Hochschule Düsseldorf

Fakultät für Medien

Studiengang: B. Sc. Data Science, AI und Intelligente Systeme (DAISY)

Modul: Big Data Engineering

Dozent: Prof. Dr. Florian Huber

Sommersemester 2024

Image Recommender

Vorgelegt von: Jonah Gräfe und Anas Alhariri

E-Mail: jonah.graefe@study.hs-duesseldorf.de
anas.alhariri@study.hs-duesseldorf.de

Abgabetermin: 18.08.2024

Inhaltsverzeichnis

Teil 1 – Image recommender software

1. Motivation/Ziel des Projekts

Das Ziel dieses Projekts ist es, eine optimierte Python-Software zur Bildempfehlung zu entwickeln. Diese Software soll in der Lage sein, basierend auf einem Eingabebild ähnliche Bilder aus einem großen Datensatz zu identifizieren und zu empfehlen. Die Ähnlichkeit wird dabei durch verschiedene Metriken bestimmt, darunter Farbschemata und Embeddings.

2. Programm Design

Das Programm besteht aus mehreren Hauptkomponenten, die in enger Wechselwirkung stehen (siehe Abb. 1). Zu den zentralen Elementen gehören ein Generator zum Laden der Bilder und eine Datenbank, die die Metadaten jedes einzelnen Bildes enthält und zur Verknüpfung der Bild-IDs mit ihren Speicherorten dient.

Die Berechnung der Ähnlichkeiten erfolgt basierend auf verschiedenen Metriken:

1. Eine Metrik basiert auf dem Farbprofil bzw. der Farbähnlichkeit.
2. Eine andere Metrik verwendet Embeddings, die mithilfe eines vortrainierten Inception v3 Modells aus der PyTorch-Bibliothek erzeugt werden.
3. Eine weitere Metrik nutzt die Objekterkennung mit dem YOLO-Modell (You Only Look Once), um erkannte Objekte und deren Positionen in den Bildern zu vergleichen. Diese Objekterkennung ermöglicht eine detaillierte Analyse der Bildinhalte, indem sie Klassen und Begrenzungsrahmen (Bounding Boxes) extrahiert und die Ähnlichkeiten zwischen Bildern anhand der Übereinstimmung dieser Objekte berechnet.

