BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Dalam pembahasan ini, peneliti akan memaparkan beberapa studi yang terkait dengan kerusakan pada kerusakan bodi mobil. Berikut adalah beberapa penelitian yang relevan dengan topik ini:

Table 2.1 Penelitian terdahulu

Peneliti (Tahun)	Judul Penelitian	Metode Penelitian	Kesimpulan
Kalpesh Patil, Mandar Kulkarni, Shirish Karande (2017)	*Deep learning Based Car Damage Classification*	*Transfer Learning* dan *Ensemble Learning*	• *Transfer learning* lebih baik daripada *fine-tuning* berdomain spesifik. • Mencapai akurasi 89,5% dengan kombinasi transfer dan *ensemble learning.*
Karthik V, Rakshata P, Padma H V, Pooja M, Yashaswini H V (2019)	*Car Damage Detection and Analysis Using Deep learning Algorithm For Automotive*	CNN dan *Mask* RCNN untuk segmentasi	• Penelitian ini menyimpulkan bahwa, CNN terus berkembang dengan arsitektur inovatif. *Mask*-RCNN memungkinkan deteksi yang lebih tepat dan akurat

Peneliti (Tahun)	Judul Penelitian	Metode Penelitian	Kesimpulan
			sebagai evolusi deteksi objek.
Phyu Mar Kyu, Kuntpong Woraratpanya (2020)	*Car Damage Detection and Classification*	*Deep learning-based algorithm*, VGG 16 dan VGG 19	• Performansi dari VGG-19 Lebih baik dari pada VGG-16 dalam hal deteksi dan klasifikasi kerusakan pada mobil.
Cesar G. Pachón Suescún, Javier O. Pinzón Arenas, Robinson Jiménez Moreno (2019)	*Detection of Scratches on Cars by Means of CNN and R-CNN*	Algoritma berdasarkan jaringan saraf konvolusional dan varian penggunaan wilayah tersebut (CNN dan R-CNN)	• R-CNN memiliki persentase presisi sebesar 98,3%, sedangkan CNN mencapai 96,89%. • Waktu pemrosesan R-CNN dan CNN adalah 1,6563 dan 1,264 detik masing-masing.
Mahavir Dwivedi, Hashmat Shadab Malik, S. N. Omkar, Edgar Bosco Monis, Bharat	*Deep learning-Based Car Damage Classification and Detection*	Penggunaan model CNN yang telah dilatih sebelumnya pada kumpulan data *Image-Net*, serta detektor objek YOLO.	• Mencapai akurasi tertinggi sebesar 96,39%. • Deteksi wilayah kerusakan menggunakan detektor objek YOLO dengan

Peneliti (Tahun)	Judul Penelitian	Metode Penelitian	Kesimpulan
Khanna, Satya Ranjan Samal, Ayush Tiwari, dan Aditya Rathi (2021)			skor peta maksimum.

Dari table 2.1, dapat disimpulkan bahwa beberapa teknik seperti Pemindahan Pembelajaran, Mask R-CNN, algoritma berbasis Pembelajaran Mendalam, VGG 16, VGG 19, dan YOLO mampu mengidentifikasi kerusakan pada bodi mobil. Pemindahan Pembelajaran menonjol dengan tingkat akurasi tertinggi, mencapai 89,5% melalui gabungan pemindahan dan pembelajaran kolektif. Meskipun Jaringan Saraf Konvolusional (CNN) terus berkembang, Mask-RCNN memberikan deteksi objek lebih akurat, dan VGG-19 menunjukkan kinerja lebih baik dibandingkan VGG-16 dalam mengidentifikasi dan mengklasifikasikan kerusakan. R-CNN memiliki tingkat presisi lebih tinggi daripada CNN, tetapi memerlukan waktu pemrosesan yang lebih lama. YOLO, meskipun efektif dalam mengidentifikasi area kerusakan, memiliki beberapa kekurangan yang perlu diperbaiki. Beberapa penelitian kurang memberikan informasi rinci tentang dataset, mempengaruhi generalisasi hasil (kemampuan model untuk memberikan prediksi atau kinerja yang baik pada data yang tidak digunakan selama pelatihan), dan fokus pada jenis kerusakan tertentu dapat membatasi validitas eksternal terkait keragaman kerusakan mobil.

Selain itu, parameter seperti Variasi pembagian data, jumlah epoch, jumlah lapisan konvolusi, jenis *pooling*, ukuran kernel, learning rate, dan fungsi aktivasi yang digunakan tidak diperhitungkan dalam hasil penelitian. Oleh karena itu, penelitian mendatang direkomendasikan untuk menerapkan metode CNN dan menguji tingkat akurasi dengan memvariasikan parameter tersebut. Hal ini diharapkan dapat memberikan pemahaman yang lebih mendalam

tentang faktor-faktor yang mempengaruhi kinerja sistem dalam deteksi kerusakan pada bodi mobil.

