BAB 2

LANDASAN TEORI

2.1. Deep Learning

Deep Learning merupakan sub-bidang dari *Machine Learning* yang bekerja dengan melakukan representasi dari data pada lapisan pembelajaran yang berlapis-lapis hingga representasi menjadi lebih bermakna. *Deep* pada *Deep Learning* memiliki makna bahwa *Deep Learning* menggagas lapisan representasi yang berurutan. Sekarang ini, *Deep Learning* biasanya memiliki lapisan dengan jumlah puluhan hingga ratusan lapisan yang berurutan dan lapisan-lapisan tersebut dapat secara otomatis mempelajari data *training* yang diberikan. Pada *Deep Learning* lapisan representasi tersebut bernama *Neural Networks*. *Neural Networks* memiliki struktur yang bertumpuk yang berarti suatu lapisan akan berada diatas lapisan lain. *Neural Networks* mengambil konsep dari bidang *neurobiology* dan terinspirasi dari kemampuan dalam memahami sesuatu seperti yang dilakukan oleh otak manusia. Meskipun konsep utama *Deep Learning* dikembangkan atas inspirasi tersebut, tetapi model *Deep Learning* bukanlah model dari otak manusia. Hal tersebut dikarenakan belum ada bukti yang menunjukan bahwa otak manusia bekerja layaknya model *deep learning* seperti sekarang ini (Chollet, 2018).

2.2. Convolutional Neural Network

Convolutional Neural Network yang dikenal sebagai CNN atau ConvNet merupakan suatu lapisan atau *layer* yang terdiri dari *neuron* yang banyak dan dioptimalkan dengan melakukan pembelajaran (O'Shea, K. dan Nash, R., 2015). CNN bekerja dengan melakukan ekstraksi terhadap *low-level features* dari *raw*

input dan semakin dalam *raw input* tersebut berada pada lapisan CNN, maka semakin tinggi *level features* yang akan diekstraksi (Schirrmeister, *et al*, 2017).

Pada dasarnya, CNN memiliki tiga jenis lapisan, yaitu lapisan *Convolutional*, lapisan *Pooling*, dan lapisan Full Connection (K O'Shea, R Nash, 2015). Berikut penjelasan dari masing-masing lapisan.

1. Lapisan Convolutional: Merupakan lapisan yang berfungsi untuk mempelajari, mendeteksi, dan menangkap fitur dari sebuah gambar *input*. Jika gambar berada pada lapisan *convolutional* awal, maka fitur yang ditangkap akan memiliki sifat *low-level features* seperti warna gambar dan jika gambar berada pada lapisan yang semakin dalam maka fitur yang akan ditangkap akan memiliki sifat *high-level features* seperti tekstur (Liu, Shen dan Van Den Hengel, n.d.). Lapisan *convolutional* diikuti dengan lapisan *Rectified Linear Unit* (ReLu). *ReLu* bertugas untuk memperbaiki pixel yang bersifat *nonlinear* setelah selesai melewati lapisan konvolusi dengan menggunakan fungsi 2.1 berikut (Agarap, A.F., 2018).

$$f(x) = \max(0, x) \tag{2.1}$$

2. Lapisan *Pooling*: Merupakan lapisan yang berfungsi untuk menggabungkan beberapa aktivasi untuk menghasilkan vektor dengan ukuran yang lebih kecil sehingga memiliki ukuran yang sesuai dengan lapisan CNN. Terdapat 2 jenis Pooling yang umum digunakan, yaitu *Max Pooling* dan *Average Pooling* (Christlein, V., *et al*., 2019). Pada Max Pooling, nilai terbesar pada matriks yang telah dikelompokan sesuai dengan ukuran *filter* akan menjadi representasi dari grup tersebut. Pada Average Pooling, seluruh nilai yang termasuk pada filter

akan dirata-rata terlebih dahulu, dan nilai tersebut akan menjadi representasi dari grup tersebut. (H., Gholamalinezhad dan H., Khosravi, 2020)

3. Lapisan Full Connection (FC): Merupakan lapisan akhir pada CNN dan memiliki fungsi untuk melakukan proses klasifikasi final. Hal tersebut dapat dilakukan dengan menggabungkan tiap fitur yang telah telah di proses pada lapisan sebelumnya untuk menjadi vektor dan membentuk sebuah model yang digunakan untuk melakukan klasifikasi.