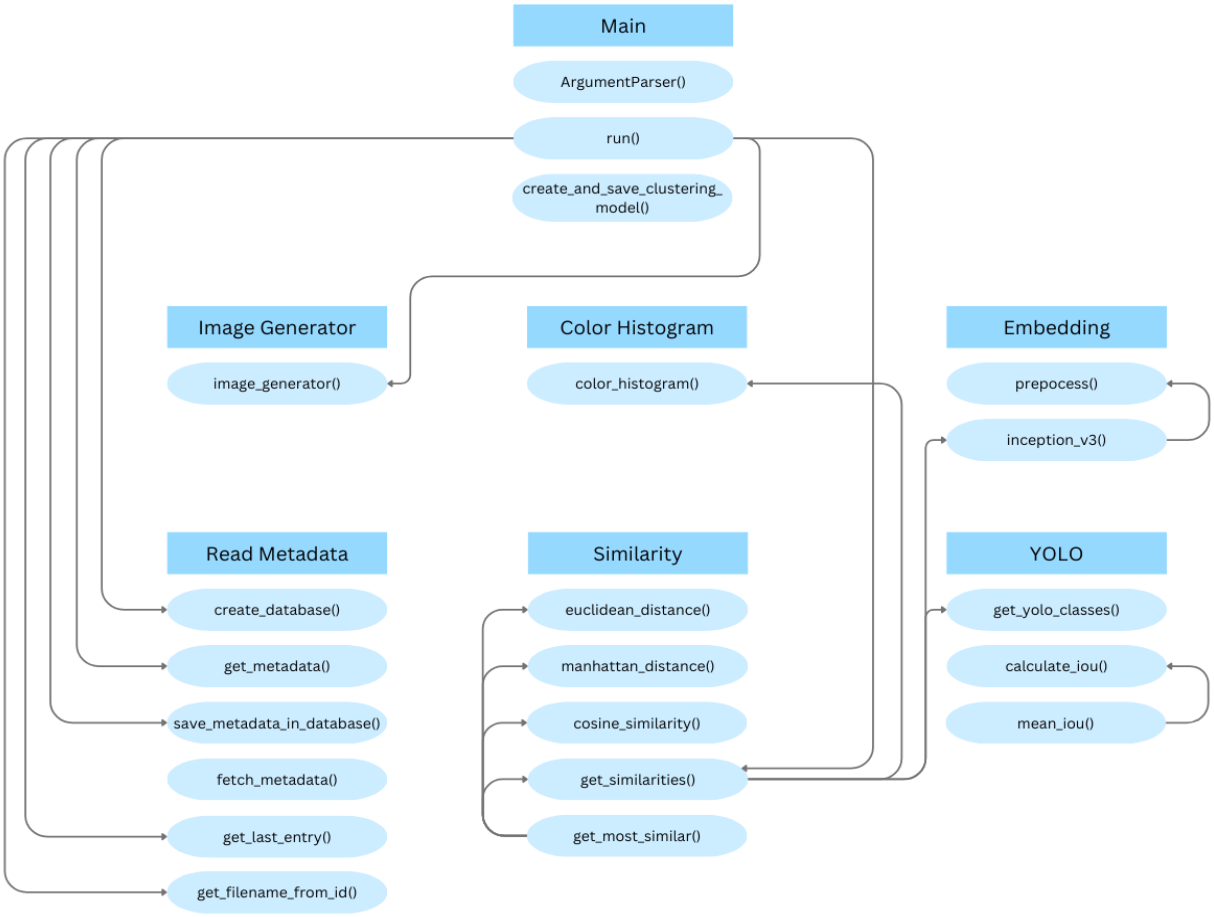

Abb. 1: Flowchart Programmdesign. Die Pfeile zeigen die Interaktionsreihenfolge der Programmbausteine.

Im Folgenden werden die wichtigsten Bausteine näher erläutert.

Die ImageGenerator-Klasse dient zum Generieren von Bildpfaden aus einem angegebenen Verzeichnis. Sie enthält eine Methode image_generator, die nacheinander Bilder (ihre Pfade) aus dem Verzeichnis liefert, beginnend bei einem optional spezifizierten Startpfad. Die Methode durchläuft rekursiv das Verzeichnis, identifiziert Bilddateien mit den Endungen .png, .jpg und .jpeg, und liefert diese als Pillow Images zurück. Falls beim Öffnen eines Bildes ein Fehler auftritt, wird eine entsprechende Fehlermeldung ausgegeben. Sollte der Generator aus irgendeinem Grund abstürzen, kann das Generieren der Bilder fortgesetzt werden, indem der letzte verarbeitete Pfad als Startpfad angegeben wird.

Die Nutzung einer SQLite-Datenbank image_metadata.db erlaubt die systematische Speicherung und Verwaltung von Bildmetadaten (siehe Abb. 2).

Eine Tabelle metadata wird definiert, die wichtige Eigenschaften wie ID, Dateiname (Dateipfad), Höhe, Breite, Format und Farbmodus der Bilder enthält. Funktionen wie create_database und save_metadata_in_database tragen dazu bei, Metadaten effizient in die Datenbank einzufügen und zugänglich zu machen, was eine schnelle Abfrage und Analyse von Bilddaten ermöglicht und die Identifikation spezifischer Dateien über deren ID erleichtert.

	id	filename	height	width	format	mode
0	1	data\image_data\coco2017_train\train2017\00000...	425	640	JPEG	RGB
1	2	data\image_data\coco2017_train\train2017\00000...	500	381	JPEG	RGB
2	3	data\image_data\coco2017_train\train2017\00000...	426	640	JPEG	RGB
3	4	data\image_data\coco2017_train\train2017\00000...	612	612	JPEG	RGB
4	5	data\image_data\coco2017_train\train2017\00000...	425	640	JPEG	RGB
5	6	data\image_data\coco2017_train\train2017\00000...	480	640	JPEG	RGB
6	7	data\image_data\coco2017_train\train2017\00000...	480	640	JPEG	RGB
7	8	data\image_data\coco2017_train\train2017\00000...	640	480	JPEG	RGB
8	9	data\image_data\coco2017_train\train2017\00000...	424	640	JPEG	RGB
9	10	data\image_data\coco2017_train\train2017\00000...	640	428	JPEG	RGB

Abb. 2: Ausschnitt der Datenbank zur Verwaltung von Bildmetadaten

3. Bildähnlichkeit

Nun werden die Methoden zur Messung der Bildähnlichkeit erläutert, erklärt und verglichen. Jede Methode hat ihre eigenen Vor- und Nachteile, die wir im Folgenden diskutieren.

