Ve svém programu se snažím o kompresi obrázku pomocí singulárního rozkladu. Cílem je, aby si obrázek zachoval co nejvíce kvality, zatímco se jeho velikost v paměti co nejvíce zmenší. Uživatel si tak může vybrat, zda obrázek zmenší manuálně (řekne konkrétně kolik singulárních hodnot zachová), nebo jen určí požadovanou kvalitu výsledného obrázku vůči původnímu.

Jak bylo již zmíněno, v programu je využíván singulární rozklad (anglicky Singular Value Decomposition), který rozloží libovolnou matici A podle vzorce *A = USV**T*, kde U a V jsou ortonormální a S má na diagonále singulární hodnoty, které jsou seřazeny sestupně. Jelikož většina informací je uložena v prvních singulárních hodnotách je možné zahodit singulární hodnoty, které nesou méně informací s minimální ztrátou kvality.

Program se nejdříve zeptá, jaký obrázek chce uživatel načíst. Poté, co je obrázek vybrán, by se měl zobrazit přímo na obrazovce. Po zavření obrázku a potvrzení volby se program zeptá, jaký rank si přejete. Rank je počet singulárních hodnot, které jsou zachovány v diagonální matici S, která je získána singulárním rozkladem. Čili čím větší číslo (avšak nesmí být větší než menší strana vloženého obrázku v pixelech), tím méně informací zahodí a tím blíže bude kvalitou k původnímu obrázku. Avšak jak bylo zmíněno, většina informací je uložena v prvních singulárních hodnotách, proto je dosaženo nejlepšího poměru u ranku okolo 20 % menší strany. Tedy rank je možné zadat třemi způsoby - přímo napsat kolik singulárních hodnot zachovat, určit procentuálně kolik singulárních hodnot zachovat (čili v poměru k celkovému počtu singulárních hodnot - maximálnímu ranku), nebo zadat požadovanou kvalitu vůči původnímu obrázku a program si rank spočítá.

Poté se program zeptá, v jaké podobě s obrázkem pracovat - v původní barevné, bezbarevné (té je dosaženo tak, že se změní obrázek ze tří dimenzí na dvě), či černobílé. V případě výběru jiné podoby než barevné, dochází k dalšímu malému zmenšení velikosti obrázku.

V případě, že byl vybrán původní barevný obrázek, můžete jej ještě zmenšit pixelově. Jelikož singulární rozklad obrázek velikostně nezmenšuje - jen zahazuje přebytečné informace - je možnost obrázek zmenšit i takto, čímž dojde k dalšímu šetření paměti. Toto zmenšení funguje na principu zachování každého i-tého pixelu, proto se doporučuje hodnota do 5 (avšak programem jste limitováni až velikostí strany obrázku).

Na konci bude zobrazeno několik užitečných funkcí, které uživeteli řeknou něco o výsledném obrázku:

- Porovnání s původním obrázkem, kde se vám vedle sebe ukáže původní a vámi zmenšený obrázek
- Zobrazení hned dvou grafů - graf závislosti ranku na kvalitě, kde na ose x je rank (počet singulárních hodnot) a na ose y je kvalita v % vůči původnímu obrázku, a graf závislosti singulárních hodnot na ranku, kde na ose x je opět rank a na ose y je hodnota dané singulární hodnoty (na tomto grafu je vidět, o kolik více informací mají první singulární hodnoty),
- Porovnání výsledného obrázku pomocí standardních technik, jako jsou Mean Squared Error, Peak Signal-to-Noise Ratio a Structural Similarity Index. Čím větší jste zvolili rank tím by se měl MSE blížit 0, PSNR by se mělo blížit 50 a SSIM by se mělo blížit 1 (tato funkce nefunguje, pokud jste si zvolili zmenšení obrázku, jelikož mají matice výsledného obrázku a původního obrázku jiné rozměry, tudíž nejdou porovnávat).
- A nakonec znovu zobrazení výsledného obrázku.

Nejdůležitější funkce v mém programu jsou Compression_Color() and Compression_Gray(), které vykonávají samotný singulární rozklad. Obě fungují na velmi podobném principu. Obě funkce jsou volány s parametry A (původní matice) a rank (barevná ještě s DOWNSAMPLE, což je míra zmenšení, kterou si může uživatel vybrat). Poté je matice pomocí numpy funkce svd() rozložená na tři matice U, S, VT (funkce Compression_Color ještě předtím rozdělí matici A na tři matice podle barev a poté pracuje s každou maticí zvlášť). Matice U, S, VT jsou poté zmenšeny až po rank r a opět vynásobeny zpět v jednu matici (v Compression_Color jsou tyto tři výsledné matice složeny do jedné pomocí np.stack()). Výsledkem je zmenšená matice. Tyto dvě funkce ještě vypočítávají hodnoty jako jsou n (počet singulárních hodnot původního obrázku), ranks (rozsah ranků od nuly po n) a quality retained (list akumulovaných hodnot S). Tyto hodnoty se využívají při zobrazení grafu a jelikož je funkce np.linalg.svd() časově náročná (obzvláště u velkých obrázků) počítají se již teď.

Další důležitou funkcí je Calculate_Rank(), která vypočítá požadovaný rank podle uživatelem určené kvality. Je volána s parametry A (původní matice), rank a DOWNSAMPLE (opět jen pokud si uživatel zvolil zmenšit obrázek). Tato funkce provede singulární rozklad matice A. Pro urychlení výpočtu ji ale nejdřív převede do černobílé podoby. Poté vypočítá hodnoty n, ranks a quality_retained, které jsem již zmiňoval výše. Quality_retained je list délky n, který má jako hodnoty kvalitu vůči původnímu, tudíž stačí najít hodnotu požadované kvality v tomto listu a index této hodnotu bude námi hledaný rank. Jelikož se jedná o srovnaný list, budeme tuto hodnotu hleadt funkcí Find_closest_number_index(), která pomocí půlení intervalů najde číslo nejblíže k námi hledanému a vrátí jeho index, což je požadovaný rank.

Dále program obsahuje funkce Graph_quality() a Graph singular, které obě zobrazují zmenšený obrázek a graf pomocí knihovny matplotlib.

Funkce Compare() je zavolána s parametry A (původní obrázek), compressed_image (zmenšený obrázek) a gray (který hlídá, zda zobrazit obrázek v černobílé)

Další tři funkce počítají již zmíněné MSE, PSNR a SSIM. MSE se počítá podle vzorce

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(original - compressed)^2.$$

PSNR podle vzorce

$$PSNR = 10 \cdot log_{10}(\frac{MAX^2}{MSE}) = 20 \cdot log_{10}(MAX) - 10 \cdot log_{10}(MSE),$$

kde MAX je v našem případě 255, protože počítáme s 8 bitovými obrázky. A SSIM podle vzorce

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x + \mu_y + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

kde μ značí vážený průměr a σ váženou varianci (resp. kovarianci) signálů (x pro původní matici, y pro zmenšenou). A $c_i = (K_i L)^2$, kde K jsou malé konstanty (v mém případě 0,01 a 0,03) a L je rozsah hodnot pixelů (čili pro 8 bitů 255).

Poslední funkce Open_file() umožňuje otevřít obrázek přímo z počítače pomocí knihovny tkinter.