library.uns.ac.id digilib.uns.ac.id

BAB II

TINJAUAN PUSTAKA

2.1. Dasar Teori

2.1.1. Citra Digital

Sebuah citra adalah sebuah fungsi dua dimensi $f(x, y)$ dimana x dan y adalah koordinat bidang spasial dan besar dari (x, y) adalah tingkat intensitas atau *gray level* citra pada titik tersebut (Gonzalez, 2009) . Disaat x, y dan nilai dari f adalah kuantitas diskrit dan terbatas maka citra tersebut dapat disebut sebagai citra digital. Citra yang memiliki ukuran tinggi H dan lebar W dapat dinotasikan pada Persamaan 2.1.

$$f(x, y) = \begin{bmatrix} P_{0,\ 0} & \cdots & P_{0,\ W-1} \\ \vdots & \ddots & \vdots \\ P_{H\ -\ 1,0} & \cdots & P_{H-1,\ W-1} \end{bmatrix} \quad (2.1)$$

Citra digital terdiri dari jumlah elemen yang terbatas dimana tiap-tiapnya memiliki nilai dan posisi. Elemen tersebut disebut *pixel*. *Pixel* adalah istilah yang paling umum digunakan untuk menandakan elemen dalam citra digital. Setiap elemen dari sebuah citra digital mendeskripsikan model warna yang digunakan oleh citra. Komponen dari model warna ini dapat disebut dengan *channel*. Pada citra digital *grayscale*, komponen *pixel* yang berada pada posisi $(x,\ y)$ hanya memiliki satu komponen warna. Sedangkan pada citra digital berwarna yang menggunakan model warna RGB (*Red*, *Green*, *Blue*), komponen *pixel* yang berada pada posisi (x, y) memiliki 3 komponen warna.

2.1.2. Discrete Wavelet Transform

Wavelet transform adalah metode yang digunakan untuk mendekomposisi sinyal menjadi beberapa bagian atau *subband*, output dari metode ini disebut dengan *wavelets* (Gupta and Choubey, 2015). *Discrete wavelet transform* sendiri adalah *wavelet transform* yang mendekomposisi sinyal diskrit menjadi *wavelets* diskrit. *Wavelet transform* menggunakan *low pass filter* dan *high pass filter* yang sudah dibagi *range* nya persis setengah dari *range* minimum dan maksimum dari

library.uns.ac.id

digilib.uns.ac.id

kedua *filter* tersebut. Pertama *low pass filter* digunakan pada tiap data untuk mendapatkan komponen *low frequency*. Hasil output hanya memiliki dari filter ini hanya memiliki setengah dari output aslinya, dan untuk setengahnya akan diisi oleh ditaruh berdampingan output dari *high pass filter* setelah dilakukan pada semua data.

DWT akan menghasilkan *array* 2 dimensi yang memiliki 4 koefisien data yang tiapnya dilabeli dengan LL (low-low), HL(high-low), LH (low-high) dan HH (high-high). Dekomposisi ini dapat dilakukan lagi berulang kali dengan cara yang sama pada LL *band.*

2.1.3. *Bilinear Interpolation*

Bilinear interpolation adalah metode interpolasi pada citra dengan menggunakan 4 titik koordinat terdekat untuk memperkirakan nilai pada suatu posisi (Gonzalez, 2009). Misal ada sebuah titik (x, y) yang akan dicari nilainya menggunakan 4 titik terdekat yaitu $Q(x_1, y_1), Q(x_1, y_2), Q(x_2, y_1),$ dan $Q(x_2, y_2)$ maka *bilinear interpolation* untuk titik tersebut dapat dinyatakan seperti pada Persamaan 2.2.

$$f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{2.2}$$

dimana a_0, a_1, a_2, dan a_3 adalah konstanta yang didapat dari Persamaan 2.3.

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(Q_{11}) \\ f(Q_{12}) \\ f(Q_{21}) \\ f(Q_{22}) \end{bmatrix} \tag{2.3}$$

2.1.4. *Discrete Convolution*

Discrete Convolution merupakan operasi transformasi matriks pada fungsi diskrit. Sebuah data yang tersusun pada grid atau matriks dapat dianggap sebagai fungsi dua dimensi. Sehingga operasi *discrete convolution* pada bidang grid I dengan kernel K berukuran M × N dapat dinyatakan seperti Persamaan 2.4.

commit to user

library.uns.ac.id

digilib.uns.ac.id

$$S(i,j) = (I * K)(i,j) = \sum_{m=-\lfloor M \rfloor}^{\lceil M \rceil - 1} \sum_{n=-\lfloor N \rfloor}^{\lceil N \rceil - 1} I(m,n)K(i-m, j-n) \quad (2.4)$$

dengan notasi $*$ sebagai *discrete convolution* serta M dan N adalah pusat dari *kernel*.

Proses *discrete convolution* tidak memiliki interaksi antar *axis* (Dumoulin and Visin, 2016). Sehingga setiap properti dari *N-dimensional discrete convolution* seperti ukuran *kernel*, *stride*, *padding*, dan *dilasi* untuk setiap *axis* dapat dinotasikan sebagai vektor. Berikut aritmatik pada *discrete convolution*:

1) Dilatasi

Dilatasi menambahkan luas *receptive field* dari *output* tanpa menambah ukuran *kernel* (Yu and K`oltun, 2015). Pada *kernel*, proses dilatasi dilakukan dengan menyisipkan interval antar elemen *kernel*.

2) *Kernel* dan *Stride*

Ukuran dari bidang output hasil proses konvolusi dipengaruhi oleh ukuran *kernel* dan unit *stride* (langkah pergeseran *kernel*). Hubungan antar ukuran bidang *input* i, ukuran bidang *output* o, ukuran *kernel* k dan *stride* s dapat dilihat pada Persamaan 2.5.

$$o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1 \quad (2.5)$$

3) Padding

Proses konvolusi dengan *kernel* berukuran $k > 1$ dan atau ketika *stride* s > 1 dapat menyebabkan *output* tereduksi. Untuk mencegah hal ini maka digunakanlah *Padding*. *Padding* menambah nilai pada pinggiran unit pada *input*. Penambahan nilai oleh proses *padding* dapat dilihat pada Persamaan 2.6.

$$o = \left\lfloor \frac{i+2p-k}{s} \right\rfloor + 1 \quad (2.6)$$

commit to user

library.uns.ac.id

digilib.uns.ac.id

2.1.5. *Deep Feedforward Network*

Deep feedforward network adalah sebuah model *neural network* yang berfungsi untuk mencari fungsi aproksimasi F dari fungsi f dengan melalui proses optimasi *parameter* θ, sehingga didapatkan pemetaan fungsi aproksimasi F untuk *input* x dan *output* y dengan nilai terbaik (Goodfellow et al., 2016).

$$y = f(x) \approx F(x; \theta) \tag{2.7}$$

Model ini disebut *feedforward* karena informasi dari input x mengalir melalui serangkaian fungsi yang akan menghasilkan *output* y. Model ini disebut *network* dikarenakan model ini tersusun dari beragam fungsi yang berbeda. Sebagai contoh, suatu model tersusun atas empat fungsi $F^{(1)}$, $F^{(2)}$, $F^{(3)}$ dan $F^{(4)}$, maka fungsi-fungsi tersebut akan membentuk sebuah fungsi: $F(x; \theta) = F^{(4)}\left(F^{(3)}\left(F^{(2)}\left(F^{(1)}(x; \theta_1); \theta_2\right); \theta_3\right); \theta_4\right)$. $F^{(1)}$ adalah *layer* pertama dari *network*, $F^{(2)}$ bisa disebut dengan *layer* kedua, dan seterusnya. *Layer* pertama disebut dengan *input layer*, *layer* terakhir disebut dengan *output layer* dan *layer-layer* diantara kedua *layer* tersebut disebut *hidden layer*.

