

**Punkt 0 & 4:**

Um unsere Datensätze von .PNG zu .JPEG/.JPEG2000/.JPRG XR zu komprimieren, haben wir uns für eine automatisierte Variante über Java entschieden. Dort haben wir unseren Datensatz mit folgenden compression ratio komprimiert:

JPEG bis zu 1:40

JPEG2000 bis zu 1:40

AVIF bis zu 1:25

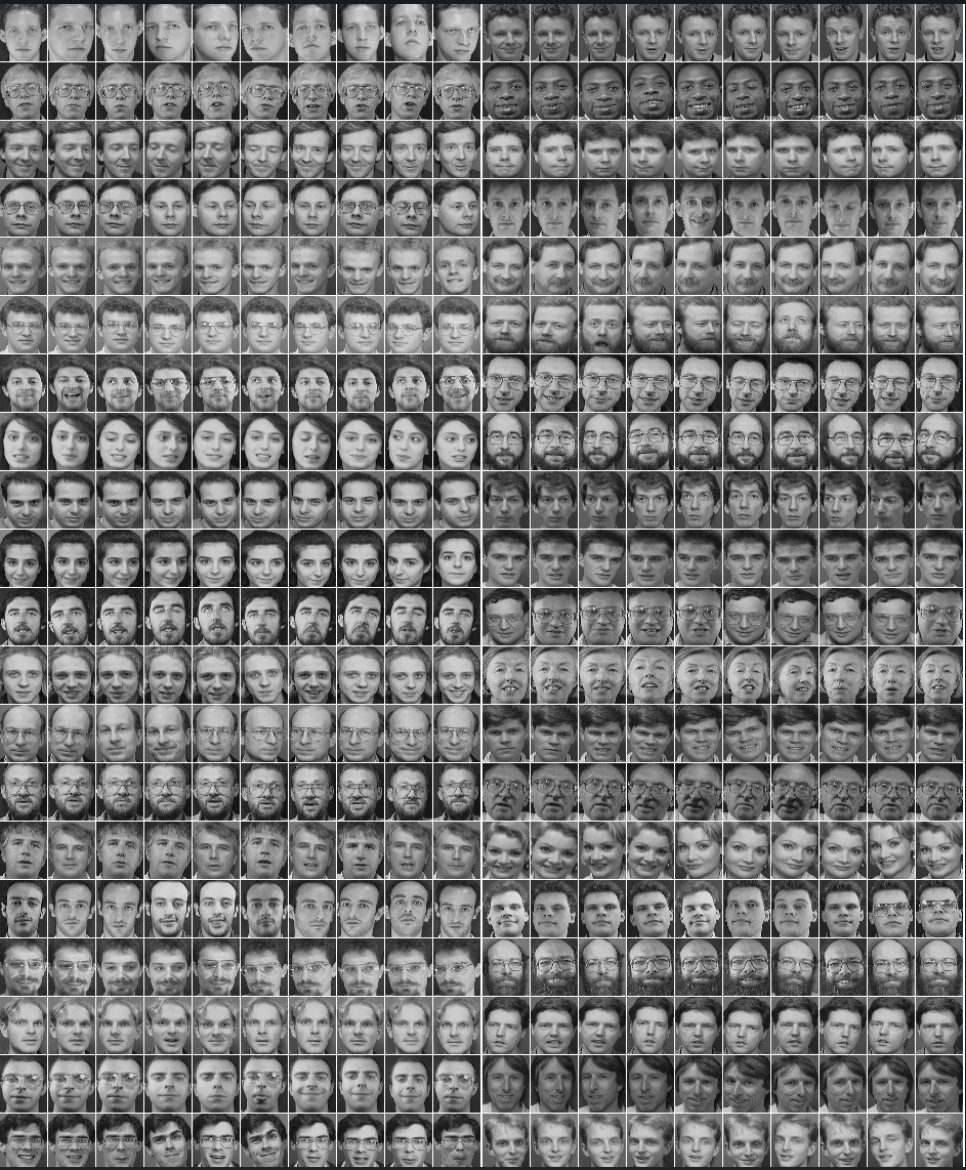
JPEGXR bis zu 1:25

Diese wurden dann sortiert um unsere Face Recognition zu testen.