1. Farbprofil- bzw. Farbähnlichkeitsmetrik

Die Farbprofil- bzw. Farbähnlichkeitsmetrik ist ein Verfahren zur Bildanalyse, das auf der Verwendung von Histogrammen basiert. Dabei werden für jedes Bild Histogramme erstellt, die die Verteilung der Farben in den RGB-Kanälen (Rot, Grün, Blau) darstellen. Das Ziel dieser Methode ist es, die Ähnlichkeit zwischen Bildern anhand ihrer Farbverteilungen zu bestimmen.

Histogrammberechnung für jeden Farbkanal:

Für jedes Bild werden drei separate Histogramme berechnet, die die Intensitätsverteilung der Farben in den jeweiligen Kanälen Rot, Grün und Blau darstellen. Diese Histogramme werden typischerweise auf eine bestimmte Anzahl von Bins aufgeteilt, was die Genauigkeit und die Detailtiefe der Farbverteilung steuert. Der Standardwert für Bins ist oft 256, was den möglichen Intensitätswerten eines 8-Bit-Bildes entspricht.

Normalisierung:

Nach der Berechnung werden die Histogramme normalisiert. Dies bedeutet, dass jeder Wert des Histogramms durch die Gesamtanzahl der Pixel des jeweiligen Kanals geteilt wird. Durch die Normalisierung können Histogramme unabhängig von der Bildgröße oder der Gesamthelligkeit miteinander verglichen werden.

Gruppierung und Mittelung:

Die Werte des Histogramms können in Gruppen zusammengefasst werden, und für jede Gruppe wird der Durchschnitt berechnet. Diese Gruppierung reduziert die Anzahl der Werte und somit die Komplexität der Daten, was den Vergleich zwischen Bildern erleichtert. Somit wird die Berechnung effizienter.

Konkatenation:

Die resultierenden Histogramme der drei Farbkanäle werden hintereinander zu einem einzelnen Vektor zusammengesetzt. Dieser Vektor repräsentiert das gesamte Farbprofil des Bildes und wird zur Berechnung der Ähnlichkeit zwischen verschiedenen Bildern herangezogen.

Vorteile:

- Schnelle Berechnung: Da Histogramme einfach zu berechnen sind, ist die Methode sehr schnell.
- Unabhängigkeit von Bildgröße: Die Methode funktioniert unabhängig von der Größe der Bilder, da die Histogramme normalisiert werden.

Nachteile:

- Ignoriert die räumliche Anordnung: Die räumliche Anordnung der Farben im Bild wird nicht berücksichtigt. Das bedeutet, dass zwei Bilder mit identischen Histogrammen visuell sehr unterschiedlich aussehen können, wenn die Farben in unterschiedlichen Bereichen angeordnet sind.
- Verlust von Detailinformationen: Durch das Zusammenfassen der Histogrammwerte kann es zu einem Informationsverlust kommen, insbesondere wenn die Gruppengröße groß ist. Dadurch könnten feine Unterschiede zwischen Bildern übersehen werden.

2. Embeddings:

Die Implementierung besteht aus zwei Hauptfunktionen, die ein Bild für das Inception v3 Modell vorbereiten und anschließend die tiefen Merkmale des Bildes extrahieren. Die erste Funktion, preprocess, optimiert das Bildformat durch Anpassungen wie Skalierung und Kanalnormalisierung, um es kompatibel mit den Anforderungen des Modells zu machen. Die zweite Funktion, inception_v3, nutzt das vorbereitete Bild, um es durch das Inception v3 Modell zu verarbeiten und eine Merkmalsextraktion durchzuführen.

Bildvorbereitung (preprocess Funktion):

Das Bild wird zunächst auf eine einheitliche Größe transformiert und anschließend in einen Tensor umgewandelt, um die Verarbeitung durch das neuronale Netz zu ermöglichen. Bei Graustufenbildern wird eine Anpassung vorgenommen, um die erforderlichen drei Kanäle für das Modell zu simulieren. Schließlich wird das Bildformat so angepasst, dass es als einzelnes Element in einem Batch verarbeitet werden kann.

Merkmalsextraktion (inception_v3 Funktion):

Das Inception v3 Modell wird mit Standardgewichten initialisiert und zur Merkmalsextraktion angepasst, indem die letzte Schicht entfernt wird. Das vorbereitete Bild wird dem Modell zugeführt, das auf CPU oder GPU läuft, um die Bildmerkmale zu extrahieren. Die extrahierten Merkmale werden als flacher Vektor ausgegeben, der weiterverwendet oder analysiert werden kann.

Vorteile:

- Automatisierte Vorverarbeitung: Die preprocess Funktion automatisiert die Skalierung und Anpassung der Bilder an die Eingabeanforderungen des Modells, was den Workflow vereinfacht und Fehlerquellen reduziert.
- Hohe Genauigkeit der Merkmalsextraktion: Das Inception v3 Modell ist bekannt für seine tiefe und präzise Merkmalsextraktion, was es besonders leistungsfähig macht.