2.3. VGG19

Terdapat banyak model arsitektur CNN untuk melakukan pengolahan citra, salah satunya adalah VGG atau Visual Geometry Group. Pada tahun 2010, sebuah lomba diadakan oleh ImageNet yang disebut *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC). Lomba tersebut bertujuan untuk melakukan klasifikasi terhadap gambar dengan jumlah yang sangat banyak dan pada tahun 2014, tim bernama VGG berhasil membuat model CNN yang memiliki tingkat error klasifikasi yang rendah. Hal tersebut membuat mereka berhasil meraih posisi *runner-up* pada kompetisi ILSVRC. VGG sendiri merupakan *Convolutional Neural Network Model* yang diciptakan oleh Karen Simonyan dan Andrew Zisserman dari *University of Oxford*. Setelah memenangkan ILSVRC pada tahun 2014, diluncurkan *paper* yang berjudul *"Very Deep Convolutional Networks for Large-Scale Image Recognition"*. VGGNet memiliki beberapa jenis, dimana masing-masing jenis memiliki perbedaan pada jumlah lapisan konvolusi yang digunakan. Arsitektur VGG dengan jumlah layer terbanyak saat ini adalah VGG19 yang secara spesifik memuat 19 lapisan utama dengan *filter* atau *kernel* berukuran 3x3. 19 lapisan utama tersebut terdiri dari 16 lapisan konvolusi yang masing-masing

lapisannya diikuti oleh lapisan ReLu dan 3 lapisan FC. Selain itu, terdapat juga lapisan lain seperti 5 lapisan MaxPool dan 1 lapisan SoftMax. Berikut Tabel 1.1 yang menunjukan lapisan VGG19.

Tabel 1.1 Lapisan VGG19

1.	Input: Fixed Size (224, 224, 3)
2.	Conv1_1 (3, 64, stride = 1, padding = 1) + ReLu
3.	Conv1_2 (64, 64, stride = 1, padding = 1) + ReLu
4.	MaxPool (kernel = 2x2, stride = 2, padding = 0)
5.	Conv2_1 (64, 128, stride = 1, padding = 1) + ReLu
6.	Conv2_2 (128, 128, stride = 1, padding = 1) + ReLu
7.	MaxPool (kernel = 2x2, stride = 2, padding = 0)
8.	Conv3_1 (128, 256, stride = 1, padding = 1) + ReLu
9.	Conv3_2 (256, 256, stride = 1, padding = 1) + ReLu
10.	Conv3_3 (256, 256, stride = 1, padding = 1) + ReLu
11.	Conv3_4 (256, 256, stride = 1, padding = 1) + ReLu
12.	MaxPool (kernel = 2x2, stride = 2, padding = 0)
13.	Conv4_1 (256, 512, stride = 1, padding = 1) + ReLu
14.	Conv4_2 (512, 512, stride = 1, padding = 1) + ReLu
15.	Conv4_3 (512, 512, stride = 1, padding = 1) + ReLu
16.	Conv4_4 (512, 512, stride = 1, padding = 1) + ReLu
17.	MaxPool (kernel = 2x2, stride = 2, padding = 0)
18.	Conv5_1 (512, 512, stride = 1, padding = 1) + ReLu
19.	Conv5_2 (512, 512, stride = 1, padding = 1) + ReLu
20.	Conv5_3 (512, 512, stride = 1, padding = 1) + ReLu
21.	Conv5_4 (512, 512, stride = 1, padding = 1) + ReLu
22.	MaxPool (kernel = 2x2, stride = 2, padding = 0)
23.	Fully Connected (4096)
24.	Fully Connected (4096)
25.	Fully Connected (1000)
26.	SoftMax

