

Sluttprosjekt bildebehandling

Jim-Alexander Berger Gundersen

Våren 2019

1 Introduksjon

I denne rapporten vil jeg implementere to metoder for kompresjon av bilder. Generelt sett finnes det to typer kompresjon, tapsfri og komprimering med tap. Denne rapporten vil kun diskutere komprimering med tap, som betyr at resultatbildet ikke vil være 100% likt originalbildet, men målet er å komme så nære som mulig.

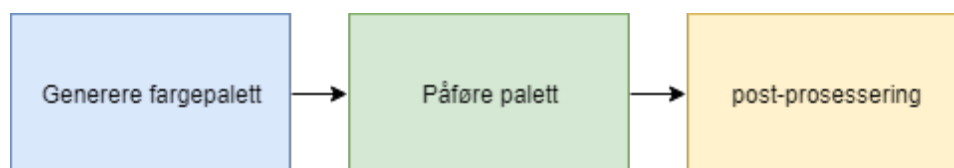
2 Metode

Det er to metoder som er implementert i dette prosjektet, kvantifisering og autoenkoding. I dette kapitlet vil jeg forklare hvordan disse algoritmene opererer.

2.1 Kvantifisering

Kvantifisering går ut på at bildets piksler blir fastsatt til et fast satt farger. En må da på forhånd bestemme hvilke farger som skal benyttes for kvantifiseringen, og deretter for hver piksel finne ut hvilken av de fargene som skal benyttes. Disse fargene kalles en palett eller fargepalett.

Selve komprimeringen kommer av at hver piksel ikke lenger trenger å representerer av 3 RGB verdier, disse er ofte mellom 0 og 255, og man trenger da $8 * 3 = 24$ bits per piksel i bildet. Med denne metoden kan man i stedet på forhånd definere 8 farger i paletten, og assosiere disse med en verdi mellom 0 og 7 og trenger da kun 3 bits per piksel for å representere bildet. Det vil komme med et tap i verdi, men i dette tilfellet vil en da få en komprimering på ca $3/24 = 12.5\%$. Dette er ikke medregnet plassen man må også bruke på å lagre paletten ved siden av bildet.



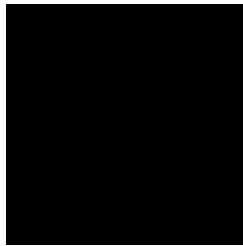
Figur 1: Sekvens for kvantifiseringsprosessen

I figur 1 kan en se sekvensen som må utføres for å utføre kvantifiseringsalgoritmen. Først må man bestemme seg for en fargepalett, deretter påføre denne paletten til bildet, til slutt kan man utføre post-prosesseringer for å forbedre kvaliteten på bildet.

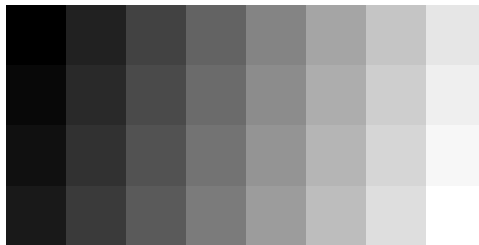
2.1.1 Valg av palett

Det finnes utallige måter å bestemme hvilken palett en skal benytte for kvantiseringsalgoritmen, og alle har fordeler og ulemper. I dette prosjektet har jeg valgt å implementere tre metoder for generering av fargepalett.

Den første er en uniform gråskala palett. Denne opererer kun med gråskala farger og henter ut disse uniform fra spekteret av gråskala farger. I figur 2 og 3 kan en se eksempler på gråskala paletter med 2 og 32 farger respektivt.

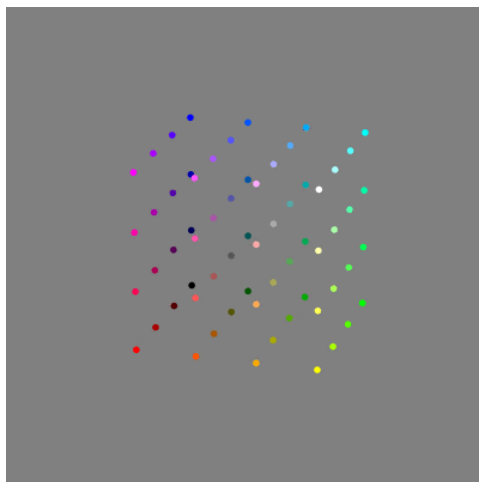


Figur 2: Gråskala palett med 2 farger (Bildet viser kun 2 piksler, om det vises en gradient av farger ber jeg deg skru av interpolering i ditt PDF-lesingsprogram)

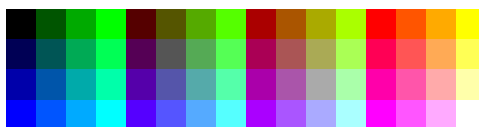


Figur 3: Gråskala palett med 32 farger

En annen palett er en hvor man uniformt henter ut farger fra alle RGB fargekanaler. Disse regnes ut ved en uniform distribusjon av hver fargekanal, lengden av disse vil alltid da være kubens av en toerpotens, som en kan se i figur 4.