Nachteile:

- Potenzieller Informationsverlust: Durch das Skalieren und Zuschneiden der Bilder auf eine feste Größe können wichtige Details verloren gehen, die für die Bildanalyse von Bedeutung sind.
- Spezialisierung auf ein Modell: Die spezifischen Anpassungen für das Inception v3 Modell machen die Implementierung weniger flexibel für den Einsatz mit anderen Modellen ohne zusätzliche Anpassungen.

3. YOLO-Modell (You Only Look Once):

In diesem Ansatz wird ein vortrainiertes YOLO v8 Modell verwendet, um Objektklassen und deren zugehörige Begrenzungsrahmen aus den Eingabebildern zu extrahieren. Zusätzlich stehen Funktionen zur Verfügung, die die Intersection over Union (IoU) zwischen den erkannten Objekten verschiedener Bilder berechnen und die durchschnittlichen IoU-Werte ermitteln. Das Hauptziel besteht darin, die Ähnlichkeiten zwischen unterschiedlichen Bildern anhand der Position und Größe der enthaltenen Objekte zu analysieren.

Objekterkennung (get_yolo_classes Funktion):

Das vortrainierte YOLO-Modell wird verwendet, um Vorhersagen über die Objekte im Bild zu generieren. Nachdem das Modell auf das Bild angewendet wurde, werden die Klassenindizes und die zugehörigen Begrenzungsrahmen (Bounding Boxes) der erkannten Objekte aus den Vorhersagen extrahiert. Diese Informationen werden dann in einem Dictionary gespeichert, das eine strukturierte Darstellung der erkannten Objekte ermöglicht.

IoU-Berechnung (calculate_iou Funktion):

Zur Bewertung der räumlichen Überschneidung zwischen zwei Begrenzungsrahmen wird zunächst die Fläche der Überschneidung berechnet. Dies beinhaltet das Finden des Bereichs, der von beiden Bounding Boxes gemeinsam abgedeckt wird. Anschließend wird die Gesamtfläche ermittelt, die von beiden Begrenzungsrahmen zusammen umfasst wird. Aus diesen Flächen wird der IoU-Wert berechnet, der ein quantitatives Maß für die räumliche Überschneidung der Bounding Boxes darstellt.

Durchschnittliche IoU-Berechnung (mean_iou Funktion):

Um ein Gesamtmaß für die Ähnlichkeit zwischen Objekten in zwei Bildern zu ermitteln, wird für jedes Paar von Begrenzungsrahmen, die zur selben Klasse gehören und aus verschiedenen Bildern stammen, der IoU-Wert berechnet. Aus diesen Werten wird der Durchschnitt gebildet, der als Maß für die allgemeine Ähnlichkeit der Objekte in den Bildern dient.

Vorteile:

- Skalierbarkeit: Die Implementierung kann für große Bildsätze eingesetzt werden, da sie effizient mit großen Datenmengen umgehen kann.
- Genauigkeit bei der Lokalisierung: YOLO ist bekannt für seine Fähigkeit, Objekte genau zu lokalisieren, was präzise Bounding Box Vorhersagen ermöglicht.

Nachteile:

- Anfälligkeit für Fehlalarme: YOLO kann in komplexen oder dicht besiedelten Umgebungen zu einer erhöhten Rate an falsch positiven Erkennungen neigen.

Vergleichen aller Methoden zur Messung der Bildähnlichkeit:

Die drei Methoden zur Bildähnlichkeitsmessung – Farbprofilmetrik, Inception v3 (für Embedding) und YOLO – bedienen unterschiedliche Aspekte der Bildanalyse. Die Farbprofilmetrik, die auf Farbhistogrammen basiert, bietet schnelle und größenunabhängige Vergleiche, berücksichtigt jedoch nicht die räumliche Anordnung der Farben, wodurch ähnliche Farbverteilungen, aber unterschiedliche Bildkompositionen, als gleich bewertet werden können. Inception v3 extrahiert tiefere Bildmerkmale und ermöglicht eine präzisere Analyse, ist jedoch auf feste Bildgrößen beschränkt und nicht flexibel hinsichtlich der Modellanpassung. YOLO hingegen fokussiert auf Echtzeit-Objekterkennung und bewertet Bilder anhand der räumlichen Übereinstimmung von Objekten, was es in komplexen Umgebungen fehleranfällig macht, jedoch eine genaue Lokalisierung und gute Skalierbarkeit bietet. Jede Methode hat somit spezifische Stärken und Schwächen.

