Image Colorization Using CNN

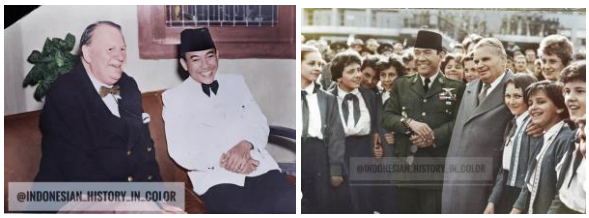
Roby Purnomo (*13520106*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520106@std.stei.itb.ac.id

***Abstract*—*Image Colorization* adalah sebuah task dari *Image Processing* yang digunakan untuk mewarnai sebuah gambar yang awalnya adalah *grayscale* menjadi gambar yang memiliki warna. Salah satu metode yang dapat digunakan untuk *Image Colorization* adalah CNN (*Convolutional Neural Network*) yakni dengan melatih model menggunakan dataset yang besar sehingga informasi polanya tersimpan. Hasil model yang dibuat dari penelitian ini memiliki hasil yang lumayan untuk mewarnai berbagai gambar bersejarah (*historical image*).**

Keywords—Image Colorization; CNN; grayscale; Image Processing; historical image

I. PENDAHULUAN

Pada zaman yang sudah maju ini, ada banyak hal yang dulunya hanya menjadi angan-angan saja tetapi sekarang dapat berubah menjadi kenyataan. Hal ini tidak lain dan tidak bukan dikarenakan pesatnya teknologi yang sudah berkembang di masa kini. Salah satu angan-angan yang penulis fokuskan disini adalah pewarnaan gambar-gambar zaman dahulu yang mayoritas memiliki kualitas rendah dan hanya memiliki format *grayscale* atau gambar skala abu-abu saja. Namun, hal ini dapat diatasi di zaman ini dengan melakukan salah satu *task* dari *Image Processing* yakni *Image Colorization*.



(sumber: Indonesian history in color (@indonesian_history_in_color))

Image Colorization adalah proses pemberian warna pada gambar skala abu-abu untuk membuatnya lebih menarik secara estetis dan bermakna secara perseptual. *Task* ini diketahui sebagai tugas yang rumit yang sering memerlukan pengetahuan sebelumnya mengenai konten dalam gambar tersebut dan harusnya penyesuaian secara manual untuk mencapai kualitas yang bagus. Selain itu, karena objek dapat memiliki warna yang berbeda, ada banyak cara yang mungkin untuk memberikan warna pada piksel dalam sebuah gambar, yang berarti tidak ada solusi yang pasti untuk masalah ini.

Ada dua pendekatan utama untuk image colorization: pertama memerlukan pengguna untuk memberikan warna pada beberapa wilayah dan memperluas informasi warna tersebut ke seluruh gambar, dan yang kedua mencoba untuk mempelajari (*learning*) warna setiap piksel dari gambar berwarna dengan konten serupa. Dalam makalah ini, penulis menggunakan pendekatan kedua; penulis mengekstrak informasi tentang warna dari suatu gambar dan mentransfernya ke gambar lain.

Belakangan ini, deep learning telah mendapatkan perhatian yang meningkat di kalangan peneliti di bidang visi komputer dan pengolahan gambar. Sebagai teknik khas, jaringan syaraf konvolusional (CNN) telah banyak diteliti dan berhasil diterapkan pada beberapa tugas seperti pengenalan gambar, rekonstruksi gambar, generasi gambar, dll. Sebuah CNN terdiri dari beberapa lapisan unit komputasi kecil yang hanya memproses bagian-bagian gambar input secara feed-forward. Setiap lapisan adalah hasil dari penerapan berbagai filter gambar, masing-masing mengekstrak fitur tertentu dari gambar input, ke lapisan sebelumnya. Dengan demikian, setiap lapisan dapat mengandung informasi berguna tentang gambar input pada tingkat abstraksi yang berbeda.

