Duale Hochschule Sachsen

Staatliche Studienakademie Leipzig

Hausarbeit zur Singulärwertzerlegung

im Rahmen des Studiengangs Bachelor of Science
in der Studienrichtung Informatik
im Modul Algorithmen und Datenstrukturen (5CS-TI2AD-30)

Eingereicht von: **Malte Luca Peukert**
Beethovenstraße 27, 06749 Bitterfeld-Wolfen
Seminargruppe: CS24-1
Matrikelnummer: 5002722

Leipzig, 28. Dezember 2025

Inhaltsverzeichnis

1 Einleitung

In unserer digitalisierten Welt sind Bilder allgegenwärtig. Durch die Weiterentwicklung von Mobiltelefonen, hat so gut wie jeder eine hochauflösende Kamera in der Hosentasche. Mit dieser steigenden Auflösung, wächst der Bedarf an effizienter Speicherung und Übertragung. Dafür ist es notwendig wirksam und zuverlässig Bilddaten zu komprimieren. Klassische Verfahren wie zum Beispiel JPEG oder PNG nutzen mathematische Transformation, um redundate Informationen zu entfernen und damit den Speicherbedarf signifikant zu reduzieren. Eine besonderst effektives und effizientes mathematisches Verfahren ist dafür die Singulärwertzerlegung (Singular Value Decomposition, SVD). Sie sie hat große Bedeutung für die Bildverarbeitung, aber auch weitreichende Anwendungen in der Signalverarbeitung, Datenanalyse, Statistik und maschinellem Lernen[1].

Die SVD ist ein zentrales Konzept der linearen Algebra[2]. Besonders relevant für Bildkompression ist ihre Eigenschaft, Matrizen optimal durch niedrigere Ränge approximieren zu können. Diese Tatsache, die im sogenannten Eckart-Young-Theorem formalisiert ist und bis heute als Grundlage zahlreicher Kompressionsverfahren dient.

Diese Hausarbeit untersucht die Anwendung der SVD zur Speicherung eines vereinfachten Bildes und zeigt wie, mithilfe der Rang-k-Approximation Speicherplatz reduziert werden kann, ohne die wesentlichen Bildinformationen zu verlieren

2 Problemstellung

Hochauflösende digitale Bilder bestehen aus einer großen Anzahl von Pixeln, die üblicherweise als eine zweidimensionalen Matrix gespeichert werden. Dementsprechend wächst der Speicherbedarf eines unkomprimierten Bildes proportional zur Anzahl der Pixel. Bei einer Größe von 15x25 enthält die Matrix bereits 375 einzelne Werte, während ein Bild im HD-Format bereits über zwei Millionen Pixel umfasst.

Die zugrunde liegende Aufgabe dieser Arbeit besteht darin,

1. die Matrixdarstellung des Buchstabens „F“ aus einer Textdatei einzulesen,
2. die vollständige SVD dieser Matrix berechnen,
3. sukzessive Rang-k-Approximation zu erzeugen und analysieren,
4. die Veränderung der Bildqualität bei zunehmendem k zu untersuchen,

5. die erzielten Speicherersparnisse gegenüber der Originalmatrix zu bestimmen.

Das zu Grunde liegende Ziel ist es zu verstehen, wie stark ein Bild komprimiert werden kann, ohne an Erkennbarkeit oder Informationswert zu verlieren.

3 Mathematische Grundlagen der Singulärwertzerlegung

3.1 Grundlagen der Matrixdarstellung von Bildern

Ein Graustufenbild lässt sich mathematisch als Matrix

$$A \in \mathbb{R}^{m \times n} \tag{3.1}$$

darstellen. Jeder Eintrag a_{ij} entspricht dabei den Helligkeitswert eines Pixels dar. Binäre Bilder, wie das in dieser Hausarbeit benutzte „F", enthalten nur die Werte 0 und 1. Sie eignen sich trotzdem hervorragend zu Untersuchung von Strukturzerlegung, da sie klare Kanten und Zusammenhänge besitzen.