Exemplarische Ausführung:

In der webbasierten Anwendung wird ein oder mehrere Bilder durch den Nutzer ausgewählt und hochgeladen, woraufhin die Bildähnlichkeit anhand ausgewählter Metriken (die bereits erwähnt wurden) wie Farbe, Embedding und YOLO analysiert wird (siehe Readme). Ergebnisse werden basierend auf der gewählten Distanzmetrik dargestellt (siehe Abb. 3 & 4).

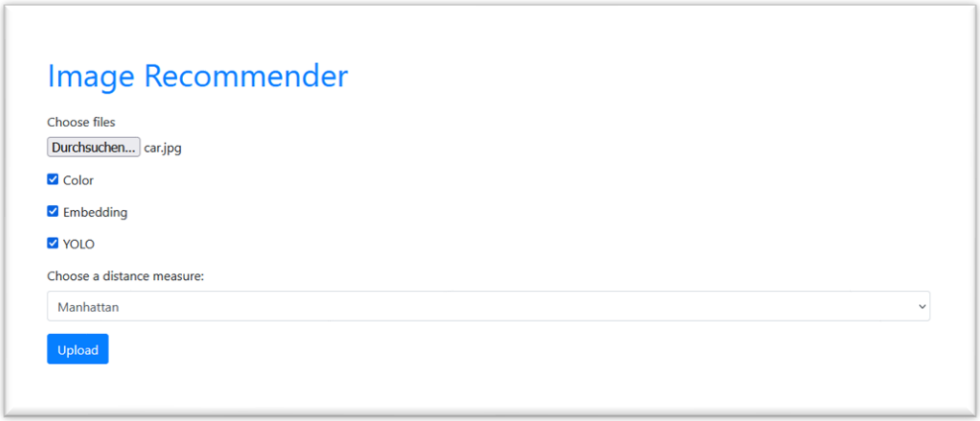

Abb. 3: Beispielhafte Bildeingabe auf der Web-Oberfläche

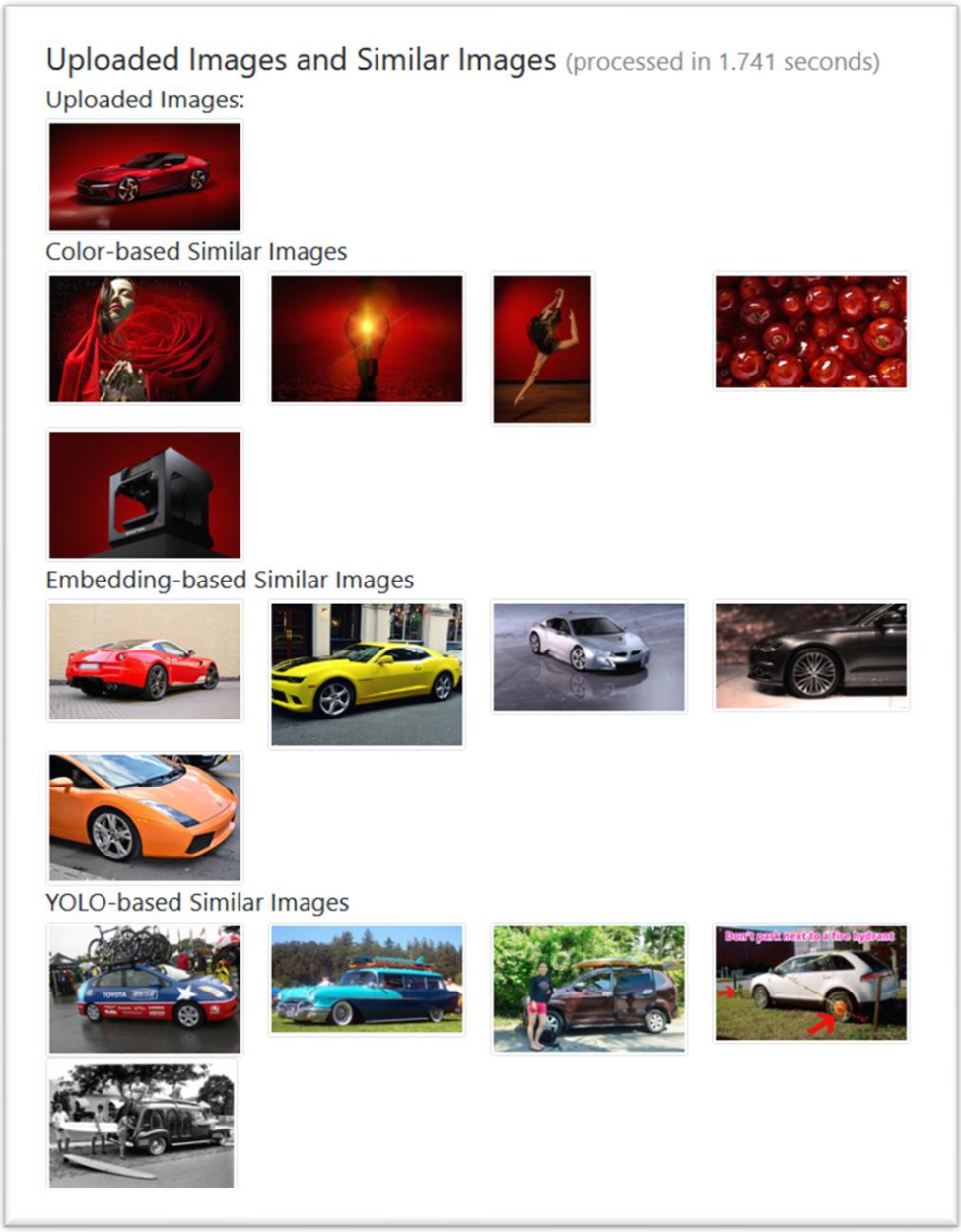

Abb. 4: Ähnliche Bilder basierend auf allen Metriken

4. Performanceanalyse und Laufzeitoptimierung

Bei der Analyse der Laufzeiten unserer Bildempfehlungssoftware für fünf verschiedene Eingabebilder hat sich gezeigt, dass die Funktion get_most_similar mit einer Ausführungszeit von 17 Sekunden pro Aufruf besonders hervorsticht (siehe Abb. 5). Diese Funktion spielt eine zentrale Rolle bei der Ermittlung ähnlicher Bilder und beansprucht den größten Teil der Verarbeitungszeit, was vor allem auf umfangreiche Distanzberechnungen zurückzuführen ist.

Um die Effizienz zu steigern und die Antwortzeiten zu verkürzen, haben wir Clustering-Techniken implementiert. Diese Vorgehensweise erlaubt es uns, die Bilder im Vorfeld in Gruppen ähnlicher Bilder zu segmentieren (siehe Teil 2). Dadurch müssen wir nicht jedes Bild mit jedem anderen im Datensatz vergleichen, sondern können die Distanzberechnungen auf die Bilder beschränken, die sich im selben Cluster befinden. Dies reduziert die Anzahl der nötigen Vergleiche drastisch und verbessert die Gesamtleistung des Systems.