Dengan evolusi sumber daya komputasi, terutama daya komputasi GPU, telah menjadi mungkin untuk melatih CNN yang sangat dalam, dan mereka telah mencapai beberapa hasil yang luar biasa belakangan ini. Sebagai contoh, sebuah CNN dalam skala besar (He dkk., 2015) telah melampaui kinerja manusia dalam klasifikasi *ImageNet*, atau sebuah *adversarial network* (Radford dkk., 2015), di mana dua CNN dilatih secara simultan, mampu menghasilkan gambar yang tampak meyakinkan dari berbagai jenis objek. Keberhasilan menakjubkan ini dari CNN telah mendorong kami untuk lebih menyelidiki dan menjelajahi potensinya dalam masalah colorisasi gambar yang disebutkan sebelumnya.

II. DASAR TEORI

A. *Image Colorization*

Terdapat dua teknik untuk menghasilkan gambar berwarna dari bentuk skala abu-abu:

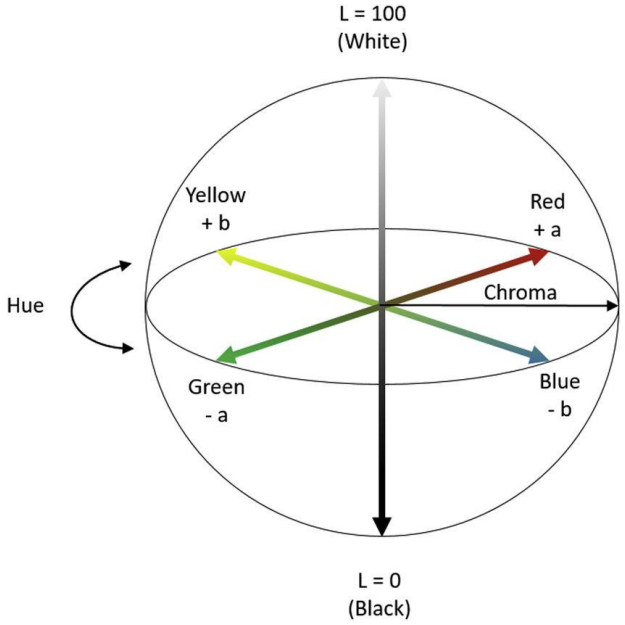
1. Mengubah Gambar RGB menjadi Gambar LAB:
   a. Mengonversi gambar RGB menjadi gambar LAB.
   b. Memisahkan nilai L dan nilai ab dari gambar.
   c. Melatih model untuk memprediksi nilai ab.
2. Mengubah Gambar RGB menjadi Gambar LUV:

a. Mengonversi gambar RGB menjadi gambar LUV.

b. Memisahkan nilai L dan nilai UV dari gambar.

c. Melatih model untuk memprediksi nilai UV.

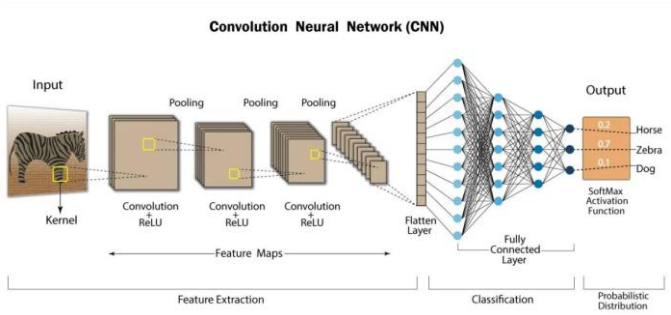
Kedua pendekatan ini menggunakan konsep konversi warna ke ruang warna yang berbeda, yaitu LAB atau LUV, di mana nilai L mewakili kecerahan dan nilai ab atau UV mewakili komponen warna. Model dilatih untuk mempelajari hubungan antara nilai kecerahan dan komponen warna, sehingga dapat memprediksi komponen warna yang sesuai untuk gambar skala abu-abu. Pendekatan ini memanfaatkan representasi warna yang terpisah untuk meningkatkan kinerja model dalam menangkap informasi warna pada gambar. Pada penelitian kali ini penulis menggunakan metode pertama yakni dengan mengubahnya menjadi LAB.