2.1.6. Fungsi Aktivasi

Pada *neural network*, *activation function* atau fungsi aktivasi memiliki kegunaan untuk menentukan nilai *output* dari unit komputer pada *neural network*. Fungsi aktivasi *non-linear* memungkinkan *neural network* berperan sebagai *universal function approximator* dengan derajat akurasi yang dapat ditentukan selama tersedia *hidden unit* yang cukup pada *network* (Hornik et al., 1989).

Salah satu fungsi aktivasi yang umum digunakan adalah *Leaky Rectified Linear Unit* (*Leaky*-ReLU). *Leaky*-ReLU adalah fungsi aktivasi *non-linear* yang berfungsi menghasilkan *sparse activation* pada unit *output*. Fungsi aktivasi ini memiliki nilai gradien yang kecil α ketika unit komputasi tidak aktif, sehingga dapat menghindari permasalahan *vanishing gradient*.

$$f(x) = \begin{cases} \alpha x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.8}$$

Pada Persamaan 2.8, nilai α yang umum digunakan adalah 0.001. *Leaky*-ReLU dengan nilai $\alpha = 0$ disebut dengan *Rectified Linear Unit* (ReLU).

library.uns.ac.id

digilib.uns.ac.id

2.1.7. Fungsi *Loss*

Loss function adalah fungsi pengukuran secara objektif yang digunakan untuk mengukur performa *neural network* pada proses *training* (Janocha and Czarnecki, 2017). Dalam pelatihan yang terpantau atau *supervised learning*, untuk memperoleh nilai parameter yang optimal *neural network* perlu meminimalkan nilai selisih antara *output layer* y_{out} dengan target yaitu *ground truth* y_{GT} .

Fungsi dari pengukuran *loss function* dapat dilihat pada Persamaan 2.9.

$$L_2 = \|y_{out} - y_{GT}\|_2^2 \tag{2.9}$$

dimana y_{out} adalah *output* dari *neural network* dan y_{GT} adalah *output* sebenarnya.

2.1.8. *Convolutional Neural Network*

Convolutional Neural Network (CNN), adalah jenis *neural network* yang bermula dari *neocognitron* oleh Kunihiko Fukushima, yang kemudian dikembangkan oleh Yann LeCun (Lecun et al., 1998) dan digunakan untuk memproses data berbentuk grid. Jenis CNN yang digunakan untuk memproses citra adalah *CNN* 2 dimensi, karena citra diproses secara *channel-wise* dan setiap *channel* dari citra merupakan data berbentuk grid dua dimensi. Model *neural network* ini menerapkan arsitektur dengan operasi *discrete convolution* pada *input* data lalu diikuti dengan *activation function*. Proses *feedforward* dari CNN dengan *input* fitur F_{in} dengan dimensi $(C_{\text{in}}, H_{\text{in}}, W_{\text{in}})$ untuk menghasilkan *output* fitur F_{out} dengan dimensi $(C_{out}, H_{out}, W_{out})$ dapat dinotasikan dengan Persamaan 2.10.

$$F_{out\ q} = b_q + \sum_{p=0}^{C_{in}-1} k_{p,q} * F_{in\ p} \tag{2.10}$$

dimana C, H dan W adalah merupakan jumlah fitur yaitu warna pada citra, tinggi dan lebar dari bidang fitur; $q = 0,1,..C_{out}$, δ adalah fungsi aktivasi; k dan b adalah parameter *weight* atau *kernel matrix* dan *bias* dari model *neural network* yang memiliki susunan sebagai *matrix* berukuran $C_{out} \times C_{in} \times M \times N$ dan $C_{out} \times 1 \times 1$. Pada *input layer*, citra dengan dimensi (H, W, C), dimana C menyatakan jumlah fitur warna dari citra dilakukan pergeseran *axis* pertama dengan terakhir, sehingga didapatkan citra berdimensi (C, H, W).

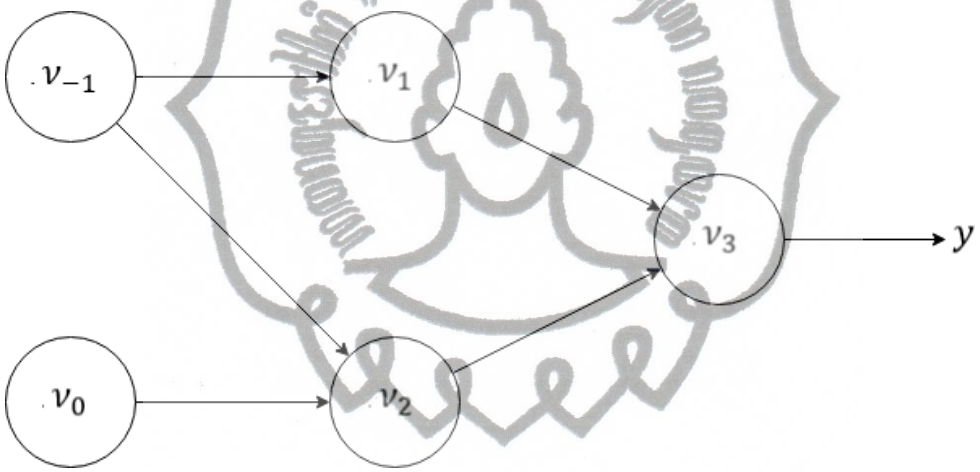
commit to user

library.uns.ac.id

digilib.uns.ac.id

2.1.9. *Automatic Differentiation*

Automatic Differentiation adalah sebuah metode mengevaluasi derivasi dari sebuah fungsi. Metode ini menggabungkan setiap perhitungan derivatif dari komponen penyusun fungsi menggunakan aturan rantai. Proses *automatic differentiation* terdiri dari 2 fase, yaitu fase *forward accumulation*, dan fase *reverse accumulation*. *Forward accumulation* akan menghasilkan sebuah *graph* komputasi dan himpunan *intermediate variable* v_i. Lalu *reverse accumulation* akan dilakukan perhitungan nilai derivatif dengan melakukan perambatan v_i dari *output* ke *input* Sebagai contoh $f(x_1 x_2) = \frac{x_1}{x_2} + \sin x_1$, maka *graph* komputasi yang didapat dapat dilihat pada Gambar 2.1.