3.2 Definition der Singulärwertzerlegung

Die SVD zerlegt eine Matrix A in drei Komponenten:

$$A = U\Sigma V^T \tag{3.2}$$

Dabei gelten folgende Eigenschaften:

- $U \in \mathbb{R}^{m \times m}$ enthält die orthogonalen Basisvektoren der Zeilenstruktur.
- $\Sigma \in \mathbb{R}^{m \times n}$ enthält die Singulärwerte $\sigma_1 \geq \sigma_2 \geq \ldots$
- $V \in \mathbb{R}^{n \times n}$ enthält die orthogonalen Basisvektoren der Spaltenstruktur

Jeder Singulärwert σ_i in Σ gibt die Bedeutung des zugehörigen Basisvektors an. Größere Werte deuten auf wichtigere Strukturen hin, während kleinere Werte oft Rauschen oder weniger relevante Details repräsentieren.

3.3 Rang-k-Approximation

Die entscheidende Eigenschaft der SVD für die Bildkompression ist die Möglichkeit, eine Matrix A durch eine Rang-k-Approximation A_k zu nähern:

$$A_k = U_k \Sigma_k V_k^T \tag{3.3}$$

Hierbei werden nur die ersten k Singulärwerte und die zugehörigen Spalten von U und V verwendet. Diese Approximation minimiert den Frobenius-Norm-Fehler zwischen A und A_k, was bedeutet, dass A_k die beste mögliche Annäherung an A mit Rang k ist.

3.4 Speicherersparnis durch Rang-k-Approximation

Originalmatrix:

$$\text{Speicherbedarf von } O = m \times n \tag{3.4}$$

Die Speicherersparnis ergibt sich aus der Reduktion der benötigten Elemente zur Speicherung der Matrix. Anstatt die gesamte $m \times n$ Matrix A zu speichern, müssen nur noch die ersten k Spalten von U, die ersten k Zeilen und Spalten von Σ sowie die ersten k Spalten von V gespeichert werden. Die Gesamtanzahl der zu speichernden Elemente reduziert sich somit auf:

$$S(k) = k(m + n + 1) \tag{3.5}$$

Dies führt zu einer erheblichen Reduktion des Speicherbedarfs, insbesondere wenn k deutlich kleiner ist als sowohl m als auch n.

Relative Speicherersparnis:

$$\text{Relative Speicherersparnis} E(k) = 1 - \frac{S(k)}{O} = 1 - \frac{k(m + n + 1)}{m \times n} \tag{3.6}$$

4 Programmbeschreibung und Auswertung

In diesem Kapitel wird das entwickelte Python-Programm `svd_image_compression.py` detailliert beschrieben. Das Programm dient der praktischen Umsetzung der Singulärwertzerlegung zur Bildkompression und erfüllt alle Anforderungen der Aufgabenstellung. Es ermöglicht das Einlesen einer Bildmatrix, die Berechnung der reduzierten

SVD, die Rekonstruktion für verschiedene Rangwerte, die grafische Darstellung der Ergebnisse, die Berechnung von Fehlermaßen, der Energieerhaltung sowie der Speicherersparnis. Zusätzlich wird eine übersichtliche CSV-Datei zur Dokumentation der Ergebnisse erzeugt.

4.1 Lösungsansatz und Wahl der Programmiersprache

Zur praktischen Umsetzung wird Python in Verbindung mit der Bibliothek NumPy und Matplotlib verwendet. NumPy stellt eine hocheffiziente Implementierung der SVD bereit, die intern auf der LAPACK-Bibliothek basiert und Mathplotlib ermöglicht die anschauliche Visualisierung der Ergebnisse.

Python bietet darüber hinaus:

- hohe Lesbarkeit und einfache Syntax,
- einfache Datenverarbeitung,
- Plattformunabhängigkeit,
- schnelle Umsetzung von mathematischen Algorithmen.

4.2 Programmstruktur und Ablauf

Das Programm ist klar modular aufgebaut und besteht aus drei Hauptbereichen:

1. **Hilfsfunktionen** (Matrix-Einlesen, SVD, Rekonstruktion, Speicherersparnis, Fehlerberechnung).
2. **Verarbeitungsfunktion** `process_file` als zentrale Pipeline
3. **Komandzeileninterface (CLI)** mit `argparse`

Der grundsätzliche Ablauf des Programms ist wie folgt:

1. Einlesen der Bildmatrix aus `SVD_F.txt`.
2. Berechnung der reduzierten Singulärwertzerlegung.
3. Rekonstruktion der Bildmatrix für verschiedene Rangwerte k.
4. Speicherung der rekonstruierten Matrizen als PNG-Dateien.
5. Optionale Speicherung als Textdateien.