Sistem warna CIELAB, atau CIE L a b, mewakili hubungan kuantitatif warna pada tiga sumbu: nilai L menunjukkan kecerahan, dan a dan b adalah koordinat kromatisitas. Pada diagram ruang warna, L direpresentasikan pada sumbu vertikal dengan nilai dari 0 (hitam) hingga 100 (putih). Nilai a menunjukkan komponen merah-hijau dari suatu warna, di mana +a (positif) dan -a (negatif) menunjukkan nilai merah dan hijau, secara berturut-turut. Komponen kuning dan biru direpresentasikan pada sumbu b sebagai nilai +b (positif) dan -b (negatif). Pada pusat bidang adalah warna netral atau akromatik. Jarak dari sumbu pusat mewakili kroma (C), atau saturasi warna. Sudut pada sumbu kromatisitas mewakili hue. Nilai L, a, dan b dapat ditranskripsikan ke parameter dermatologis. Nilai L berkorelasi dengan tingkat pigmen kulit. Nilai a berkorelasi dengan eritema. Nilai b berkorelasi dengan pigmen dan pengecapan. Seperti ruang warna RGB, Lab juga adalah ruang warna 3 saluran, tetapi tidak seperti ruang warna RGB, informasi warna dikodekan hanya dalam saluran a (komponen merah-hijau) dan b (komponen biru-kuning). Saluran L (kecerahan) hanya mengkodekan informasi intensitas.

B. CNN

Convolutional Neural Network (CNN atau ConvNet) adalah algoritma *deep learning* yang populer, umumnya digunakan untuk memproses data yang memiliki topologi seperti grid. Contoh data berbentuk grid adalah citra atau gambar.

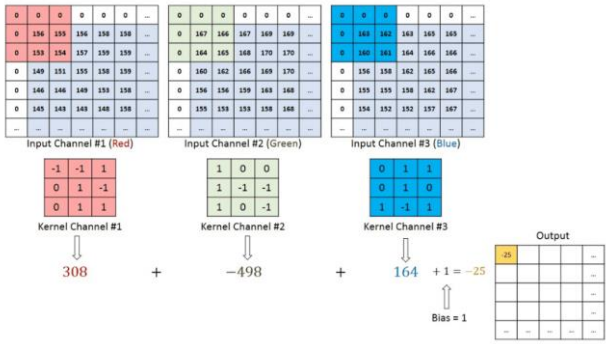


(sumber: Convolutional Neural Network | Deep Learning | Developers Breach)

CNN dapat disebut juga jaringan syaraf tiruan yang melibatkan konvolusi (CNN = ANN + convolution). CNN terdiri dari 3 lapisan utama:

1. Convolutional Layer (+ReLU)

Matriks kecil yang digunakan untuk mengekstrak fitur dari gambar. Konvolusi melibatkan pergeseran filter di seluruh gambar untuk menghasilkan peta fitur. Tiap filter menghasilkan luaran yang disebut feature map.



Beberapa hyperparameter yang mempengaruhi ukuran feature map, yaitu :

a. Jumlah filter. Dua filter berbeda menghasilkan dua feature map berbeda, sehingga output-nya memiliki dua kanal.

b. Stride: jumlah langkah pergeseran filter (default = 1). Semakin besar stride, maka semakin kecil ukuran output yang dihasilkan.

c. Padding: penambahan nilai nol di sekitar gambar untuk mempertahankan informasi di tepi (jika diperlukan).