Gambar 2.1 *Graph* komputasi dari fungs $y = \frac{x_1}{x_2} + \sin x_1$

commit to user

library.uns.ac.id

digilib.uns.ac.id

Melalui proses *forward accumulation* seperti pada Tabel 2.1 didapatkan nilai y.

Tabel 2.1 Proses *forward accumulation*

Forward Accumulation		
v_{-1}	$= x_1$	$= 1$
v_0	$= x_2$	$= 2$
v_1	$= sin\ v_{-1}$	$= 0.841$
v_2	$= \frac{v_{-1}}{v_0}$	$= \frac{1}{2}$
v_3	$= v_1 + v_2$	$= \frac{1}{2} + 0.841$
y	$= v_3$	$= 1.341$

Melalui proses *reverse accumulation* seperti pada Tabel 2.2, diperoleh nilai y.

Tabel 2.2 Proses *reverse accumulation*

Reverse Accumulation			
x_1	$= v_{-1}$		
x_2	$= v_0$		
v_{-1}	$= v_{-1} + v_1 \frac{\partial v_1}{\partial v_{-1}}$	$= v_{-1} + v_1 \times \cos v_{-1}$	
v_{-1}	$= v_2 \frac{\partial v_2}{\partial v_{-1}}$	$= v_2 \times v_{-1} \frac{-1}{x^2}$	$= \frac{1}{2}$
v_0	$= v_0 + v_2 \frac{\partial v_2}{\partial v_0}$	$= v_0 + v_2 \times v_{-1} \frac{-1}{{v_0}^2}$	$= 1$
v_1	$= v_3 \frac{\partial v_3}{\partial v_1}$	$= v_3 \times 1$	$= 1$
v_2	$= v_3 \frac{\partial v_3}{\partial v_2}$	$= v_3 \times 1$	$= 1$
v_3	$= y$	$= 1$	$= 1$

commit to user

library.uns.ac.id

digilib.uns.ac.id

2.1.10. *Back-Propagation*

Back-Propagation adalah salah satu metode yang digunakan melakukan *training* pada *neural network* (Rumelhart et al., 1986). Algoritma memiliki tiga fase yaitu perambatan maju, perambatan balik dan pembaharuan parameter dari *neural network*. Tiga fase tersebut akan terus dilakukan berulang kali hingga didapatkan nilai parameter yang optimal.

Fase perambatan maju digunakan untuk mendapatkan nilai *loss function* dari *neural network*. Nilai tersebut kemudian dikembalikan dari *output layer* ke *input layer*. Pada perambatan balik nilai *loss function* digunakan untuk menghitung nilai gradien *loss function* terhadap nilai parameter dari setiap *layer* pada *neural network*.

2.1.11. *Adaptive Momentum Estimation*

Adaptive Momentum Estimation (Adam) adalah sebuah algoritma yang digunakan untuk mengoptimasi parameter pada *neural network* yang dikembangkan oleh (Kingma and Ba, 2014). Metode ini memperbarui paramater dengan cara memperbaiki momentum hasil estimasi sehingga mempercepat laju konvergensi. Langkah-langkah dari algoritma ini dijelaskan pada Algorithm 2.1.

Algorithm 2.1: *Adaptive Momentum Estimation* (Adam)

Require: α: *Stepsize*

Require: $\beta_1, \beta_2 \in [0,1)$: Laju *exponential decay* untuk estimasi momentum

Require: ϵ: Nilai konstanta yang sangat kecil untuk stabilitas perhitungan

Require: $f(\theta)$: Fungsi objektif stokastik dengan *parameter* θ

Require: θ_0: *Parameter* awal

$m_0 \leftarrow 0$ (Inisiasi momentum orde pertama)

$v_0 \leftarrow 0$ (Inisiasi momentum orde kedua)

$t \leftarrow 0$ (Inisiasi *timestep*)

while θ_t belum konvergen ***do***

$t \leftarrow t + 1$

---

commit to user

library.uns.ac.id

digilib.uns.ac.id

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Hitung *gradient* fungsi objektif stokastik terhadap *parameter* θ)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1+\beta_1) \cdot g_t$ (Perbarui estimasi momentum orde pertama)

$v_t \leftarrow \beta_1 \cdot v_{t-1} + (1+\beta_2) \cdot g_t^2$ (Perbarui estimasi momentum orde kedua)

$\widehat{m}_t \leftarrow \frac{m_t}{(1+\beta_1^t)}$ (Perbaiki estimasi momentum orde pertama)

$\widehat{v}_t \leftarrow \frac{v_t}{(1+\beta_2^t)}$ (Perbaiki estimasi momentum orde kedua)

$\theta_t \leftarrow \theta_{(t-1)} - \alpha \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}}$ (Perbarui parameter)

end while

return θ_t (*Parameter* hasil optimasi)

dengan nilai *hyperparameter* yang umum digunakan $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\alpha = 0.001$, dan $\epsilon = 10^{-8}$.

2.1.12. *Full Reference Image Quality Assesment*

Image Quality Assessment (IQA) merupakan rancangan model matematika yang mampu memprediksi kualitas citra secara otomatis dan akurat dengan meniru kualitas prediksi dari pengamatan manusia rata-rata (Mohammadi et al., 2014). Sedangkan *Full Reference Image Quality Assessment* (FR-IQA) adalah metode penilaian kualitas citra di mana citra referensi dengan kualitas sempurna tersedia.

1. Peak Signal Noise Ratio

Peak Signal Noise Ratio (PSNR) adalah metode yang digunakan untuk menghitung rasio kekuatan maksimum dari sinyal dan kekuatan distorsi. PSNR dapat diperoleh dengan Persamaan 2.11.

$$PNSR = 10\ log\left(\frac{A^2}{MSE}\right) dB \tag{2.11}$$

commit to user

library.uns.ac.id

digilib.uns.ac.id

dimana A adalah nilai maksimum dari intensitas citra, dalam hal ini dengan 8 bit/pixel maka nilai A = 255. Kemudian MSE yaitu *mean square error* didapat dengan Persamaan 2.12.

$$MSE = \frac{1}{W \times H} \sum_{j=0}^{H-1} \sum_{i=0}^{W-1} \left(I_{ref}(i,j) - I_{test}(i,j) \right)^2 \quad (2.12)$$

dimana H & W adalah panjang dan lebar dari citra , $I_{ref}(i, j)$ adalah nilai *pixel* citra referensi pada posisi (i, j)dan $I_{test}(i, j)$ adalah nilai *pixel* citra *test* pada posisi (i, j)

2. *Structural Similarity Index* (SSIM)

mengatakan jika hubungan antara setiap pixel pada citra dapat membawa informasi yang berguna pada struktur citra. Sehingga metode yang dapat mengukur informasi struktur pada citra dapat memberikan perkiraan distorsi pada citra. Hal ini lah yang mendasari metode SSIM. Perhitungan SSIM pada sebuah citra secara garis besar dibagi menjadi tiga tahap.