Datei	Funktion	Anzahl der Aufrufe	Durchschnittszeit in Sek.
main.py	load_pkl_files	1	7,4508
color_vector.py	color_histogram	5	0,0031
embedding.py	prepocess	5	0,0014
	inception_v3	5	0,5195
yolo.py	get_yolo_classes	5	0,4842
similarity.py	get_similarities	5	1,0127
	get_most_similar	1	17,0047
Gesamter Prozess			**26,4764**

Abb. 5: Aktuelle Performance-Übersicht der Kernfunktionen

5. Machbarkeitsanalyse

Bei der Analyse unseres Bildempfehlungssystems, das fünf Eingabebilder auf einem Datensatz von fast einer halben Million Bildern verarbeitet, benötigt das System durchschnittlich 26 Sekunden, um die relevanten Vergleiche durchzuführen. Diese Zeit ist angesichts der Datenmenge als moderat bis gut zu bewerten, besonders wenn man die Komplexität des Vergleichsprozesses berücksichtigt, denn schließlich werden alle drei Methoden zur Bestimmung der ähnlichsten Bilder gleichzeitig angewendet und somit 15 Bilder zurückgegeben.

Diese Performance zeigt, dass unser System in seiner aktuellen Form ziemlich gut skalierbar ist und effektiv mit großen Bildmengen umgehen kann. Es bleibt jedoch die Frage offen, wie sich das System verhalten würde, wenn es auf noch größere Datenmengen, wie zum Beispiel 10 Millionen Bilder, skaliert wird. Hier könnte unser System an seine technischen und kapazitativen Grenzen stoßen. In solchen Fällen könnten die Verarbeitungszeiten deutlich ansteigen, was die Praktikabilität und Effizienz des Systems beeinträchtigen würde.

