren Mittelklassebereiche, in denen die Preise ausgeglichener sind.3.



Übung PDP

- Niedrige Vorhersagen (Violett/Blau):Ort: Links oben, bei niedrigen RM (unter 6) und hohen LSTAT (über 20).
- Interpretation: Kleine Häuser in ärmeren Gegenden erzielen die niedrigsten Hauspreise, unterhalb von 15.76.
- Grund: Eine ärmere Bevölkerung und kleinere Häuser führen zu geringerer Nachfrage und niedrigeren Preisen.

Übung PDP

- Aufgabe 3: Vergleich von PDPs bei verschiedenen Modellen
- Aufgabe: Trainiere zwei Modelle (z. B. Random Forest und Gradient Boosting).
- Erstelle PDPs für ein Feature (z. B. RM) aus beiden Modellen und vergleiche sie.