Tahap pertama adalah perbandingan channel luminance pada citra. Perbandingan channel luminance dapat diformulasikan dengan Persamaan 2.13.

$$l(I_{ref}, I_{test}) = \frac{2\mu_{ref}\mu_{test} + T_1}{\mu_{ref}^2 + \mu_{test}^2 + T_1} \quad (2.13)$$

dimana μ adalah intensitas kecerahan pada citra yang dapat diformulasikan dengan Persamaan 2.14.

$$\mu = \frac{1}{W \times H} \sum_{j=0}^{H-1} \sum_{i=0}^{W-1} I(i,j) \quad (2.14)$$

dan T_1 adalah konstanta yang dapat diformulasikan dengan Persamaan 2.15.

$$T_1 = (\epsilon_1 D)^2 \quad (2.15)$$

Kedua, membandingkan kontras antara kedua citra. Fungsi untuk membandingkan kontras antara kedua citra dapat dilihat pada Persamaan 2.16.

commit to user

library.uns.ac.id

digilib.uns.ac.id

$$c(I_{ref}, I_{test}) = \frac{2\sigma_{ref}\sigma_{test} + T_2}{\sigma_{ref}^2 + \sigma_{test}^2 + T_2} \quad (2.16)$$

dimana T_2 adalah konstanta untuk stabilitas perhitungan yang diformulasikan dengan Persamaan 2.17.

$$T_2 = (\epsilon_2 D)^2 \quad (2.17)$$

Dibutuhkan juga nilai standar deviasi dari citra yang dapat dilihat pada Persamaan 2.18.

$$\sigma = \sqrt{\frac{1}{W \times H - 1} \sum_{j=0}^{H-1} \sum_{i=0}^{W-1} (I(i,j) - \mu)^2} \quad (2.18)$$

Dan tahap ketiga adalah perbandingan struktur citra, yang dapat dinyatakan dengan Persamaan 2.19.

$$s(I_{ref}, I_{test}) = \frac{\sigma_{ref,test} + T_3}{\sigma_{ref}\sigma_{test} + T_3} \quad (2.19)$$

dimana $\sigma_{ref,test}$ adalah nilai korelasi dari kedua citra, yang dapat diformulasikan dengan Persamaan 2.20.

$$\sigma_{ref,test} = \frac{1}{W \times H} \sum_{j=0}^{H-1} \sum_{i=0}^{W-1} (I_{ref}(i,j) - \mu_{ref})(I_{test}(i,j) - \mu_{test}) \quad (2.20)$$

dan T_3 adalah konstanta yang dapat dinyatakan dengan Persamaan 2.21.

$$T_3 = (\epsilon_3 D)^2 \quad (2.21)$$

Pada akhirnya *Structural Similarity Index* dapat diformulasikan dengan Persamaan 2.22.

$$SSIM(I_{ref}, I_{test}) = l(I_{ref}, I_{test})^{\alpha} \cdot c(I_{ref}, I_{test})^{\beta} \cdot s(I_{ref}, I_{test})^{\gamma} \quad (2.22)$$

dimana nilai $\alpha = \beta = \gamma = 1, D = 255, \epsilon_1 = 0.01, \epsilon_2 = 0.03,$ dan ϵ_3 diberi nilai sedemikian hingga nilai T_3 adalah $\frac{T_2}{2}$.

commit to user

library.uns.ac.id

digilib.uns.ac.id

2.2. Penelitian Terkait

Tabel 2.3 memuat rangkuman mengenai penelitian yang terkait pada pengerjaan tugas akhir ini. Berikut beberapa penelitian yang berkaitan dengan penelitian yang diajukan:

1. **Color to Gray and Back: Color Embedding Into Textured Gray Images**

   Pada penelitian ini, konsep *color-embedding-grayscale image* dikembangkan. Konsep *color embedding* yang dikembangkan memungkinkan sebuah citra abu-abu/*grayscale* dikembalikan menjadi citra berwarna. *color embedding* mengimplementasikan *wavelet transform* untuk menyimpan informasi warna pada citra. Dengan menggunakan *channel* Y pada sebuah citra YcbCr sebagai representasi dari citra *grayscale*, *wavelet* mentransformasikan citra tersebut menjadi *subband* sinyal. Informasi warna pada citra yaitu Cb dan Cr disisipkan pada *subband* tersebut. Dengan mengambil informasi warna pada *subband channel* Y maka mengembalikan warna pada citra tersebut dapat dilakukan. Pada penilitian ini dilakukan reproduksi *subband*. Berikut langkah-langkah dari proses *color embedding* dan *color recovery*.

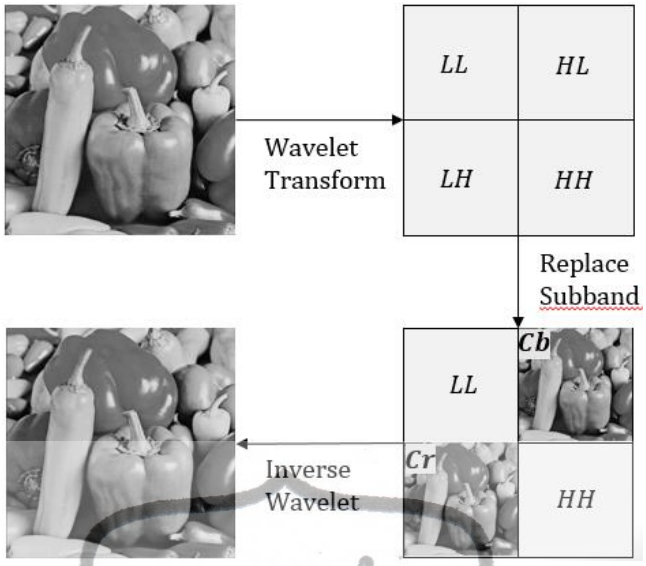
   Color Embedding:

   1) Konversikan citra RGB menjadi Y, Cb, Cr
   2) Gunakan DWT 1 *level* pada *channel* Y sehingga Y dibagi menjadi empat *subband*: Y $\rightarrow (LL, LH, HL, HH)$.
   3) Reduksi Cb dan Cr sehingga ukurannya menjadi 0.5 dari ukuran semula.
   4) Substitusi *subband* sebagai berikut:
      $LH \leftarrow Cb;\ HL \leftarrow Cr$
   5) Lakukan DWT terbalik untuk mendapatkan citra derajat abu
      $(LL, Cb, Cr, HH) \rightarrow Y^1$.
   6) Citra Y^1 adalah citra derajat abu yang didapat.