Um diese Herausforderung zu bewältigen und die Skalierbarkeit unseres Systems weiter zu verbessern, sollten wir über den Einsatz fortschrittlicher Technologien nachdenken. Techniken wie verteilte Verarbeitung und die Nutzung von GPUs könnten dazu beitragen, die Datenverarbeitung effizienter zu gestalten. Ebenso könnte der Einsatz schnellerer Indexierungsmethoden, wie etwa Faiss für Nearest-Neighbor-Suchen, eine Schlüsselrolle spielen, um die Suchzeiten in extrem großen Bilddatenbanken zu verkürzen.

Letztendlich zeigt unsere Analyse, dass unser Bildempfehlungssystem mit der aktuellen Größe des Datensatzes zurechtkommt und eine solide Basis für die Skalierung auf größere Datenmengen bietet. Jedoch sind weitere Anpassungen und Optimierungen notwendig, um die Herausforderungen bei der Skalierung auf Datensätze in der Größenordnung von 10 Millionen Bildern effektiv zu meistern.

Teil 2 – Big data image analysis

Die folgenden Abbildungen (Abb. 6) zeigen die Positionen aller Farbhistogramme, die mithilfe von UMAP auf drei Dimensionen reduziert wurden. Zusätzlich sind die verschiedenen Cluster (n_clusters=41), die wir mittels KMEANS berechnet haben, farbig dargestellt.

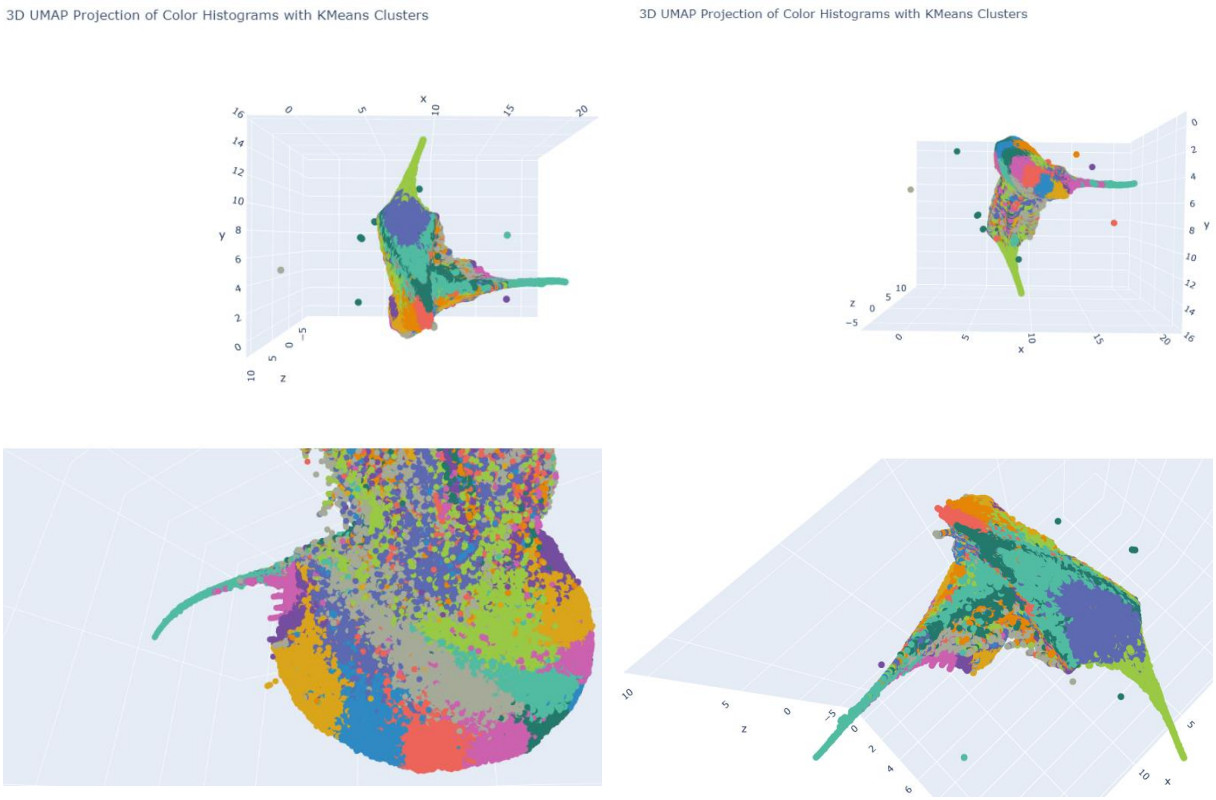

Abb. 6: Visualisierung der Farbprofile in einem 3D-Raum

Wir lernen daraus, dass die Farbschemata der Bilder insgesamt nah beieinander liegen, da die verschiedenen Cluster sich angrenzen oder überschneiden. Außerdem gibt es offensichtlich nicht viele Bilder, die aus diesem Raster fallen, da wir nur wenige Punkte außerhalb der großen Hauptstruktur sehen. Wir erkennen aber auch die Grenzen, an denen die Bilder offenbar nicht gut eingeordnet werden, da die Cluster teilweise sehr chaotisch aussehen. Das könnte zum einen an der Menge an Clustern liegen aber auch am Informationsverlust bei der Erstellung der Histogramme.