Bei dem von uns gewählten Datensatz, handelt es sich nunmehr um den AT&T-Gesichtsdatensatz[[1]](#footnote-1), welcher früher 'The ORL Database of Faces' genannt wurde. Dieser enthält einen Satz von Gesichtsbildern, die zwischen April 1992 und April 1994 im Labor aufgenommen wurden.

Das Dataset besteht aus 400 Bildern, die eine Größe von 64x64 aufweisen. Innerhalb dieser 400 Bilder finden sich 40 unterschiedliche Personen. Daraus ergibt sich, dass es für jede Personen 10 unterschiedliche Bilder gibt. Diese unterscheiden sich von Lichtverhältnis, Betrachtungswinkel und Gesichtsausdruck. Einen Überblick über das Dataset verschafft das unten beigefügte Bild auf Seite 3.

Alle Bilder haben eine Abmessung von 92x112 und liegen bereits in schwarz-weiß vor.



**Punkt 1 & Punkt 5:**

In dieser Studie wurde die Gesichtserkennung mit den Bildern aus dem AT&T-Datensatz durchgeführt. Dies wurde in Python implementiert und brachte folgende Ergebnisse

Verwendet wurde hier die Eigenface Methode und diese haben wir mit unseren verschiedenen Kompressionsraten getestet.

Bei JPEG mit einer Kompressionsrate von 1:40 war das Ergebnis 90% Accuracy. Im Vergleich bei 1:10 war diese 95%.  
Bei JPEG2000 mit 1:40 Rate wurde ein Ergebnis von 85% erzielt und mit 1:10 auch um die 95%.

Bei JPEGXR war bei einer 1:25 Rate das Ergebnis auch knapp über 90% genau wie bei AVIF.

Allgemein kann man sagen, dass die Face Recognition relativ gut mit diesen Raten und Dataset arbeitet.

**Punkt 2:**

**full-reference quality metric:**

Versucht die Qualität eines Testbildes anhand eines Referenzbildes zu vergleichen, von dem angenommen wird, dass es eine perfekte Qualität hat. Z.b das Original Bild im Vergleich zu einem JPEG komprimierten Bild

Hier haben wir uns für Python entschieden und die von Python zur Verfügung gestellten Funktionen für die full-reference Metric implementiert.[[2]](#footnote-2)

Verwendete Funktionen:

• **Mean Squared Error (MSE)** misst die durchschnittliche quadratische Different zwischen dem Original und dem komprimierten Bild. Es wird jeder Pixel in der Helligkeitskomponente betrachtet.

*Hier gilt umso niedriger desto besser, MSE = 0 gibt es nur wenn die Bilder ident sind.*

• **Peak Signal-to-Noise Ratio (PSNR)** wird vom MSE abgeleitet und gibt das Verhältnis der maximalen Pixelintensität zur Leistung der Verzerrung an. Hier gilt umso größer der MSE, desto kleiner der PSNR Wert und umso größer der PSNR desto besser.