Berdasarkan Tabel 1.1, VGG19 memiliki *input* berukuran tetap, yaitu 224x224 dengan format RGB. Ketika *input* memasuki lapisan Conv1_1, ukuran input berubah menjadi 224x224x64. Hal ini terjadi karena lapisan konvolusi Conv1_1 memiliki filter berjumlah 64 yang masing-masing berukuran filter 3x3 dengan *stride* berukuran 1. Kemudian, input akan masuk ke lapisan ReLu yang memiliki fungsi untuk mengganti nilai *non-linearity* milik input dari hasil

perhitungan pada lapisan Conv1_1. Lapisan selanjutnya yang akan dilewati adalah Conv2_2. Pada lapisan ini, tidak terjadi perubahan ukuran pada *input*, karena lapisan Conv2_2 memiliki parameter masukan sebanyak 64 *channel* dan keluaran sebanyak 128 *channel*. Setelah itu, lapisan yang akan dilewati input adalah lapisan MaxPool. Oleh karena lapisan ini memiliki format *padding*, *kernel,* dan *stride* yang berbeda dengan lapisan konvolusi, maka nilai *height* dan *width* input akan berubah dari nilai awal 224 menjadi setengahnya dan proses ini akan terus berulang hingga *input* berada pada lapisan MaxPool sebelum lapisan FC. Ketika *input* memasuki lapisan FC tersebut, maka vektor *input* tersebut akan dihubungkan satu sama lain sehingga dapat diproses pada lapisan *softmax* dan mendapatkan hasil akhir klasifikasi. (K. Simonyan dan A. Zisserman, 2014).

2.4. Neural Style Transfer

Salah satu metode *Deep Learning Style Transfer* yang popuer adalah *Neural Style Transfer. Neural Style Transfer* pertama kali diperkenalkan pada tahun 2015 melalui *paper* karya Leon A. Gatys, Alexander S. Ecker dan Matthias Bethge yang berjudul "*A Neural Algorithm of Artistic Style*". *Paper* tersebut adalah yang pertama membahas tentang cara menggunakan CNN untuk mengombinasikan gambar biasa dengan gaya lukisan yang terkenal. Hasil penelitian pertama mereka menunjukan bahwa hierarki CNN mampu untuk mengekstraksi fitur dari gambar konten dan gambar gaya. Pada hierarki tersebut, terdapat lapisan yang disebut *higher-layer* dan *lower-layer* (Gatys, *et al.*, 2015). Sesuai namanya, *higher-layer* berfungsi untuk menangkap konten tingkat tinggi pada suatu gambar, dalam konteks ini *higher-layer* dapat disebut juga sebagai representasi konten. Sedangkan *lower-layer* digunakan untuk melakukan reproduksi nilai piksel yang tepat dan sesuai dengan

gambar masukan. Kedua *layer* ini akan digunakan pada saat proses representasi pada gambar konten gambar gaya (Majumdar, *et al.*, 2018).

Berdasarkan hasil penelitian tersebut, Gatys, *et al.* mengusulkan untuk memanfaatkan kemampuan CNN untuk menggabungkan gambar biasa dengan gaya dari lukisan terkenal. Gagasan utama algoritma ini adalah mengoptimalkan suatu gambar secara iteratif dengan tujuan untuk mencocokkan distribusi fitur yang didapatkan dari hasil ekstraksi pada lapisan CNN yang diinginkan. Percobaan yang telah mereka lakukan memicu munculnya bidang penelitian yang disebut *Neural Style Transfer* (NST). Algoritma ini menggunakan arsitektur VGG untuk menciptakan gambar dengan berbagai gaya baru. Karya mereka berakhir menjadi sebuah perhatian luas di kalangan para ilmuan dan dunia industri. Banyak penelitian baru bermunculan terkait algoritma NST dengan tujuan untuk mengembangkan, menyempurnakan, dan membuat terobosan mengenai algoritma tersebut (Jing, *et al.*, 2017), seperti *paper* karya Prutha Date, Ashwinkumar Ganesan dan Tim Oates yang berjudul "*Fashioning with Networks Neural Style Transfer to Design*" yang ingin menerapkan algoritma NST untuk membentuk sebuah pakaian dengan motif dan warna yang sesuai dengan preferensi pengguna mesin mereka. Mereka membentuk pendekatan untuk menghasilkan pakaian tersebut dengan cara mempelajari pilihan mode atau gaya pakaian pengguna dari lemari pribadi mereka. Kemudian pendekatan akan di evaluasi dengan menganalisis gambar pakaian yang dihasilkan dan seberapa baik mesin bisa menyesuaikan dengan selera berpakaian pengguna (Date, *et al.*, 2017).