ReLU adalah Fungsi aktivasi yang umum digunakan dalam CNN untuk memperkenalkan non-linearitas ke dalam model. ReLu merupakan layer tambahan yang memungkinkan pelatihan yang lebih cepat dan efektif dengan memetakan nilai negatif ke nol dan

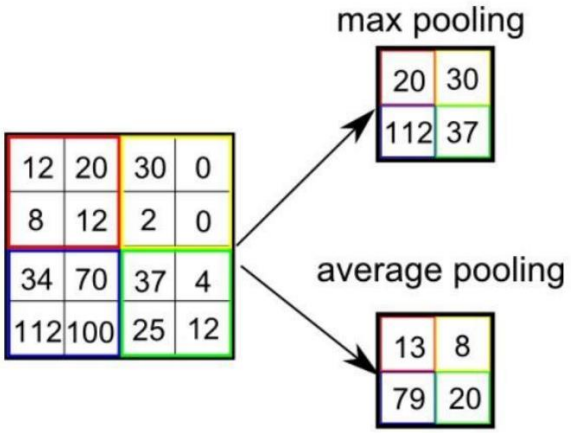
mempertahankan nilai positif. Pada dasarnya ReLU adalah operasi per-pixel dengan cara mengganti nilai negatif pixel di dalam feature map menjadi nol.

2. Pooling Layer

Pooling layer adalah layer dimana terjadi operasi pengurangan dimensi yang membantu mengurangi kompleksitas model dengan mempertahankan fitur yang paling penting. Ada dua jenis pooling:

a. *Max Pooling*, yakni mengembalikan nilai maksimum dari bagian gambar yang dicakup oleh kernel.

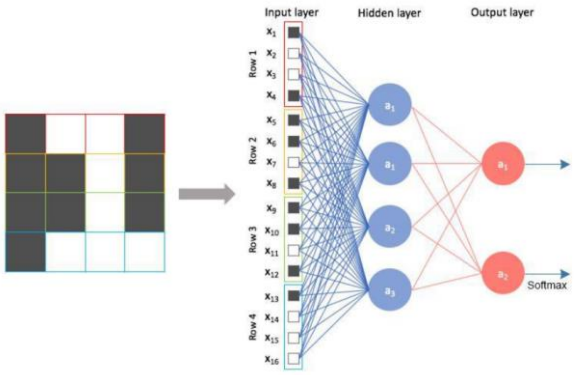
b. *Average Pooling*, yakni mengembalikan rata-rata semua nilai dari bagian gambar yang dicakup oleh kernel.



(sumber: A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science)

3. Fully-Connected Layer

Fully-Connected Layer adalah layer yang menghubungkan setiap neuron dengan setiap neuron di layer sebelumnya dan setelahnya. Layer ini menghasilkan vektor dimensi K, dalam hal ini K adalah jumlah kelas yang dapat diprediksi oleh jaringan. Vektor ini berisi probabilitas untuk setiap kelas dari setiap gambar yang diklasifikasikan.



(sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/24-CNN-2022.pdf)

III. Penelitian Terkait

Gatys dkk. (2015) mempresentasikan aplikasi dari neural network dalam mempelajari dan menerapkan pola dari satu gambar ke gambar lainnya. Diberikan sebuah gambar yang kontennya ingin dipertahankan (content image) dan gambar lain yang polanya ingin ditransfer (style image), mereka memasukkan kedua gambar tersebut ke dalam CNN yang telah dilatih sebelumnya dan mengekstrak representasi konten dan representasi pola masing-masing. Selanjutnya, mereka melakukannya pada gambar yang berderau dan membuat perubahan hingga mereka mendapatkan representasi yang serupa dengan content image dan style image. Hal bertujuan untuk meminimalkan kerugian dalam merekonstruksi konten dan pola secara simultan. Seperti yang disebutkan dalam makalah mereka, mereka menggunakan gradien descen untuk menyelesaikan masalah ini.

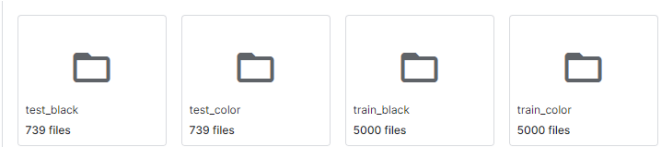
IV. Implementasi

Pada bagian ini akan dijelaskan mengenai implementasi yang dilakukan oleh penulis sesuai dengan dasar teori dan teknik-teknik yang digunakan. Pada implementasi ini digunakan dataset yang berasal dari Image Colorization Dataset (kaggle.com).