commit to user

library.uns.ac.id

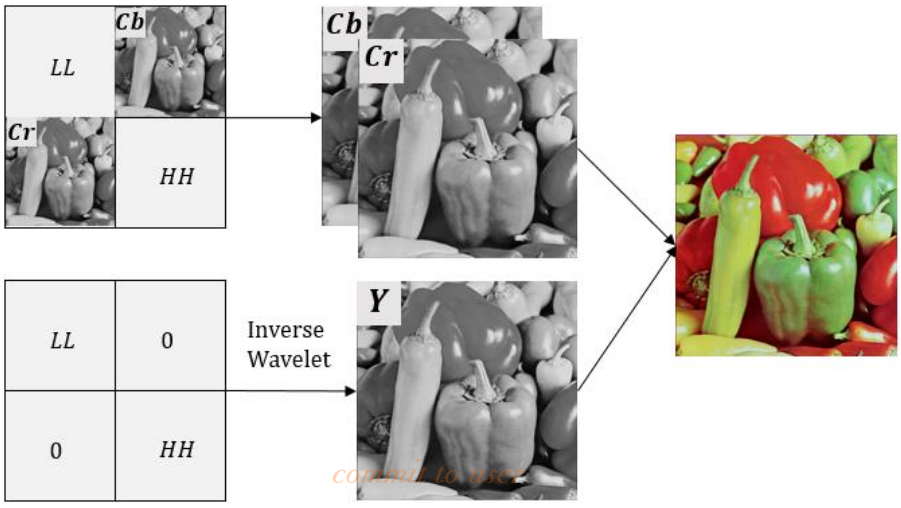
digilib.uns.ac.id



Gambar 2.2 Skema proses *color embedding* (de Queiroz and Braun 2006)

Color Recovery:

1) Lakukan DWT pada citra *grayscale* untuk dikonversi menjadi *subband* $Y^1 \rightarrow (LL, Cb, Cr, HH)$.
2) Dilakukan interpolasi pada Cb dan Cr untuk mengembalikan ukurannya menjadi 2 kali lebih besar.
3) Substitusi *subband* yang disisipkan pada citra dengan nilai nol, lalu lakukan DWT terbalik untuk mendapatkan Y
   $(LL, Cb, Cr, HH) \rightarrow (LL, 0,0, HH)$
   $(LL, 0,0, HH) \rightarrow Y$.
4) Konversi citra tersebut dari YcbCr kembali ke RGB.



Gambar 2.3 Skema proses *color recovery* (de Queiroz and Braun 2006)

library.uns.ac.id

digilib.uns.ac.id

Metode ini berhasil mengembalikan warna pada citra *color-embedded-grayscale* dan meningkatkannya dengan melakukan level 2 DWT untuk mensubstitusi nilai *subband* yang hilang. Berikut salah satu citra hasil dari penelitian ini setelah dilakukan *error diffusion halftoning* dan *scaling* dengan berbagai skala faktor K.

Gambar 2.4 Citra hasil *color recovery* setelah dilakukan *error diffusion halftoning* dan *scaling*. (de Queiroz and Braun 2006)
Dari kiri ke kanan, $K = 1, K = 4, K = 8$.

2. Swarm Intelligence on Color-Embedded-Grayscale Image

Pada penelitian ini, metode *Particle Swarm Optimazion* digunakan pada proses restorasi warna pada citra *color embedding*. PSO sendiri adalah metode optimasi yang menirukan tingkah laku kelompok hewan seperti burung atau ikan, untuk menemukan nilai paling optimum dari sebuah fungsi. PSO membuat beberapa partikel sebagai kandidat solusi. Partikel tersebut memiliki nilai yang akan diubah tiap iterasi berdasarkan dari *fitness value*. Tiap partikel diinisiasi dengan nilai random yang *uniform* dalam sebuah *range* yang ditentukan, kemudian PSO mencari partikel yang optimum x^* . Hal ini dapat dirumuskan dengan masalah optimasi menjadi:

$$\min_{x^*} f(x) \tag{2.23}$$

dimana x adalah *fitness value*

Tiap partikel merekam posisi $\boldsymbol{x_i}$, kecepatan $\boldsymbol{v_i}$, posisi terbaik $\boldsymbol{p_i}$ dan posisi terbaik global $\boldsymbol{p_g}$. Pergerakan tiap partikel ditentukan sebagai penambahan

library.uns.ac.id

digilib.uns.ac.id

antara posisi dan kecepatan pada iterasi tersebut. Pergerakan ini bisa diformulakan sebagai berikut:

$$x_i(t+1) \leftarrow x_i(t) + v_i(t) \tag{2.24}$$

dimana $\boldsymbol{t}$ adalah jumlah iterasi. Setelah beberapa iterasi PSO akan terpusat menjadi optima global atau lokal. Posisi terakhir dari *swarm* $\boldsymbol{x_i}$ pada iterasi terakhir dianggap sebagai hasil optimum $\boldsymbol{x}^*$.

Beberapa penelitian telah dilakukan untuk membuat kecepatan *swarm* yang paling cocok. Berikut beberapa formula kecepatan yang dipakai dalam penelitian ini.

1) **Classical Particle Swarm Optimation (CPSO)**

   Metode ini menggunakan formula kecepatan sebagai berikut:

   $$v_i(t+1) = wv_t(t) + c_1 r_1[\boldsymbol{p}_i - \boldsymbol{x}_i(t)], \tag{2.25}$$

   dimana w , c_1 , dan c_2 adalah bobot inersia, konstanta kognisi individu, dan konstanta kognisi sosial. Simbol r_1 dan r_2 adalah bilangan acak *uniform* dengan *range* [0,1]. Nilai w biasanya diatur dengan $w = 0.9$.

2) **Linearly Decreased Weight Particle Swarm Optimization (LDWPSO)**

   Metode ini mengubah nilai bobot inersia berdasarkan iterasi. Nilai inersia yang diberikan sangat besar di iterasi awal, kemudian berkurang secara bertahap seiring dengan bertambahnya iterasi. Bobot inersia ini dapat dikomputasikan menjadi:

   $$w_t = w_2 - \frac{t}{T}(w_2 - w_1), \tag{2.26}$$

   dimana w_1 dan w_2 adalah batas bawah dan batas dari bobot inersia. Kedua nilai tersebut umumnya digunakan dengan nilai $w_1 = 0.4$ dan $w_2 = 0.9$. Komputasi kecepatan ini dapat dikomputasi lebih jauh sebagaimana pada Persamaan 2.7

   $$w_t = w_2 - \frac{t}{T}(w_2 - w_1), \tag{2.27}$$

   dimana T adalah iterasi maksimum untuk optimasi PSO.

