BAB 2
LANDASAN TEORI

Untuk melakukan penelitian menggunakan Siamese Convolutional Neural Network diperlukan beberapa teori-teori untuk mendukung penelitian ini. Telaah literatur yang digunakan dalam penelitian ini adalah *Convolutional Neural Network*, *Artificial Neural Network*, *Siamese Convolutional Neural Network*, Triplet loss, Resnet 50, Optimizer Adam, dan Metriks Evaluasi. Berikut merupakan penjelasan lebih dalam dari teori tersebut:

2.1 Artificial Neural Network

Artificial Neural Network (Jaringan Syaraf Tiruan) merupakan jaringan tiruan yang berbasis pada struktur Syaraf Otak [11]. Struktur dari Artificial Neural Network terdiri dari tiga bagian utama yaitu input layer, hidden layer dan output layer. Informasi (α) diterima oleh input layer menggunakan bobot kedatangan (w) yang telah ditentukan. Selanjutnya, dilakukan penjumlahan bobot pada hidden layer. Hasil penjumlahan tersebut kemudian dibandingkan dengan nilai ambang batas (*threshold*). Jika nilai melebihi ambang batas, informasi akan diteruskan ke output layer. Namun, jika nilai tidak melebihi ambang batas, informasi tidak akan diteruskan ke output layer [12].

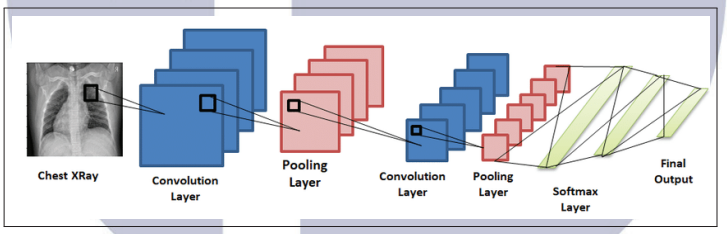
2.2 Convolutional Neural Network

Secara umum konvolusi didefinisikan sebagai cara untuk mengkombinasikan dua buah deret angka yang menghasilkan deret angka yang ketiga. Secara matematis, konvolusi adalah *integral* yang mencerminkan jumlah lingkupan dari sebuah fungsi a yang digeser atas fungsi b sehingga menghasilkan fungsi c. Konvolusi dilambangkan dengan asterisk. Sehingga, $axb = c$ berarti fungsi a dikonvolusikan dengan fungsi b menghasilkan fungsi c [13].

Konvolusi diskrit banyak digunakan dalam pengolahan citra untuk *image smoothing, edge detection* dan efek-efek lainnya, konvolusi dilakukan berdasarkan jumlah bobot dari piksel-piksel tetangga dengan bobot ditentukan berdasarkan ukuran window berupa matriks. *Window* atau disebut juga *sliding window* bergerak sepanjang piksel yang ada pada citra berukuran kecil yang biasa disebut

convolution mask atau *convolution kernel*. Orde matriks biasanya ganjil sehingga hasil konvolusi tepat berada ditengah-tengah, semakin besar ukuran *window*,beban komputasi akan semakin meningkat.[13]

Convolution Neural Network menggabungkan operasi konvolusi pada citra untuk ekstraksi fitur, dan Neural Network untuk klasifikasi. *Convolution Neural Network* dilakukan dengan cara menggunakan operasi konvolusi dari suatu pergerakan *kernel* yang terdiri dari beberapa *layer*, yaitu : *Convolution Layer, ReLu Layer, Subsampling, Pooling Layer*, dan *Fully Connected Layer (Input, Hidden, Output)*. Berikut ini adalah penjelasan tentang setiap lapisan layer yang digunakan dalam Convolutional Neural Network (CNN).