Objektif utama dari NST adalah tentang bagaimana cara untuk mengambil fitur gambar dari lapisan CNN. Itulah yang menjadi dasar digunakannya lapisan

konvolusi CNN seperti VGG19. Setiap unit lapisan konvolusi tersebut akan bekerja sebagai filter gambar yang dapat melakukan ekstraksi fitur dari gambar yang digunakan. Setiap hasil yang dikeluarkan oleh suatu lapisan, dapat disebut sebagai *feature maps* (Gatys, *et al.*, 2015). Namun, NST tidak dapat dilakukan hanya dengan melakukan ekstraksi fitur pada lapisan VGG19, oleh karena itu Gatys, *et al*., menggunakan *loss function* untuk memastikan bahwa fitur yang diambil dapat diaplikasikan pada gambar. *Loss function* pada NST terbagi menjadi dua bagian yaitu *content loss* dan *style loss*. Tidak hanya Gatys, *et al*., Majumdar, *et al.* juga menggunakan lapisan VGG tersebut untuk menghitung *style loss* dan *content loss* pada setiap lapisan pada model VGG (Majumdar, *et al.*, 2018).

Content loss berfungsi untuk mencari perbedaan representasi fitur konten pada layer konvolusi *l* antara gambar konten P_{ij}^{l} dan *output* F_{ij}^{l} dengan menggunakan *Mean Square Error* (MSE).

$$L_{content}(\vec{p}, \vec{\chi}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^{l} - P_{ij}^{l})^2 \tag{2.2}$$

Kemudian *style loss* digunakan untuk melakukan ekstraksi *style* dari gambar gaya untuk diaplikasikan pada gambar *output*. Hal tersebut dapat dilakukan dengan cara mencari korelasi antara nilai gambar gaya dengan gambar *output*. Untuk melakukan itu, digunakan *Gram Matrix* (GM) yang akan melakukan perkalian matriks gambar $T_{i,k}^{l}$ dengan matriks gambar yang telah di *transpose* $T_{j,k}^{l}$ untuk seluruh *channel* konvolusi *k*.

$$G_{i,k}^{l} = \sum_{k} (T_{i,k}^{l} T_{j,k}^{l}) \tag{2.3}$$

Dengan operasi GM yang melakukan perkalian antara matriks gambar dengan *transpose* dirinya sendiri, maka akan didapatkan matriks baru yang merepresentasikan korelasi antara nilai matriks *i* dan *j*. Setelah nilai GM didapat, maka *style loss* pada layer *l* dapat didefinisikan dengan menggunakan MSE dari GM *output* dengan GM *style*. Nilai dari MSE kemudian dikalikan dengan *weight* dari layer gaya *l* dan akan dijumlahkan seluruhnya sebelum melakukan pembagian dengan jumlah *channel* lapisan konvolusi *N* dan *height* x *width* gambar *M*.

$$L_{style}(\vec{a}, \vec{x}) = \frac{1}{4N_l^2 M_l^2} \sum_{l=0}^{L} \omega_l \sum_{i,j} (G^l_{(output)ij} - G^l_{(style)ij})^2 \qquad (2.4)$$

Setelah mendapatkan nilai *content loss* dan *style loss*, maka digunakan fungsi *total loss* yang merupakan penjumlahan dari hasil perkalian *content loss* dengan *content weight* α dan *style loss* dengan *style weight* β.