A. *Dataset*

Dataset terdiri atas 4 bagian yakni:
a. train_black
b. train_color
c. test_black
d. test_color

Berikut ini adalah struktur dan ukuran dari dataset.



B. *Preprocess*

Preprocess yang digunakan pada penelitian ini adalah konversi gambar dari RGB ke LAB sesuai pada teknik Image Colorization pada dasar teori. Pada bagian ini, setiap gambar pada dataset diubah menjadi format LAB agar dapat diproses kecerahan dan komponen warnanya. Berikut adalah kode yang digunakan.

```
#Convert from RGB to LAB
X = []
y = []
for img in train:
  try:
    lab = rgb2lab(img)
    X.append(lab[:,:,0])
    y.append(lab[:,:,1:] / 128)
  except:
    print('error')
```

Dengan melakukan iterasi pada setiap gambar, penulis mengonversi gambar dari format RGB ke Lab. Bayangkan gambar Lab sebagai gambar abu-abu di saluran L dan semua informasi warna disimpan di saluran A dan B. Input untuk jaringan akan menjadi saluran L, sehingga kami menugaskan saluran L ke vektor X. Dan menugaskan A dan B ke Y.

Untuk mengubah gambar dari format RGB menjadi Lab, kami menggunakan fungsi ***rgb2lab()*** dari perpustakaan skimage. Setelah mengonversi ruang warna menggunakan fungsi rgb2lab(), kami memilih lapisan grayskale dengan: [:, :, 0]. Ini menjadi input untuk neural network. [:, :, 1:] memilih dua lapisan warna, yaitu hijau-merah dan biru-kuning.

Ruang warna Lab memiliki rentang yang berbeda dibandingkan dengan RGB. Spektrum warna ab di Lab berkisar dari -128 hingga 128. Dengan membagi semua nilai dalam lapisan output dengan 128, rentangnya dibatasi antara -1 dan 1.

C. Training Model

Pada bagian ini dilakukan training model yang mengimplementasikan CNN dengan kode sebagai berikut.

```
#Built model

#Encoder
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2, input_shape=(224, 224, 1)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3,3), activation='relu', padding='same', strides=2))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same', strides=2))
model.add(Conv2D(512, (3,3), activation='relu', padding='same'))
model.add(Conv2D(512, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))

#Decoder
model.add(Conv2D(128, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(64, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3,3), activation='relu', padding='same'))
model.add(Conv2D(16, (3,3), activation='relu', padding='same'))
model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))
model.add(UpSampling2D((2, 2)))
model.compile(optimizer='adam', loss='mse' , metrics=['accuracy'])
model.summary()
```

Model terdiri atas 16 layer yang rinciannya adalah sebagai berikut.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 112, 112, 64)      640

 conv2d_27 (Conv2D)          (None, 112, 112, 128)     73856

 conv2d_28 (Conv2D)          (None, 56, 56, 128)       147584

 conv2d_29 (Conv2D)          (None, 56, 56, 256)       295168

 conv2d_30 (Conv2D)          (None, 28, 28, 256)       590080

 conv2d_31 (Conv2D)          (None, 28, 28, 512)       1180160

 conv2d_32 (Conv2D)          (None, 28, 28, 512)       2359808

 conv2d_33 (Conv2D)          (None, 28, 28, 256)       1179904

 conv2d_34 (Conv2D)          (None, 28, 28, 128)       295040

 up_sampling2d_6 (UpSamplin  (None, 56, 56, 128)       0
 g2D)

 conv2d_35 (Conv2D)          (None, 56, 56, 64)        73792

 up_sampling2d_7 (UpSamplin  (None, 112, 112, 64)      0
 g2D)

 conv2d_36 (Conv2D)          (None, 112, 112, 32)      18464

 conv2d_37 (Conv2D)          (None, 112, 112, 16)      4624

 conv2d_38 (Conv2D)          (None, 112, 112, 2)       290

 up_sampling2d_8 (UpSamplin  (None, 224, 224, 2)       0
 g2D)

=================================================================
Total params: 6219410 (23.73 MB)
Trainable params: 6219410 (23.73 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