3) **Constriction Particle Swarm Optimization (CoPSO)**

   Metode ini bertujuan untuk mengurangi jumlah *parameter* yang digunakan pada proses *training*. Di sini digunakan *parameter constriction*

library.uns.ac.id

digilib.uns.ac.id

K sebagai fungsi kombinasi antar c_1 dan c_2. Selain itu, bobot dari inersia dihindari sebagaimana yang diformulasikan sebagai berikut:

$$v_i(t+1) = K\{v_i(t) + c_1 r_1[p_i - x_i(t)] + c_2 r_2[p_g - x_i(t)]\}, \quad (2.28)$$

dimana $K = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$ dengan mengatur nilai φ sebagai $\varphi = c_1 + c_2 > 4$. Sebagai contoh, jika digunakan $c_1 = c_2 = 2.05$, maka didapat $\varphi = 4.1$. Dari situ didapat $K = 0.729$.

4) Absolute Gaussian Particle Swarm Optimization (AGPSO)

Metode ini didasarkan pada observasi pada perubahan sebelumnya. Pada beberapa penelitian umumnya sering digunakan $w = 0.6$ dan $c_1 = c_2 = 1.7$. Hal ini berarti koefisien *stochastic* dari $\boldsymbol{p_i} - \boldsymbol{x_i}(t)$ dan $\boldsymbol{p_g} - \boldsymbol{x_i}(t)$ berubah menjadi $c_1 r_1 = 0.85$ dan $c_2 r_2 = 0.85$. Sehingga, nilai absolut dari distribusi probabilitas *Gaussian* memberikan kandidat yang bagus untuk menghasilkan bilangan acak. Hal ini dapat disimpulkan sebagai berikut:

$$v_i(t+1) = |\mathcal{N}(0,1)| * [\boldsymbol{p_i} - \boldsymbol{x_i}(t)] + |\mathcal{N}(0,1)| * [p_g - x_i(t)], \quad (2.29)$$

dimana $|\mathcal{N}(0,1)|$ adalah nilai absolut dari bilangan acak yang dihasilkan oleh *Gaussian* dalam *range* $[0,1]$. Metode ini menghindari beberapa *parameter* untuk PSO.

Terdapat informasi yang hilang saat dilakukan *color embedding* yang membuat citra menjadi terdistorsi. Penelitian ini menggunakan PSO meningkatkan kualitas dari citra tersebut sehingga kualitasnya lebih mirip dengan citra aslinya. Peningkatan ini dapat dilakukan dengan mencari bobot atau nilai optimum yang akan digunakan untuk menyesuaikan citra hasil restorasi. Proses ini dapat dinyatakan sebagai:

$$\hat{I}_p \Leftarrow \alpha^* \hat{I}, \quad (2.30)$$

dimana $\hat{I}_p$ adalah citra hasil restorasi yang telah ditingkatkan dengan metode diajukan dan α^* adalah bobot optimum. Nilai α^* didapat dengan menggunakan PSO untuk mendapatkan nilai optimum.

commit to user

library.uns.ac.id

digilib.uns.ac.id

Proses optimisasi dari PSO dapat dinyatakan sebagai:

$$\min_{\alpha^*} \left\| A^i - \alpha R^i \right\|_2^2 , \tag{2.31}$$

dimana $\|\cdot\|_2^2$ adalah Euclidian norm, A^i adalah training data dari citra asli, dan R^i adalah citra hasil restorasi.

Secara kualitatif, metode PSO memperbaiki informasi warna dan tingkat kecerahan pada citra. Hasil citra memiliki warna dan tingkat kemiripan yang lebi tinggi dibandingkan dengan metode sebelumnya. Dilakukan 4 jenis PSO pada citra dan metode Constriction PSO merupakan metode terbaik secara kuantitatif dengan hasil terbaik dan standar deviasi yang dapat diterima.

Gambar 2.5 Citra hasil *color recovery* pada *testing image* (Prasetyo, 2019) (a, d) citra hasil restorasi dari citra *color-embedded-grayscale*, (b, e) citra hasil restorasi dari metode PSO, dan (c, f) adalah citra asli.

3. Color-Embedded-Grayscale Quality Enhancement Using Deep Convolutional Network

commit to user

Pada penilitian ini digunakan metode *deep learning* yang berhasil melakukan *color recovery* pada citra *color embedded*. Berbeda dengan metode

library.uns.ac.id

digilib.uns.ac.id

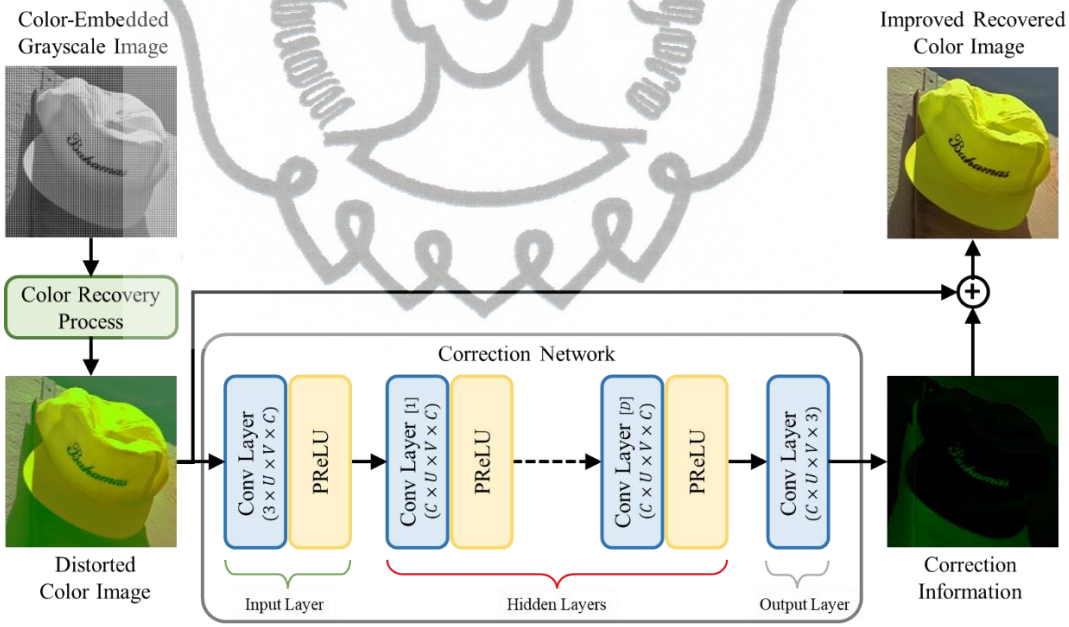
sebelum-sebelumnya dimana metode tersebut mencoba menyesuaikan komponen warna pada citra, metode ini memperbaiki kesalahan informasi pada citra *embedded* yang sudah direstorasi. Informasi pengoreksian ini adalah bagian pelengkap dari citra terdistorsi yang dipresentasikan dengan bentuk *residual*. Hal ini berarti bahwa kualitas citra restorasi bisa ditingkatkan dengan Persamaan 2.30.

$$\hat{I} = \tilde{I}^* + \tilde{I}_{res} \tag{2.32}$$

dimana $\tilde{I}_{res}$ adalah komplemen dari citra terdistorsi. Informasi ini akan diambil dari hasil *Correction Network* (CorNet) dibawah *deep convolutional networks framework*. Aturan tersebut dapat ditulis dengan Persamaan 2.31.

$$\hat{I} = \tilde{I}^* + Cor(\tilde{I}^*, \Theta) \tag{2.33}$$

dimana $Cor(\cdot)$ dan Θ adalah CorNet dan *parameter* pada semua *layer* secara urut.