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = (\alpha \times L_{content}(\overrightarrow{p}, \overrightarrow{\chi}, l)) + (\beta \times L_{style}(\vec{a}, \vec{x})) \qquad (2.5)$$

Total loss selanjutnya akan diproses untuk membentuk gradien dengan melakukan *backward*. *Backward* bekerja dengan melakukan *derivatives* terhadap fungsi yang berkaitan dengan *total loss* dan menggunakan hasil turunan tersebut untuk membentuk matriks gradien yang memiliki bentuk yang sama dengan matriks *output*. Matriks gradien tersebut kemudian akan diteruskan pada *optimizer* Adam untuk melakukan *update value* terhadap nilai matriks gambar *output* sehingga gambar *output* dapat memiliki wujud yang menyerupai gambar konten dan gaya yang menyerupai gambar gaya (Gatys, *et al.*, 2015).

2.5. Adam Optimizer

Adaptive Moment Estimation atau Adam merupakan salah satu metode optimasi untuk model *Deep Learning*. Adam pertama kali diperkenalkan oleh Diederik P. Kingma dan Jimmy Lei Ba melalui *paper* yang berjudul "*Adam: A Method for Stochastic Optimization*" pada tahun 2015. Kingma *et al*. menyebutkan bahwa Adam merupakan kombinasi dari algoritma AdaGrad dan RMSProp sehingga Adam mampu untuk melakukan optimasi terhadap permasalahan *sparse gradient* dan *noisy problem*. Dengan kombinasi tersebut, Kingma *et al*. berhasil membuat Adam mampu untuk menyelesaikan permasalahan *Deep Learning* dengan lebih efektif pada model dan dataset yang besar jika dibandingkan dengan algoritma lainnya **(**Kingma, D.P. dan Ba, J., 2014). Hasil performa Adam menjanjikan membuat banyak penelitian melakukan pembahasan mengenai perbandingan Adam dengan beberapa algoritma lain, seperti pada *paper* karya Sebastian Ruder yang berjudul "*An overview of gradient descent optimization algorithms*". Melalui penelitiannya, Sebastian Ruder menyebutkan bahwa performa optimasi Adam mungkin adalah pilihan secara umum yang terbaik dalam melakukan optimasi terhadap gradien (Ruder, S., 2016).

Untuk menjalankan fungsinya, Adam membutuhkan parameter seperti alpha α yang lebih dikenal sebagai *learning rate* atau *step size*, beta1 β_1 dengan nilai awal 0.9 dan beta2 β_2 dengan nilai awal 0.999 yang menunjukan nilai *exponential decay rate* dan epsilon ϵ dengan nilai awal 10^{-8} yang berfungsi untuk mencegah pembagian dengan angka nol. Dengan parameter tersebut, Adam dapat melakukan memperbarui nilai gradien θ_t dengan menggunakan persamaan sebagai berikut.

$$\theta_{t+1} = \theta_t - \frac{\alpha\, \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \tag{2.6}$$

Estimasi momen pertama $\widehat{m_t}$ dan estimasi momen kedua $\widehat{v_t}$ memiliki sebuah persamaan masing-masing. Berikut persamaan dari $\widehat{m_t}$ dan $\widehat{v_t}$ dimana t merupakan nilai iterasi yang sedang berjalan.

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{2.7}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{2.8}$$

Estimasi momen pertama $\widehat{m_t}$ dan kedua $\widehat{v_t}$ membutuhkan nilai vektor momen pertama m_t dan kedua v_t yang memiliki persamaan masing-masing. Berikut persamaan m_t dan v_t.

$$m_t = (1 - \beta_1) g_t + \beta_1 m_{t-1} \tag{2.9}$$

$$v_t = (1 - \beta_2) g_t^2 + \beta_2 v_{t-1} \tag{2.10}$$

Dimana gradien g_t didapatkan dari proses backward yang telah dilakukan pada saat training model berlangsung **(**Kingma, D.P. dan Ba, J., 2014).