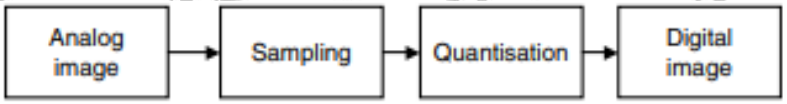
2.2 Bodi Mobil

Bodi mobil adalah kerangka eksternal atau struktur luar dari kendaraan bermotor yang meliputi bagian depan, samping, belakang, serta atap kendaraan. Bodi mobil bertujuan untuk memberikan perlindungan kepada penumpang dari elemen-elemen luar seperti hujan, angin, debu, dan potensi bahaya di jalan[14].

2.3 *Digital image* (Gambar Digital)

Gambar merupakan fungsi dua dimensi (2D) yang mewakili ukuran dari beberapa karakteristik seperti kecerahan atau warna pemandangan yang dilihat. Gambar adalah proyeksi pemandangan 3D pada bidang proyeksi 2D. Ini dapat didefinisikan sebagai fungsi dari dua variabel *f(x, y)*. Untuk setiap bidang proyeksi *(x, y)*, *f(x, y)* menentukan intensitas cahaya pada titik tersebut. Intensitas koordinat tersebut adalah *x* dan *y* dari citra tingkat keabuan (*grayscale*).[8].

Gambar digital terdiri dari bagian-bagian kecil yang disebut piksel. Setiap piksel merepresentasikan kecerahan pada satu titik. Proses mengubah gambar dari bentuk analog ke digital melibatkan pengambilan sampel dan pengelompokan nilai, yang disebut kuantisasi [15]. Hal ini ditunjukkan pada Gambar 2.1.



Gambar 2.1 Gambar digital dari Gambar Analog

Pembuatan dan manipulasi gambar digital memiliki keunggulan dalam kecepatan, efisiensi, dan biaya lebih rendah. Gambar-gambar ini mudah disimpan, dipindahkan, serta disalin tanpa kehilangan kualitas. Pemrosesan citra digital adalah cabang dari ilmu komputer yang bertujuan meningkatkan

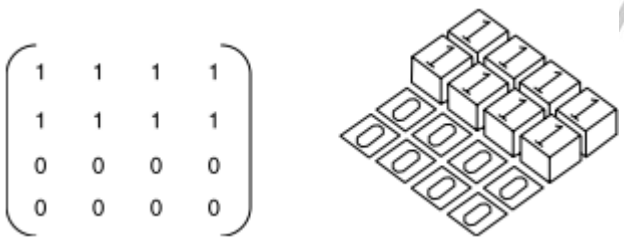
kualitas gambar, mengekstrak informasi, dan membuatnya lebih sesuai untuk analisis manusia atau sistem komputer.

Pemrosesan citra digital melibatkan teknik seperti penghapusan derau, peningkatan kontras, segmentasi, pengenalan pola, karakter, kompresi, analisis *spasial*, *spektroskopi*, dan pengolahan gambar 3D untuk animasi. Komputer digital yang fleksibel memungkinkan pengaturan ulang tanpa mengubah perangkat keras, ideal untuk pemrosesan adaptif sinyal gambar.

Keuntungan lainnya termasuk efisiensi penyimpanan dan transmisi data yang didukung oleh algoritma kompresi gambar. Representasi gambar digital menggunakan algoritma untuk mengubah gambar menjadi format digital yang berguna dalam berbagai bidang seperti pengenalan pola, penginderaan jauh, penajaman gambar, pemrosesan warna dan video, dan aplikasi medis.

Digital Image Representation (DIP) adalah proses gambar digital melalui berbagai algoritma dan telah diterapkan dalam berbagai bidang seperti pengenalan pola, pengindraan jauh, penajaman gambar, pemrosesan warna dan video, serta aplikasi medis.

Direpresentasikan sebagai sinyal diskrit dua dimensi dalam bentuk matriks elemen N × N, dimana setiap elemen matriks mewakili intensitas sampel gambar. Sebagai contoh, representasi gambar 4×4 dalam suatu matriks dapat ditunjukan pada tampilan tiga dimensi pada Gambar 2.2.