Gambar 2.6 *Correction Network* (CorNet) (Bagaskara, 2020)

Gambar 2.7 mengilustrasikan skema diagram dari arsitektur CorNet. CorNet terdiri dari 3 bagian yaitu *input layer*, *hidden layers*, dan *output layer*. *Input layer* merubah citra hasil restorasi warna yang terdistorsi yang berukuran $H \times W \times 3$ menjadi *feature maps* yang berukuran $H \times W \times N$. *Feature maps* tersebut akan dilanjutkan ke fungsi aktivasi PReLU yang dapat dilihat pada Persamaan 2.32.

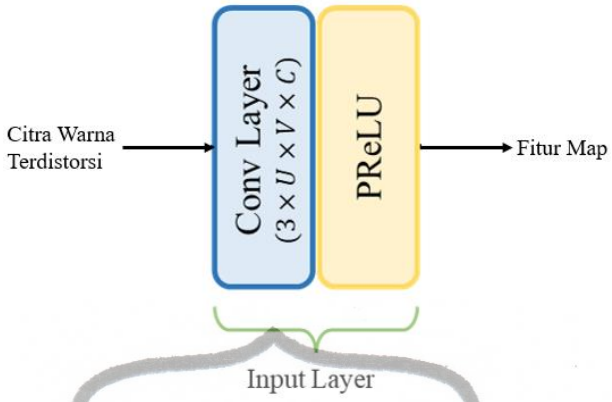
commit to user

$$\boldsymbol{x}_{in} = \delta(\tilde{I}^* * \boldsymbol{w}_{in} + \boldsymbol{b}_{in}) \tag{2.34}$$

library.uns.ac.id

digilib.uns.ac.id

dimana $\boldsymbol{w}_{in}$, $\boldsymbol{b}_{in}$ dan $\boldsymbol{x}_{in}$ adalah berat, bias, dan *feature maps* dari *input layer* (Gambar 2.8).

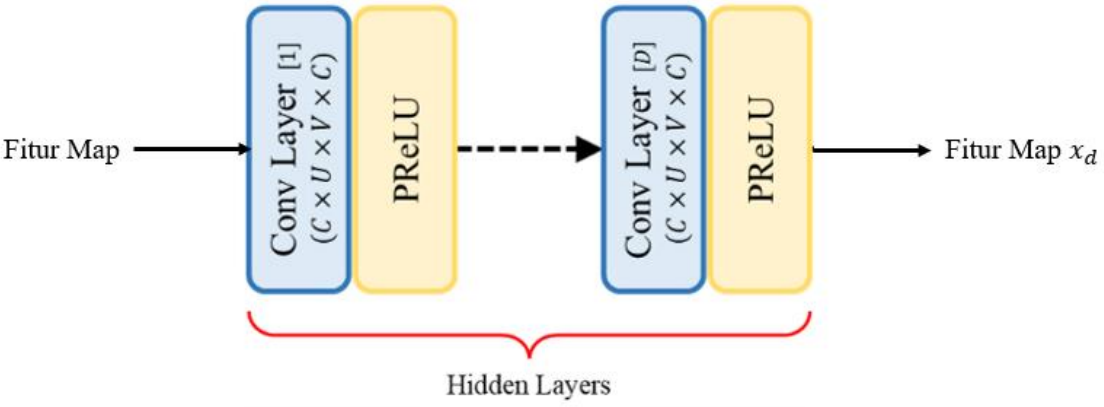


Gambar 2.7 *Input layer* pada *CorNet* (Bagaskara, 2020)

Feature maps pada $\boldsymbol{x}_{in}$ diteruskan melalui beberapa seri *hidden layer* yang serupa. Tiap *hidden layer* dibentuk oleh D *layer* (*Conv Layer*) diikuti oleh PReLU (Gambar 2.8). Proses *feedforward* dari *hidden layer* dapat diformulasikan dengan Persamaan 2.33.

$$\boldsymbol{x}_{d+1} = \delta(\boldsymbol{x}_d * \boldsymbol{w}_d + \boldsymbol{b}_d) \;\; \forall d = 1,2,\dots,D, \tag{2.35}$$

dimana $\boldsymbol{x}_d \in \mathbb{R}^{H \times W \times N}$ adalah *feature maps* yang diproduksi di layer ke-$\boldsymbol{d}$ pada *hidden layer*. $\boldsymbol{x}_1 = \boldsymbol{x}_{in}$ sebagai kondisi permulaan.



Gambar 2.8 *Hidden layer* pada *Cornet* (Bagaskara, 2020)

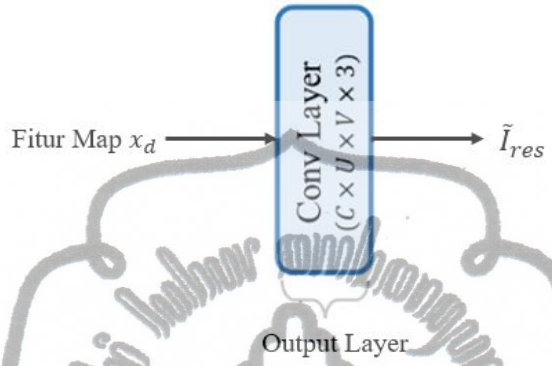
commit to user

library.uns.ac.id

digilib.uns.ac.id

Pada *layer* terakhir, *output layer* mengonversi *feature maps* $\boldsymbol{x}_{D+1}$ menjadi $\tilde{I}_{res} \in \mathbb{R}^{H \times W \times 3}$ (Gambar 2.9). Layer ini terdiri dari CNN tanpa aktivasi yang dapat dilihat pada Persamaan 2.34.

$$\tilde{I}_{res} = \boldsymbol{x}_{D+1} * \boldsymbol{w}_{out} + \boldsymbol{b}_{out} \tag{2.36}$$



Gambar 2.9 *Output layer* pada *CorNet* (Bagaskara, 2020)

Hasil akhir yaitu $\tilde{I}_{res}$ adalah komplemen dari citra terdistorsi yang dibutuhkan dalam komputasi untuk memperoleh citra restorasi warna yang lebih baik.

Metode ini berhasil mengembalikan warna pada citra *color embedded* dengan hasil PSNR dan SSIM terbaik dibandingkan dengan metode-metode sebelumnya.