Mit derselben Methode haben wir auch die Embeddings geclustert (n_clusters=45) und dann wieder auf drei Dimensionen reduziert (Abb. 7).

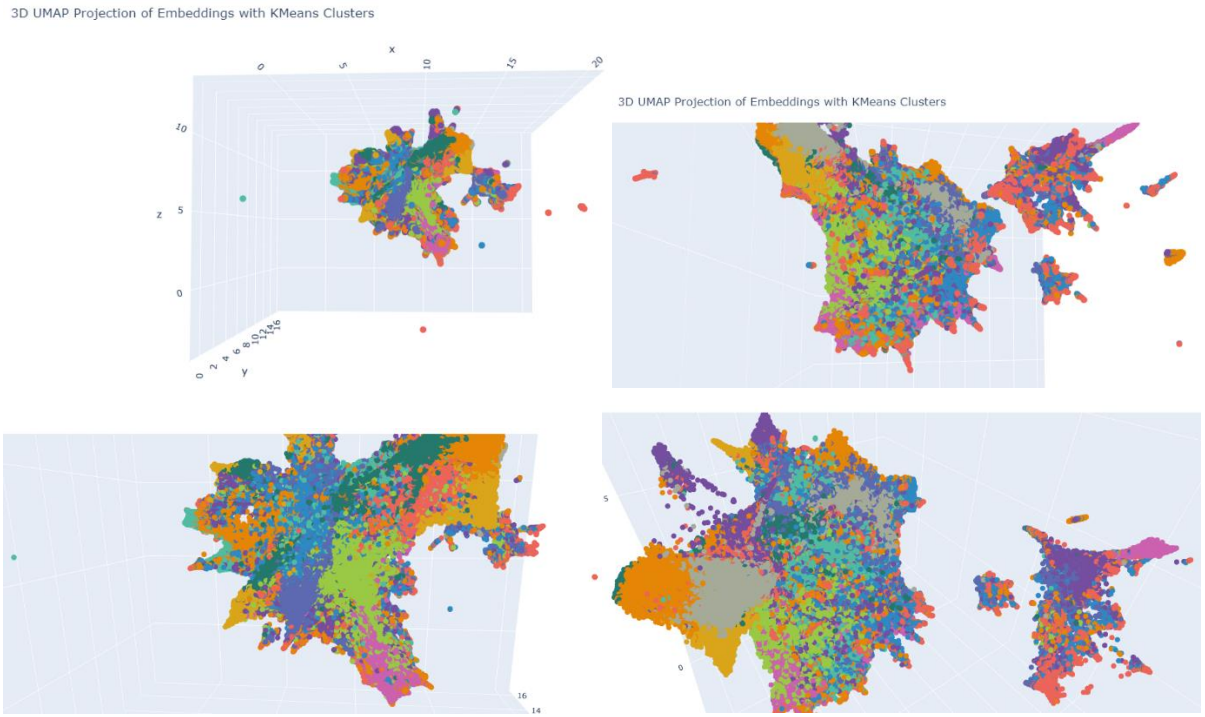

Abb. 7: Visualisierung der Embeddings in einem 3D-Raum

Wir sehen, dass das Clustering oft sehr gut funktioniert hat, aber auch wieder teilweise die Bilder nicht gut einordnen konnte. Dies kann ebenfalls wieder an der Anzahl der Cluster liegen oder auch schon an den ermittelten Feature Vektoren. Erwähnenswert ist hier allerdings noch, dass es neben einer großen Hauptstruktur einige kleinere Strukturen gibt. Große Unterschiede zwischen den Embeddings werden also korrekt erkannt und sogar zu Substrukturen zusammengefasst. Insgesamt erscheint dieser Plot aber chaotischer und komplexer als der vorherige.

Insgesamt kann man sagen, dass die Plots durchaus Sinn ergeben aber sicherlich noch verbessert werden können, indem anderen Algorithmen zur Dimensionsreduzierung und Clustering wie z.B. T-SNE bzw. DBSCAN benutzt werden. Alternativ könnte man optimalere Parameter für KMEANS durch entsprechende Analyse der Elbow und Silhouette Plots ermitteln