2.6. Batik

Batik merupakan sebuah tradisi melukis diatas kain khas Indonesia yang sudah ada di Nusantara sejak zaman dahulu. Hal itu berarti bahwa batik telah menjadi bagian yang tak terpisahkan dengan orang Indonesia. Tradisi batik diyakini muncul pertama kali, khususnya di Jawa pada masa kerajaan Majapahit (Abad ke-12). Ditandai dengan ditemukannya sebuah Arca Prajnaparamita, yang berarti Dewi Kebijaksanaan) di Jawa Timur pada abad ke-13. Motif, corak, dan warna yang dimiliki pada batik sangatlah beragam tergantung dari mana batik tersebut berasal.

Motif batik sendiri diwariskan turun temurun dan memiliki maknanya masing-masing. Sebelumnya, batik matih kental dengan nuansa konservatif, namun sejak tahun 2009, batik menjadi lebih dikenal oleh banyak kalangan termasuk pada kaum muda. Hal ini dapat terjadi karena adanya momentum dimana batik diresmikan sebagai Warisan Kemanusiaan untuk Budaya Lisan dan Nonbendawi (*Masterpieces of the Oral and Intangible Heritage of Humanity*) oleh UNESCO (Asdhiana, I Made, 2013).

2.7. Likert Scale

Likert Scale atau skala Likert merupakan salah satu teknik pengukuran skala berbasis survey atau kuesioner yang dikemukakan oleh seorang psikolog sosial bernama Rensis Likert Skala Likert pertama kali digagaskan oleh Rensis Likert menggunakan *range* skala bernilai 1 yang merepresentasikan jawaban sangat tidak setuju sampai skala bernilai 5 yang merepresentasikan jawaban sangat setuju. (Likert, 1932). Berikut Tabel 1.2 yang berisi nilai beserta representasi jawaban pada skala 5 poin Likert.

Tabel 1.2 Skala 5 poin Likert

Nilai	Representasi Jawaban
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju

Berdasarkan Tabel 1.2, terlihat rentang nilai pada skala 5 poin Likert. Setelah mengumpulkan data melalui kuesioner, Adapun cara untuk menghitung persentase nilai rata-rata dari jawaban yang dipilih responden pada kuesioner dapat dilihat pada persamaan 2.11 (Nazir, 2005).

$$Total\ Score = (R1 * 1) + (R2 * 2)\ +\ (R3 * 3)\ +\ (R4 * 4)\ + (R5 * 5) \quad (2.11)$$

Pada persamaan 2.11, R1 merepresentasikan jumlah responden yang menjawab Sangat Tidak Setuju, R2 merepresentasikan jumlah responden yang menjawab Tidak Setuju, R3 merepresentasikan jumlah responden yang menjawab Netral, R4 merepresentasikan jumlah responden yang menjawab Setuju, dan R5 merepresentasikan jumlah responden yang menjawab Sangat Setuju. Setelah didapatkan nilai *total score*, selanjutnya adalah menghitung persentase interval skor. Adapun persamaan dari interval skor dapat dilihat pada persamaan 2.12.

$$Interval = \frac{100}{Nilai\ Skala\ Tertinggi} \quad (2.12)$$

Pada persamaan 2.12, nilai skala tertinggi pada skala 5 poin Likert adalah 5, sehingga didapatkan nilai interval sebesar 20. Setelah nilai interval didapatkan, dilakukan perhitungan untuk menghitung nilai skor akhir dalam wujud persentase seperti pada persamaan 2.13 berikut.

$$Persentase = \frac{Total\ Score}{Nilai\ Skala\ Tertinggi\ *\ Jumlah\ Responden} \times 100\% \quad (2.13)$$