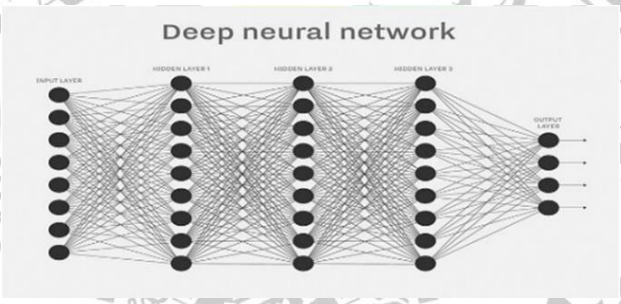
Gambar 2.2 Representasi Gambar Digital (Medium Samuel Sena, 2017)

Konversi gambar ke format digital dapat dilakukan dengan kamera digital atau scanner. Gambar digital dapat dibuat langsung di layar komputer. Namun,

terbatas pada koordinat *spasial* (*sampling*) dan intensitas yang diperbolehkan (kuantisasi).

2.4 Pembelajaran Mendalam *(Deep learning)*

Deep learning merupakan jenis jaringan saraf tiruan yang digunakan untuk mempelajari struktur dan fungsi dari cara kerja otak manusia. Deep learning adalah proses pembelajaran dari Deep Neural Network (DNN) dengan tiga lapisan atau lebih (*input* layer, N > hidden layers, *output* layer). Keuntungan dari deep learning adalah hidden layers dapat mengubah data yang tidak dapat dipisahkan secara linier menjadi data yang dapat dipisahkan secara linier.



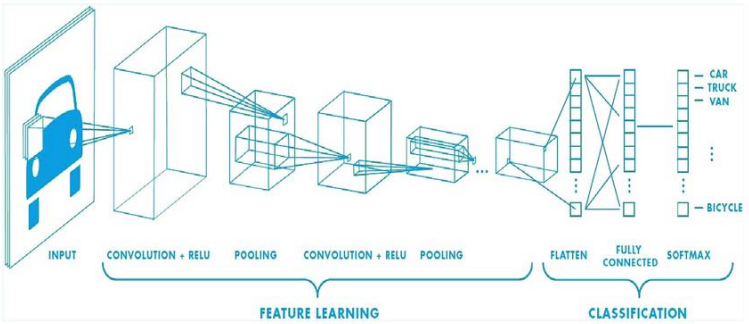
Gambar 2.3 *Deep Neural Network* (Medium Samuel Sena, 2017)

Terinspirasi oleh sistem biologis neuron, jaringan saraf dalam memiliki banyak lapisan pemrosesan nonlinier menggunakan elemen sederhana yang beroperasi secara paralel. Ini terdiri dari lapisan *input,* beberapa lapisan tersembunyi (*Hidden Layer*), dan lapisan *output* melalui lapisan node atau neuron yang saling berhubungan. *Deep learning* bekerja dengan menggunakan beberapa algoritma tertentu, seperti *Convolutional Neural Network, Recurrent Neural Network*, *Long Short-Term Memory* dan *Self Organizing Maps*. Algoritma dianggap tidak lengkap dan memiliki kemampuan yang berbeda-beda, oleh karena itu, developer menggunakan deep learning dengan algoritma yang sesuai dengan kebutuhannya.

2.5 *Convolusional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan arsitektur jaringan saraf tiruan yang telah merevolusi pemrosesan gambar dan pengenalan pola. Ini

menggunakan lapisan konvolusi dan *pooling* untuk mengekstraksi fitur-fitur penting dari data gambar. Setelah konvolusi dengan filter, dilanjutkan dengan fungsi aktivasi ReLU, diikuti oleh proses *pooling*. Proses ini terus diulang untuk mendapatkan peta fitur yang diteruskan ke fully connected Neural Network untuk menghasilkan *output* kelas. CNN sangat efektif dalam klasifikasi gambar, deteksi objek, dan pengenalan wajah.



Gambar 2.4 *Convolusional Neural Network* (Medium Samuel Sena, 2017)

Dari gambar 2.4 terlihat bahwa tahap pertama dari arsitektur CNN adalah tahap konvolusi. Langkah ini dilakukan dengan menggunakan sebuah kernel dengan ukuran tertentu. Banyaknya jumlah kernel yang digunakan bergantung pada jumlah fitur yang dihasilkan. Kemudian dilanjutkan menuju proses fungsi aktivasi, biasanya menggunakan fungsi aktivasi ReLU (*Rectifier Linear Unit*) Setelah keluar dari proses fungsi aktivasi, maka dilanjutkan dengan proses *pooling*. Proses ini diulangi beberapa kali hingga terdapat cukup peta fitur untuk diteruskan ke lapisan terhubung penuh (*fully connected Neural Network*), dan dari fully connected Network adalah *output* class.