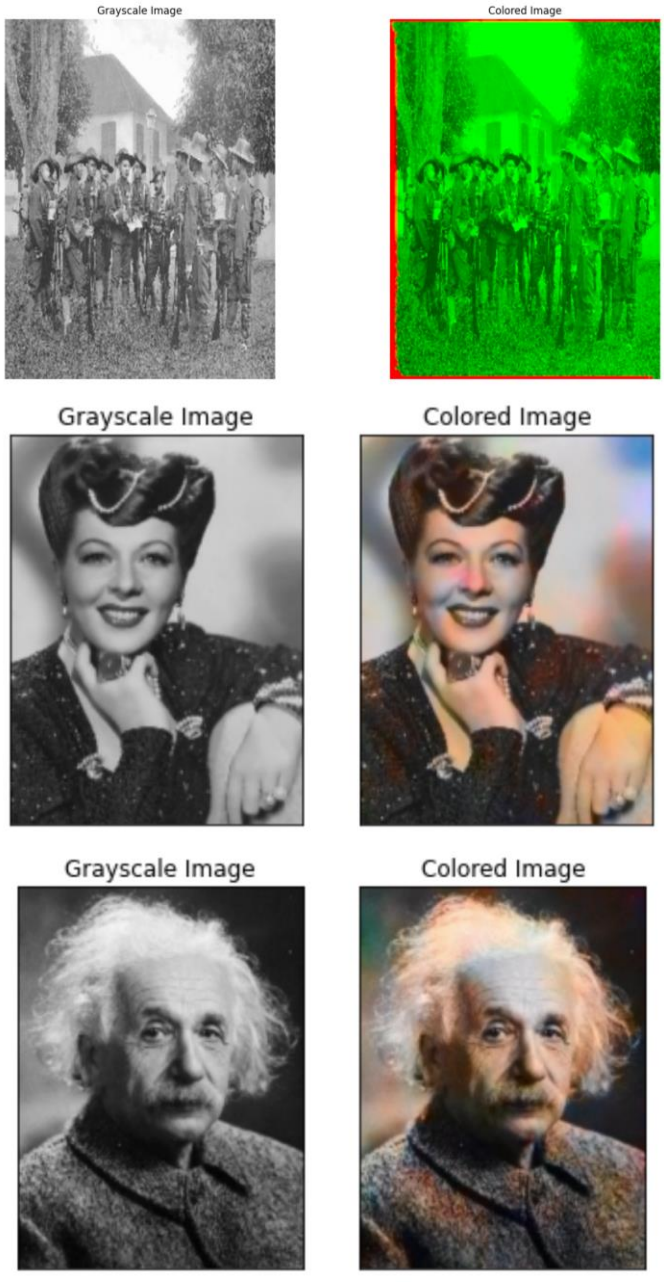
D. Prediction

Pada bagian ini dilakukan prediksi untuk pewarnaan gambar menggunakan model yang telah dilatih sebelumnya dengan CNN. Dengan melakukan hal yang sama seperti teknik preprocess yakni mengubah gambar RGB menjadi LAB sesuai pada kode di bawah.

```
#Convert to LAB
test_img = []
for img in test[0]:
  try:
    lab = rgb2lab(img)
    test_img.append(lab[:,:,0])
  except:
    print('error')
test_img = np.array(test_img)

test_img = test_img.reshape(test_img.shape+(1,)) #dimensions to be the same for test_img
```

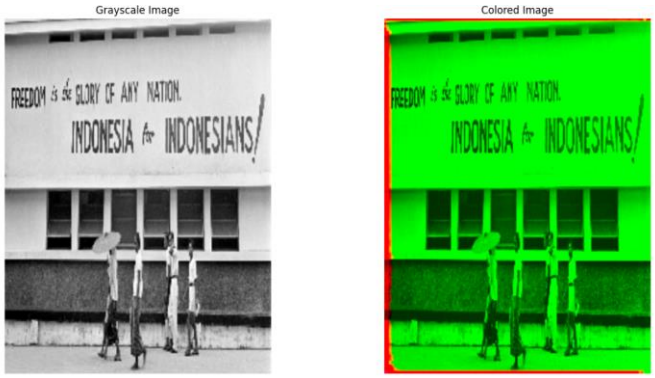
Lalu dilakukan prediksi dengan model yang sudah dilatih sesuai dengan kode di bawah ini.