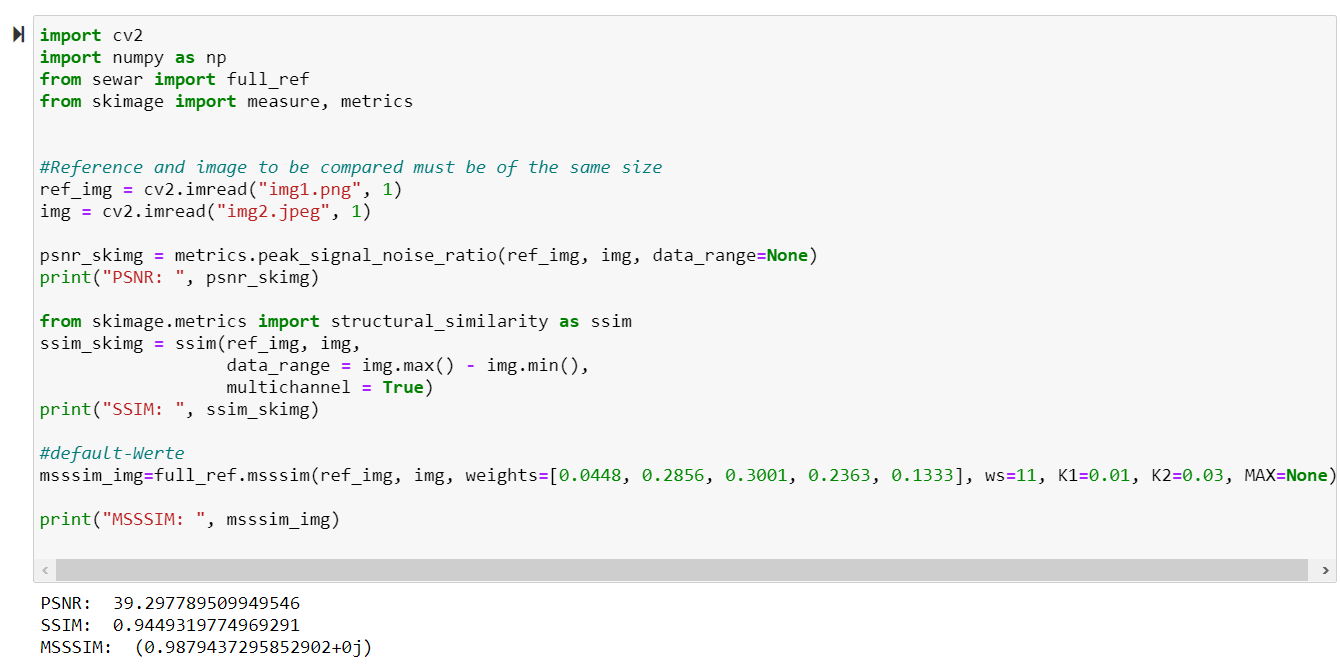
*Typische Werte sind hier zwischen 30 und 50dB und umso höher desto besser.*

• **Structural Similarity Index (SSIM**) Hier wird versucht, die Ähnlichkeit der Bildstruktur zu erfassen. Der Begriff „Struktur“ eines Bildes bezeichnet die Abhängigkeiten zwischen benachbarten Bildpunkten, die unabhängig von der Helligkeit und dem Kontrast in dieser Region des Bildes sind.

*Hier liegt die Maßzahl zwischen 0 (total unterschiedlich) und 1(identisch)*

• **Multi-scale Structural Similarity Index (MS-SSIM)** erweitert den SSIM-Index, indem sie Luminanz Informationen auf der höchsten Auflösungsebene mit Struktur- und Kontrastinformationen in mehreren heruntergerechneten Auflösungen bzw. Skalen kombiniert. Die verschiedenen Skalen berücksichtigen die Variabilität in der Wahrnehmung von Bilddetails, die durch Faktoren wie den Betrachtungsabstand zum Bild, den Abstand der Szene zum Sensor und die Auflösung des Bilderfassungssensors verursacht werden.

*Wenn der Wert näher bei 1 ist, ist die Qualität höher und Richtung 0 sinkt die Qualität. [[3]](#footnote-3)*



**Punkt 3:**

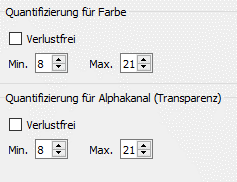
Hier haben wir uns für eine **Autoencoder** deep learning compression entschieden. Als Beispiel hier:

<https://www.tensorflow.org/tutorials/generative/autoencoder>

<https://colab.research.google.com/github/Justin-Tan/high-fidelity-generative-compression/blob/master/assets/HiFIC_torch_colab_demo.ipynb#scrollTo=9BKccvcTpj1k>

**Punkt 4 AVIF:**

Hier wurde unser AT&T Dataset auf das Format AVIF umgewandelt. Für das Format AVIF konnten die 3,30 KB durch das Ausprobieren von den unten gezeigten Parametern erreicht werden:



Dabei bezieht sich der 2. Parameter auf die Transparenz und der 1. auf die Farb-Quantifizierung selbst.

1. <https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/> [↑](#footnote-ref-1)
2. <https://sewar.readthedocs.io/en/latest/> [↑](#footnote-ref-2)
3. <https://de.mathworks.com/help/images/image-quality-metrics.html#:~:text=Full-Reference%20Quality%20Metrics,-Full-reference%20algorithms&text=Mean-squared%20error%20(MSE),the%20human%20perception%20of%20quality.&text=Peak%20signal-to-noise%20ratio%20(pSNR)>. [↑](#footnote-ref-3)