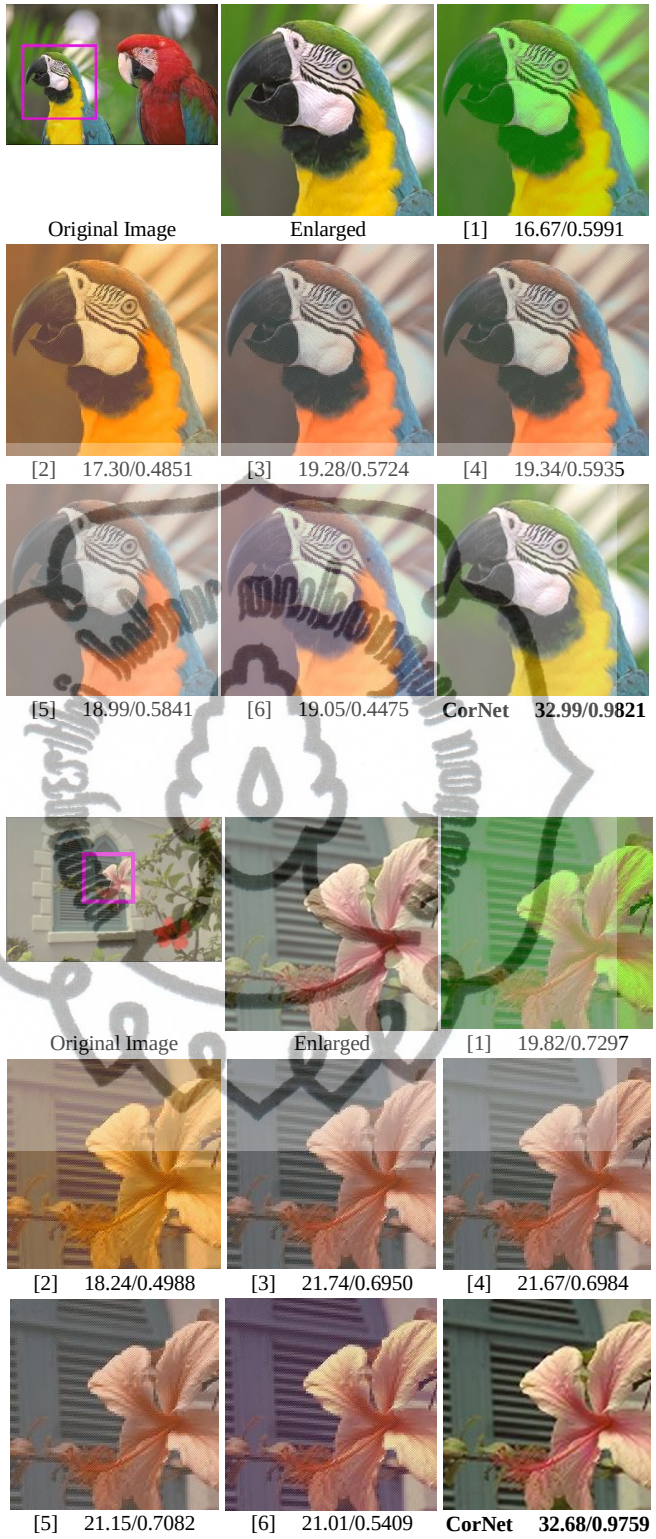
Tabel 2.3 Hasil rata-rata PSNR dan SSIM

Methods	PSNR	SSIM
Former Scheme [1]	16.70	0.5132
Classical PSO [2]	17.23	0.3863
LDWPSO [3]	20.35	0.6151
CoPSO [4]	20.34	0.6246
AGPSO [5]	19.98	0.6226
Pseudo Inverse [6]	20.31	0.5296
CorNet ($\gamma = 0.50$)	**29.61**	**0.9320**

commit to user

library.uns.ac.id

digilib.uns.ac.id



Gambar 2.10 Perbandingan citra hasil restorasi warna tiap metode pada penelitian Bagaskara (2020)

Pada halaman selanjutnya adalah tabel rangkuman dari penelitian-penilitian terkait yang sudah disebutkan dan dijelaskan sebelum ini.

commit to user

library.uns.ac.id digilib.uns.ac.id

Tabel 2.3 Rangkuman Penelitian Terkait

NO	JUDUL JURNAL , SITASI	MASALAH	TUJUAN	METODE	HASIL	KETERKAITAN
1.	*Color to Gray and Back: Color Embedding Into Textured Gray Images* (de Queiroz and Braun 2006)	Cara merubah sebuah citra berwarna menjadi citra hitam-putih dan kembali menjadi berwarna. Citra hitam-putih yang sudah di*print* juga harus bisa dikembalikan ke citra berwarna saat di*scan*	Merubah citra warna menjadi hitam-putih dan kembali menjadi berwarna bahkan setelah citra sudah dipindahkan scara fisik (kertas hasil *print* citra)	Menggunakan *discrete wavelet transform* untuk merubah citra menjadi hitam-putih sekaligus menyimpan informasi warna pada citra. Dilakukan proses sebelumnya secara terbalik untuk mengembalikan citra menjadi berwarna. Dilakukan penyesuaian pada citra seperti *scaling* dan *halftoning* agar citra bisa semirip mungkin dengan aslinya.	Metode ini berhasil merubah sebuah citra berwarna menjadi hitam-putih dan diubah menjadi berwarna kembali. Hasil citra secara kuantitatif (PSNR) juga meningkat seiring dengan diaikkannya faktor K yang digunakan pada saat proses *scaling* dan *halftoning*	Mengembangkan metode restorasi warna citra dari hasil citra *color-embedding*

library.uns.ac.id digilib.uns.ac.id

Tabel 2.3 Lanjutan

NO	JUDUL JURNAL , SITASI	MASALAH	TUJUAN	METODE	HASIL	KETERKAITAN
2.	*Swarm Intelligence on Colo-Embedded-Grayscale Image* (Prasetyo, 2019)	Mengimplementasikan *Particle Swarm Optimization* (*PSO*) pada proses *color recovery* yang menggunakan *discrete wavelet transform*	Memperbaiki metode *color recovery* terdahulu sehingga menghasilkan kualitas citra yang lebih baik dengan menambahkan metode *PSO*	Pada proses pengembalian warna dari citra *grayscale* ke berwarna ada nilai *subband* atau potong sinyal yang hilang. Pada metode sebelumnya, nilai yang hilang ini hanya diganti dengan nilai 0. Dengan PSO akan diprediksi nilai *subband* yang hilang tersebut agar hasil restorasi warna pada citra bisa semirip mungkin dengan aslinya.	Secara kualitatif, metode PSO memperbaiki informasi warna dan tingkat kecerahan pada citra. Hasil citra memiliki warna dan tingkat kemiripan yang lebi tinggi dibandingkan dengan metode sebelumnya. Dilakukan 4 jenis PSO pada citra dan metode CoPSO merupakan metode terbaik secara kuantitatif dengan hasil terbaik dan stand deviasi yang dapat diterima.	Mengembangkan metode restorasi warna citra dari hasil citra *color-embedding*