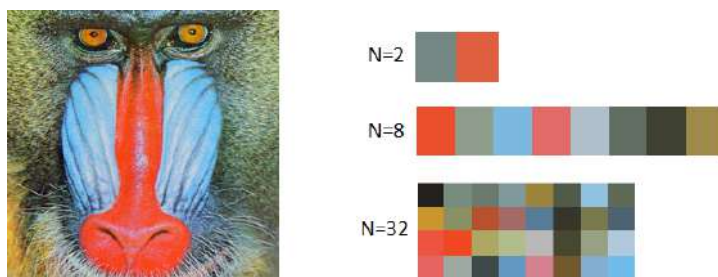


Figur 4: Distribusjonen av farger i RGB fargerom med 64 fager



Figur 5: RGB uniform palett med 64 farger som i figur 4

Den siste paletten er en dynamisk palett, her kommer fargevalget ann på selve bildet. Alle pikslene til bildet blir kartlagt i RGB fargerommet og ved hjelp av K-means algoritmen segmenteres disse fargene i N antall farger. I figur 6 kan en se hvilke farger algoritmen kommer opp med for et bilde av en mandrill



Figur 6: K-means algoritmen for generering av fargepalett, med $K=2,8,32$

Som sagt har disse tre metodene fordeler og ulemper, K-means algoritmen bruker ganske lang tid å prosessere bildet, og må gjøres for hvert enkelt bilde som skal komprimeres. Paletten som blir funnet kan heller ikke gjenbrukes på

andre bilder. Men de uniforme fargepalettene kan disse lagres på alle enheter som skal åpne den komprimerte filen, som fører til enda mer minskning av filstørrelse.

2.1.2 Hvordan påføre paletten

Etter man har funnet hvilken palett som skal brukes må denne påføres bildet som skal komprimeres. I dette prosjektet blir dette utført ved å gå gjennom hver piksel i bildet og finne ut hvilken farge i paletten som har den korteste distansen til pikselen ved bruk av distanseformelen.

$$\sqrt{(Pixel_R - Palett_R)^2 + (Pixel_G - Palett_G)^2 + (Pixel_B - Palett_B)^2}$$

Når den nærmeste palett fargen er funnet vil denne bli satt som fargen på pikselen, og man kan da fortsette til neste piksel.

2.1.3 Post-prosessering

Det finnes metoder for å forbedre resultatet av kvantifisering, den som er utført i dette prosjektet er floyd-steinberg dithering. Denne utføres ved at for hver piksel som man påfører palettfargen regner man ut differansen mellom originalpikselfargen og palettfargen, denne differansen kalles for kvantifiseringserror. Erroren blir ført videre til nabopiksler med varierende styrke som vist i matrisen under.

$$\begin{bmatrix} & & \\ \frac{3}{16} & * & \frac{7}{16} \\ \frac{5}{16} & & \frac{1}{16} \end{bmatrix}$$

Ved hjelp av denne metoden kan man oppnå et dithering mønster og bildet vil se mer utjevnet ut.

2.2 Autoencoder

Den andre metoden for komprimering er en såkalt autoencoder, dette er en type neuralt-nettverk som har som oppgave å lære seg effektive data enkoder. Den gjør dette ved å enten ha samme data som både input og output under læringen, eller ha data med støy som input og data uten støy som output for å bygge en mer robust modell.

Autoenkodere er ofte bygd opp av lag med konvolusjon etterfulgt av et lag kalt maxpooling, maxpooling tar en NxN kvadrat med piksler og henter ut den største verdien av disse og fører denne videre til neste lag. Om man har et input bilde av størrelse 32x32 og kjører dette gjennom et maxpooling lag som

er 2x2 vil det resulterende bilde blir 16x16 stort, en kan da gjenta prosessen til en når den størrelsen en vil komprimere bildet. For å dekomprimere kjører man og konvolusjon, men etterfulgt av et upsampling lag, dette laget tar en piksler og gjentar denne i et NxN kvadrat, en kan da gjenta samme prosess som med maxpooling for å dekomprimere bildet.

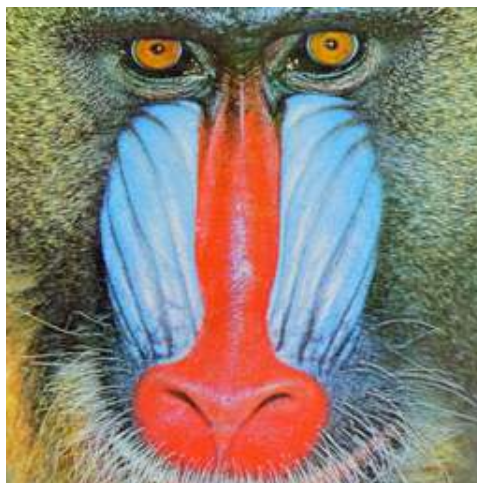
I denne rapporten har jeg brukt en struktur som har en inputstørrelse på 32x32x3, så følger en serie med konvolusjon + maxpooling2D lag til nettverket når en størrelse på 4x4x3, før jeg igjen har en serie med konvolusjon + upsampling2D lag til jeg når samme størrelse som input. Dette betyr da at hvert bilde komprimeres ned til $\frac{4*4*3}{32*32*3} = 1.5621\%$ originalstørrelse før de blir dekomprimert igjen. For å komprimere et helt bilde kan vi kjøre dette nettverket på 32x32 ruter av bildet og så sette de resulterende 4x4 rutene sammen til et komprimert bilde, når vi så skal dekomprimere kan vi kjøre andre halvdel av nettverket på disse 4x4 rutene og sette sammen originalbildet igjen.