Gambar 2.1. CNN Layer
source: [14]

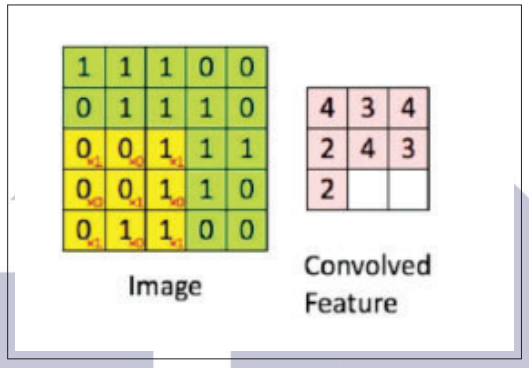
2.2.1 Convolution Layer

Convolution Layer melakukan operasi konvolusi pada output dari layer sebelumnya. Layer ini merupakan proses utama yang menjadi dasar dalam Convolutional Neural Network (CNN). Konvolusi dalam konteks ini mengacu pada penerapan fungsi pada output fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berarti menerapkan sebuah kernel (kotak kuning) pada citra dengan semua offset yang memungkinkan, seperti yang ditunjukkan dalam gambar di bawah ini[15].

Pada tahap ini, dilakukan operasi konvolusi antara matriks input dan matriks *filter*. Filter-filter ini akan digeser secara berulang-ulang pada seluruh area citra, menghasilkan matriks *feature map* sebagai keluaran. Matriks *feature map* ini dapat dihitung dengan menggunakan rumus berikut.

$$n_{out} = (\frac{n_{in} - k + 2p}{s}) + 1 \tag{2.1}$$

1. n_{out} : merupakan ukuran *feature map*



Gambar 2.2. Operasi Konvolusi
source: [15]

2. n_{in} : merupakan ukuran matriks input
3. k : merupakan ukuran matriks filter
4. p : merupakan ukuran *padding*
5. s : *stride*

Berikut adalah rumus untuk operasi konvolusi:

$$FM[i]_{j,k} = (\sum_{m}\sum_{n} N_{[j-m,k-n]} F_{[m,n]} + bF) \tag{2.2}$$

1. FM[i] : merupakan matriks *feature map* ke-i
2. N : merupakan matriks citra masukan
3. F : merupakan matriks *filter* konvolusi
4. bF : merupakan nilai bias pada filter
5. j,k : merupakan posisi piksel pada matriks citra masukan
6. m,n : merupakan posisi piksel pada matriks *filter* konvolusi

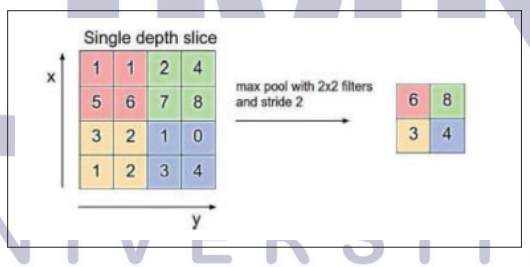
Setelah proses konvolusi selesai, langkah selanjutnya adalah menerapkan fungsi aktivasi menggunakan *Rectified Linear Unit (ReLu)*. Setiap piksel dalam *feature map* akan diinputkan ke fungsi ReLU, di mana piksel dengan nilai kurang dari 0 akan diubah menjadi 0. Rumus yang digunakan untuk ReLU adalah f(x) = max(0, x).

2.2.2 ReLu (Rectified Linear Unit Layer)

Layer ReLU, atau *Rectified Linear Unit Layer*, dapat dianggap sebagai suatu proses *thresholding* atau fungsi aktivasi dalam jaringan saraf tiruan. Tujuannya adalah untuk memastikan bahwa hasil dari proses konvolusi berada dalam domain nilai positif. Angka yang dihasilkan harus memiliki nilai positif karena fungsi aktivasi yang digunakan dalam propagasi balik jaringan saraf tiruan dalam penelitian ini adalah sigmoid. Oleh karena itu, setiap angka hasil dari proses konvolusi yang bernilai negatif akan melalui proses ReLU terlebih dahulu, yang akan mengubah nilai negatif menjadi nol.