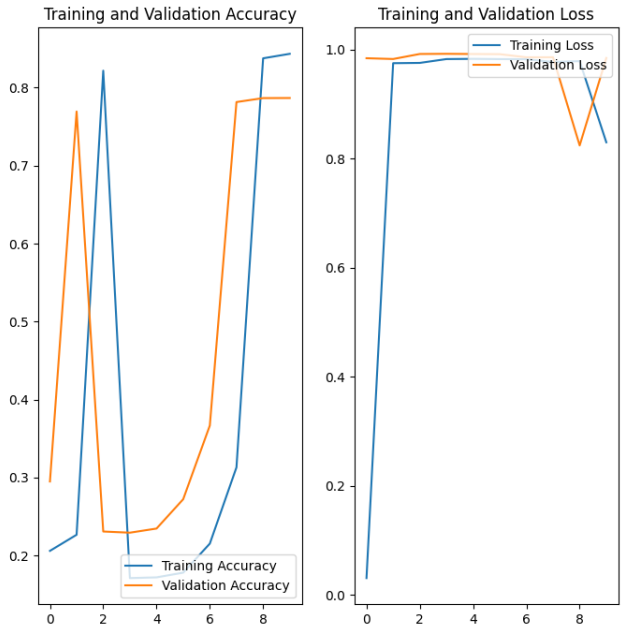
```
i = 0
plt.figure(figsize=(15, 100))
for img in test_img:
    grayscale = np.zeros((224, 224, 3))
    grayscale[:,:,0] = img[:,:,0]
    grayscale = resize(grayscale, (800,600))
    gray_img = lab2rgb(grayscale)

    output1 = model.predict(test_img)
    output1 = output1*128

    result = np.zeros((224, 224, 3))
    result[:,:,0] = img[:,:,0]
    result[:,:,1:] = output1[0]
    result = resize(result, (800,600))
    color_img = lab2rgb(result)

    i = i + 1
    ax = plt.subplot(11, 2, i)
    plt.imshow(gray_img)
    plt.title(f"Grayscale Image")
    plt.axis("off")
    i = i + 1
    ax = plt.subplot(11, 2, i)
    plt.imshow(color_img)
    plt.title(f"Colored Image")
    plt.axis("off")
```



V. HASIL PENELITIAN

Setelah dilakukan training atas model yang dibuat dengan CNN dan model prediksi pewarnaannya. Selanjutnya model akan dicoba melakukan pewarnaan atas beberapa gambar. Berikut ini adalah hasil konversi gambarnya. Model akan dicoba untuk dataset yang berbeda.





Di atas ini adalah hasil dari training model beserta training dan validation loss. Dapat dilihat bahwa hasil yang didapatkan cukup sesuai dan loss rate-nya juga kecil.

VI. KESIMPULAN

Image Colorization merupakan suatu tantangan yang saat ini sudah mulai terjawab karena perkembangan zaman. Dapat dilihat untuk hasil yang didapat walaupun tidak terlalu bagus tetapi sudah mulai menunjukkan identitas asli dari warna gambar yang seharusnya ada dalam *citra grayscale* tersebut.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision 2015, pp. 1026-1034.

[2] A. Radford, L. Metz, and S. Chintala, 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Under review for International Conference on Learning Representations 2016 (arXiv:1511.06434v2).

[3] L. A. Gatys, A. S. Ecker, and M. Bethge, 2015. A Neural Algorithm of Artistic Style. arXiv:1508.06576v2.

[4] Munir, Rinaldi. (2023). Bahan Kuliah IF4073 Interpretasi dan Pengolahan Citra. Program Studi Informatika ITB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2023

Roby Purnomo (13520106)