6. Berechnung und Ausgabe von Fehlermaßen, Energieerhaltung und Speichereinsparnis.
7. Erzeugung einer CSV-Datei mit den Ergebnissen.

4.3 Einlesen der Bildmatrix

Das Einlesen der Bilddaten erfolgt über die Funktion:

```
def load_matrix(path: str):
```

Die Matrix wird aus einer Textdatei eingelesen:

```
A = np.loadtxt(path)
```

Die Datei `SVD_F.txt` enthält die binäre Matrix des Buchstabens „F“. Jeder Eintrag entspricht einem Pixelwert (0 oder 1). Nach dem Einlesen wird die Matrix in ein NumPy-Array vom Typ `float` umgewandelt, das für die weitere Verarbeitung verwendet wird.

4.4 Berechnung der Singulärwertzerlegung

Die Funktion:

```
def compute_svd(A: np.ndarray):
```

berechnt die reduzierte SVD der Matrix A mittels NumPy:

```
U, S, Vt = np.linalg.svd(A, full_matrices=False)
```

Die Zerlegung erfolgt gemäß der Definition:

$$A = U\Sigma V^T \tag{4.1}$$

Dabei enthalten:

- U: linke Singulärvektoren,
- S: Singulärwerte (als 1D-Array),
- V^T: transponierte rechte Singulärvektoren.

Die reduzierte Form (`full_matrices=False`) sorgt dafür, dass nur die notwendigen Komponenten berechnet werden, was Speicher spart.

4.5 Rekonstruktion der Rang-k-Approximation

Die Rang-k-Approximation wird durch die Funktion:

```
def reconstruct_rank_k(U: np.ndarray, S: np.ndarray, Vt: np.ndarray, k: int)
```

berechntet. Die mathematische Umsetzung lautet:

$$A_k = U_k \Sigma_k V_k^T \tag{4.2}$$

Dabei werden nur die ersten k Spalten von U, die ersten k Singulärwerte und die ersten k Zeilen von V^T verwendet. Die Implementierung erfolgt direkt als Matrixprodukt:

```
return U[:, :k] @ np.diag(S[:k]) @ Vt[:k, :]
```

Durch die vektorisierte Form ist die Rekonstruktion sowohl numerisch stabil als auch effizient. Fals ein ungültiger Wert für k übergeben wird, wird eine ValueError ausgelöst:

```
if k <= 0:
  raise ValueError("k muss >= 1 sein")
```

4.6 Speicherung der Rekonstruktionen

Die rekonstruierten Matrizen werden als PNG-Bilder gespeichert. Dies erfolgt in der Funktion:

```
def save_reconstruction(Ak: np.ndarray, filename: str)
```

als Textdateien gespeichert:

```
np.savetxt(filename, Ak, fmt="%.3f")
```

Die Speicherung mit drei Nachkommastellen ermöglicht eine kompakte Dokumentation der Näherungsmatrix, mit Werten wie 0.000, 0.333, 0.667 und 1.000, in einer Textdatei entsprechend des jeweiligen k, wie z.B. `reconstruction_k_1.txt`.

Zusätzlich werden alle Rekonstruktionen als Graustufenbilder gespeichert:

```
plot_matrix(Ak, title=f'Rang-{k}', savepath=png_name)
```

Damit entsteht eine visuelle Darstellung der Qualität der Approximation für jedes k.

Literatur

1. GOLUB, Gene H.; VAN LOAN, Charles F. *Matrix Computations*. 4. Aufl. Baltimore, MD: Johns Hopkins University Press, 2013.
2. STRANG, Gilbert. *Linear Algebra (MIT OpenCourseWare)* [https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/]. 2010. Accessed: 2025-11-25.

Anhang

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Hausarbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Bestandteile der Arbeit, die mittels Künstlicher Intelligenz entstanden sind, wurden ausdrücklich gekennzeichnet. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht noch einer anderen Prüfungsbehörde vorgelegt.

Leipzig, 10. Oktober 2025

Malte Luca Peukert