2.2.3 Subsampling

Subsampling merupakan proses mengurangi ukuran data citra. Dalam pengolahan citra, *subsampling* juga bertujuan untuk meningkatkan invariansi posisi fitur. Pada sebagian besar CNN, metode *subsampling* yang digunakan adalah *max pooling*. *Max pooling* membagi keluaran dari lapisan konvolusi menjadi beberapa grid kecil, lalu mengambil nilai maksimal dari setiap grid untuk membentuk matriks citra yang telah direduksi. Seperti yang terlihat pada gambar di bawah ini, grid berwarna merah, hijau, kuning, dan biru adalah kelompok grid yang akan memilih nilai maksimumnya. Dengan demikian, hasil dari proses ini dapat dilihat pada kumpulan grid di sebelah kanan. Proses ini memastikan bahwa fitur yang diperoleh tetap sama meskipun objek citra mengalami translasi (pergeseran).

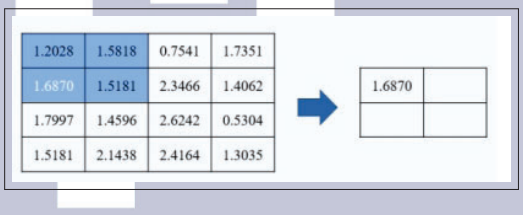


Gambar 2.3. Operasi Max Pooling
source: [15]

Penggunaan lapisan pooling pada CNN dimaksudkan untuk mengurangi ukuran citra sehingga dapat digantikan dengan lapisan konvolusi yang memiliki stride yang sama dengan lapisan pooling terkait [16].

2.2.4 Pooling Layer

Pooling layer digunakan untuk mengurangi ukuran dari peta fitur (*feature map*). Jenis pooling yang umum digunakan adalah *max pooling*, di mana nilai maksimum dipilih dalam suatu jendela atau area tertentu. Prosesnya mirip dengan layer konvolusi, di mana jendela digeser melintasi seluruh permukaan citra, namun dalam hal ini jendela digunakan sebagai acuan untuk memilih nilai maksimum dalam area yang ditentukan. Hasil dari proses tersebut adalah matriks *feature map* yang berisi nilai-nilai maksimum yang telah dipilih



Gambar 2.4. Contoh Gambar Max Pooling
source: [15]

2.2.5 Fully-connected Layer

Fully-connected Layer, yang sering digunakan dalam penerapan MLP, memiliki tujuan untuk melakukan transformasi pada dimensi data sehingga data dapat diklasifikasikan secara linear. Setiap neuron pada layer konvolusi perlu diubah menjadi data satu dimensi sebelum dimasukkan ke dalam *fully connected layer*. Namun, transformasi tersebut menyebabkan data kehilangan informasi spasialnya dan tidak dapat dikembalikan ke bentuk aslinya. Oleh karena itu, *fully connected layer* biasanya hanya diimplementasikan di akhir jaringan. Konvolusi layer dengan kernel berukuran 1x1 dapat memenuhi fungsi yang sama seperti *fully connected layer*, namun tetap mempertahankan karakter spasial dari data.

Fully-connected layer mengandung tiga komponen yaitu *input layer, hidden layer dan output layer.*

1. *Input Layer*

   Input Layer menggabungkan semua matriks *feature map* yang diperoleh dari *pooling layer* dan menjadikannya vektor dengan panjang sama dengan jumlah piksel pada matriks *pooling layer*. Seluruh nilai dalam *input layer* digunakan untuk perhitungan di *hidden layer*.

2. *Hidden Layer*

Perhitungan pada *hidden layer* melibatkan perkalian antara nilai-nilai pada *input layer* dengan bobot yang telah diinisialisasi sebelumnya, kemudian ditambahkan dengan nilai bias. Rumus perhitungannya adalah sebagai berikut:

$$z_i n_i = \sum_{j=1}^{m} X_j * V_{j,i} + V_{o,i} \tag{2.3}$$

(a) Z_in : masukkan untuk *node hidden layer* ke-i dengan total *node* n

(b) X_j : *node* X ke-j

(c) $V_{j,i}$: bobot V untuk Xj dan *node* Zi

(d) V_0 : *bias* V untuk zin_i

Setelah perhitungan dilakukan, fungsi aktivasi ReLU diterapkan pada semua hasil perhitungan, menghasilkan nilai keluaran Z. Hasil perhitungan ini akan digunakan dalam perhitungan pada *output layer*.