3 Resultater

Under vil jeg vise resultatene av de to komprimeringsalgoritmene og sammenligne forskjellige instillinger som kan gjøres.











3.1 Kvantifisering

Kvantifisering er under kjørt på paletter med størrelse 2,4,8,16 og 32. I figur 7, kan en se originalbildet som kvantifiseringen er kjørt på.





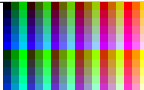
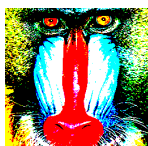



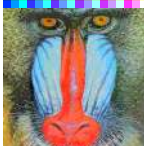


Figur 7: Referansebilde av mandrill brukt i kvantifiseringsalgoritmen











Gråskala kvantifisering:

Antall farger	2	4	8	16	32
Palett					
Resultat					

Uniform RGB kvantifisering:


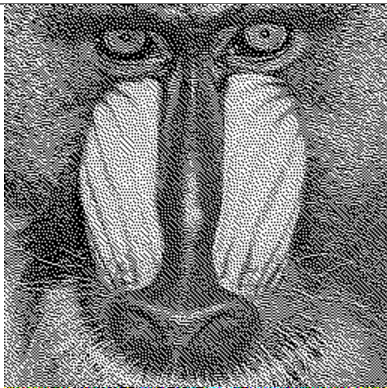




Antall farger	$2^3 = 8$	$3^3 = 27$	$4^3 = 64$	$5^3 = 125$	$6^3 = 216$
Palett					
Resultat					


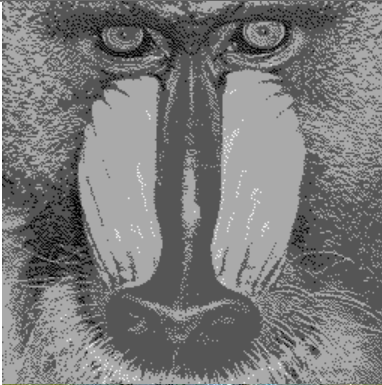

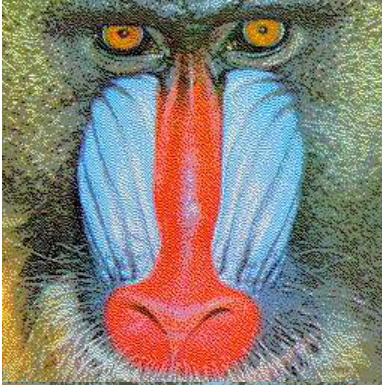


K-means kvantifisering:

Antall farger	2	4	8	16	32
Palett					
Resultat					

For å vise forskjellen dithering gir velger jeg kun å vise forskjellen mellom med og uten dithering på de to minste palett størrelsene vist i tabellene over, da forskjellen dithering har med store paletter er liten.

Da paletten regnes ut hver gang på nytt for k-means vil fargene være forskjellige i sammenligningen, men kvaliteten i hva man kan se av detaljer vil være tilsvarende.

	Uten Dithering	Med Dithering
Gråskala		
Uniform RGB		
K-means		

	Uten Dithering	Med Dithering
Gråskala		
Uniform RGB		
K-means		

3.2 Autoencoder

Under kan en se resultatene fra tre forskjellige autoenkodere som hver komprimerer bildet av forskjellig grad. Autoenkoderne som er valgt komprimerer 32x32 bildet ned til 2x2, 4x4, og 8x8 før de igjen dekomprimerer tilbake til 32x32. Filstørelsen på de komprimerte bildene kan da regnes ut med formelen $\frac{N*N*3}{32*32*3}$ hvor N er størelsen på 32x32 bildet etter komprimering.

Orginalbilde



Autoencoder 2x2
0.390625%



Autoencoder 4x4
1.5625%



Autoencoder 8x8
6.255%



Noe som ikke kommer godt frem i tabellen over er at mellom hvert 32x32 kvadrat er det en ganske skarp kant, denne synes ikke så godt på store bilder da disse kvadratene er relativt små. Om man zoomer inn på 4x4 bildet fra tabellen over kan en se kanten mellom rutene, dette er vist i figur 8.



Figur 8: Mellom 32x32 ruter kan en se en kant