3. *Output Layer* Perhitungan pada *output layer* melibatkan perkalian antara nilai-nilai hasil perhitungan di *hidden layer* dengan bobot yang telah diinisialisasi sebelumnya, kemudian ditambahkan dengan nilai bias. Rumus perhitungannya adalah sebagai berikut:

$$y_i n_i = \sum_{j=1}^{m} Z_j * W_{j,i} + W_{o,i} \tag{2.4}$$

(a) Yin_i : Masukkan untuk *node hidden layer* Z ke-i dengan total *node* m

(b) Z_j : *node* Z ke j

(c) $W_{j,i}$: bobot W untuk Z_j dan *node* Y_i

(d) W_0 : *bias* W untuk Yin_i

Setelah perhitungan dilakukan, fungsi aktivasi Softmax diterapkan pada semua hasil perhitungan, menghasilkan nilai keluaran Y. Rumus perhitungan fungsi *softmax* adalah sebagai berikut:

$$Y_i = \frac{e^{y-in_i}}{\sum_{i=1}^{m} e^{M}} \tag{2.5}$$

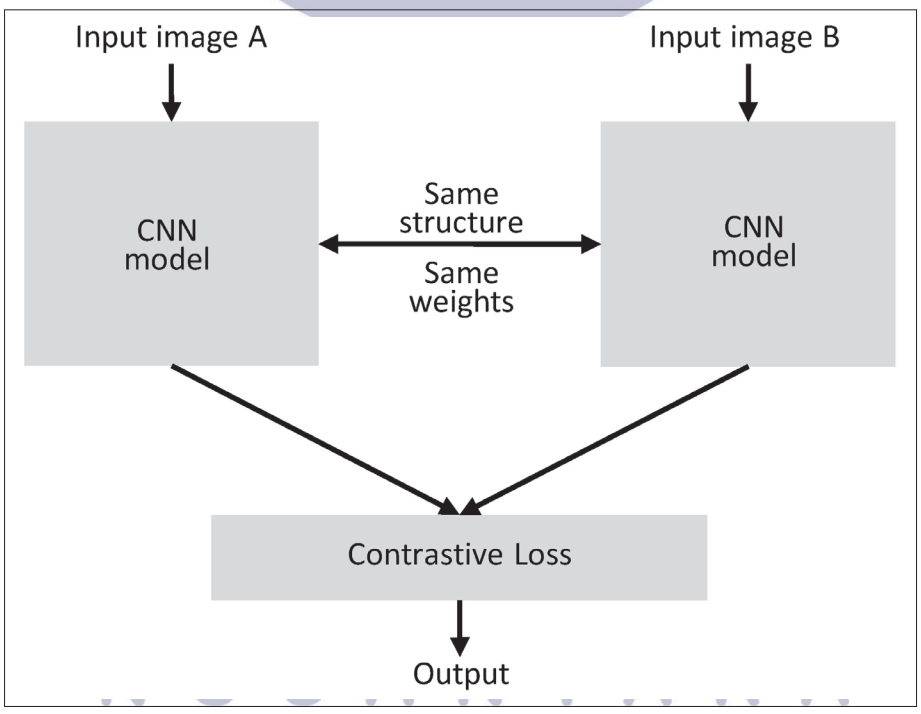
(a) Y_i : keluaran untuk *output layer* ke-i

(b) Y_in_i : masukan untuk *node layer* ke-i

(c) M : semua masukan untuk *output layer* sejumlah m buah

2.3 Siamese Convolutional Neural Network

Siamese Convolutional Neural Network adalah salah satu pendekatan yang termasuk dalam teknik deep learning. Pada awal tahun 1990, Bromley dan LeCun memperkenalkan Siamese Convolutional Neural Network sebagai metode untuk memverifikasi tanda tangan [17]. Siamese Convolutional Neural Network digunakan untuk mempelajari metrik kesamaan dengan melatih jaringan menggunakan convolutional neural network yang memiliki bobot yang sama[18].Siamese Convolutional Neural Network menerima dua input untuk membandingkan kedua pola tersebut, dan menghasilkan satu output yang menunjukkan nilai kesamaan antara kedua pola tersebut. Berikut merupakan gambar dari arsitektur *Siamese Convolutional Neural Network*.



Gambar 2.5. Arsitektur Siamese Convolutional Neural Network

2.3.1 Triplet Loss

Triplet Loss merupakan sebuah fungsi yang digunakan dalam membuat model Siamese Convolutional Neural Network. Metode triplet loss memanfaatkan input sebanyak 3 buah identik. Komponen dari triplet loss adalah terdapat 3 pasang gambar atau disebut triplet. Triplet terdiri dari tiga gambar yaitu anchor, positive, dan negative. Anchor merupakan gambar acuan, positive merupakan gambar yang sama dengan anchor, sedangkan negative merupakan gambar yang berbeda dari anchor dan positive. Dari gambar tersebut akan dihitung jarak antara embedding anchor dengan positive lebih kecil dari jarak embedding antara anchor dengan negative[19].

2.4 Resnet 50

Resnet 50 merupakan salah satu arsitektur Convolutional Neural Network. Resnet 50 dikembangkan oleh Kaiming He, Xiangyu Zhang , Shaoqing Ren dan Jian Sun pada tahun 2015[20]. ResNet merupakan singkatan dari Residual Network yang sering disebut Deep Residual Network. Arsitektur ini dibangun karena terdapat masalah ketika pelatihan deep learning, membutuhkan waktu yang cukup lama. Solusi dari permasalahan tersebut maka Resnet mengusulkan untuk menerapkan skip connection atau shortcut.

2.5 Optimizer Adam

Adam adalah sebuah optimizer yang merupakan pengembangan dari algoritma Stochastic Gradient Descent (SGD). Perkembangan ini terutama terjadi pada pembaharuan bobot jaringan. Algoritma Adam pertama kali diperkenalkan oleh Diederik Kingma [21].

Dalam penggunaan optimizer Adam, penting untuk menentukan ukuran learning rate saat melatih dataset. Learning rate adalah salah satu parameter yang digunakan untuk menghitung perubahan bobot selama proses pelatihan. Menentukan nilai learning rate dapat menjadi tugas yang sulit karena jika nilai learning rate terlalu kecil, waktu pelatihan akan menjadi lama. Namun, jika nilai learning rate terlalu besar, waktu pelatihan akan menjadi lebih cepat tetapi hasilnya mungkin kurang optimal karena penyebaran bobot terjadi terlalu cepat [22].

2.6 Metriks Evaluasi

Metriks evaluasi yang digunakan untuk klasifikasi terhadap model yang telah dibuat yaitu Accuracy, F-1 Score, Precision, dan Recall. Berikut penjelasan secara detail:

2.6.1 Accuracy

Accuracy adalah prediksi rasio yang benar (True Positive plus True Negative) dibagi dengan jumlah total prediksi yang dibuat (True Positive plus True Negative plus False Positive plus False Negative)[23].

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

2.6.2 Precision

Precision mengukur ketepatan classifier. Semakin tinggi precision maka lebih sedikit false positives, sedangkan precision yang lebih rendah berarti lebih banyak false positives. precision dihitung sebagai True Positive dibagi dengan jumlah True Positive ditambah False Positive [23].

$$precision = \frac{TP}{(TP + FP)} \tag{2.7}$$

2.6.3 Recall

Recall mengukur kelengkapan, atau sensitivitas, dari classifier. Recall yang lebih tinggi berarti lebih sedikit false negatives, sedangkan Recall yang lebih rendah berarti lebih banyak false negatives. Recall dihitung sebagai True Positive dibagi dengan jumlah True Positive ditambah False Negative[23].

$$Recall = \frac{TP}{(TP + FN)} \tag{2.8}$$

2.6.4 F1-Score

Precision dan Recall dapat digabungkan untuk menghasilkan metrik tunggal yang disebut sebagai F1-Score. F1-score yang merupakan rata-rata harmonik

yang mempertimbangkan nilai dari Precision dan Recall[23]. F1-Score memiliki rentang nilai antara 0 hingga 1, dimana nilai 1 menunjukkan performa terbaik dan nilai 0 menunjukkan performa terburuk. Perhitungan Skor F1 dilakukan dengan menggunakan rumus berikut:

$$F1 = 2x\frac{(precisionrecall)}{(precision + recall)} \tag{2.9}$$