2.5.1 Convolution Layer

Convolution layer adalah bagian dari tahapan dalam arsitektur CNN. Tahap ini melakukan operasi konvolusi pada keluaran dari lapisan sebelumnya. Lapisan ini merupakan proses utama yang mendasari arsitektur jaringan CNN. Konvolusi adalah istilah matematika untuk penerapan berulang satu fungsi ke keluaran fungsi lain. Operasi konvolusi melibatkan dua fungsi argumen bernilai nyata. Proses ini fitur keluaran yang disebut peta fitur dari citra *input*. *Input* dan

output ini dapat dianggap sebagai dua argumen yang bernilai nyata. Operasi konvolusi dapat dijelaskan sebagai berikut:

$$s(t) = (x * t)(t) = \sum_{\alpha=-\infty}^{\infty} x(\alpha) * w(t - \alpha) \quad (2.1)$$

Keterangan :

S(t) = Fungsi hasil operasi konvolusi

X = *Input*

W = bobot (kernel)

Fungsi s(t) memberikan satu keluaran dalam bentuk peta fitur. Argumen pertama adalah *input* x dan argumen kedua w adalah kernel atau filter. Mengingat masukan sebagai citra dua dimensi, kita dapat mengganti t dengan i dan j sebagai piksel. Maka dari itu, untuk masukan dengan lebih dari satu dimensi dapat ditulis sebagai berikut :

$$s(i,j) = (K * I)(i,j) = \sum_{\infty}\sum_{\infty} I(i - m, j - n)K(m,n) \quad (2.2)$$

$$s(i,j) = (K * I)(i,j) = \sum_{\infty}\sum_{\infty} I(i + m, j + n)K(m,n) \quad (2.3)$$

Berdasarkan kedua persamaan di atas, ini adalah perhitungan dasar dalam operasi konvolusi, dimana i dan j adalah piksel dari citra. Perhitungan ini bersifat kumulatif dan terjadi jika K yang merupakan kernel, dan kemudian I adalah *input*nya, yang kemudian dapat dibalikkan secara relatif terhadap masukan. Alternatifnya, operasi konvolusi dapat dilihat sebagai perkalian matriks antara gambar masukan dan kernel, dimana keluarannya dihitung melalui perkalian titik. Selain itu, penentuan volume *output* juga dapat ditentukan oleh masing-masing lapisan dengan menggunakan *hyper-parameter*. *Hyper-parameter* yang digunakan dalam persamaan berikut digunakan untuk menghitung jumlah neuron yang diaktifkan dalam satu *output*. Perhatikan persamaan berikut.

$$Output = \frac{W - F + 2P}{S} + 1 \quad (2.4)$$

Keterangan :

W = Ukuran volume gambar

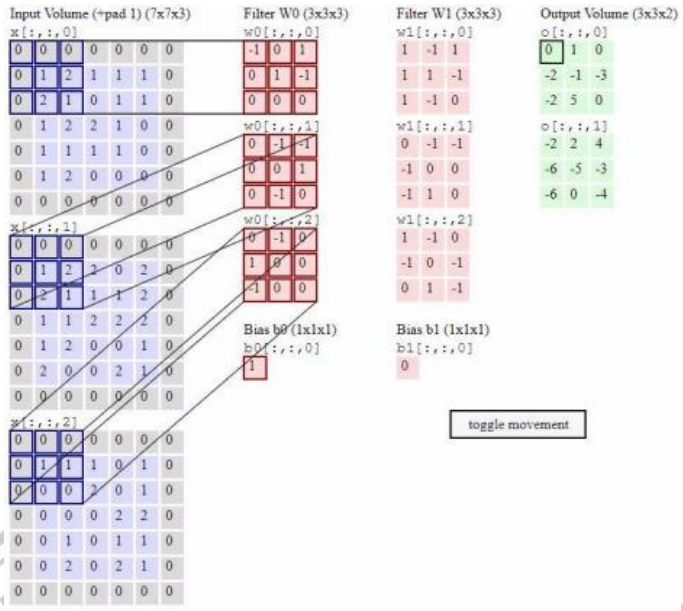
F = Ukuran Filter

P = Nilai Padding yang digunakan

S = Ukuran Pergeseran (*Stride*)

Berdasarkan persamaan di atas, kita dapat menghitung dimensi spasial dari volume *output* dengan menggunakan *hyper-parameter* berikut: ukuran volume (w), filter (f), *stride* yang digunakan (s), dan jumlah *padding* nol yang diterapkan (p). *Stride* adalah nilai yang digunakan untuk memfilter gambar masukan, sedangkan *zero-padding* adalah nilai yang digunakan untuk menambahkan angka nol di sekitar tepi gambar.

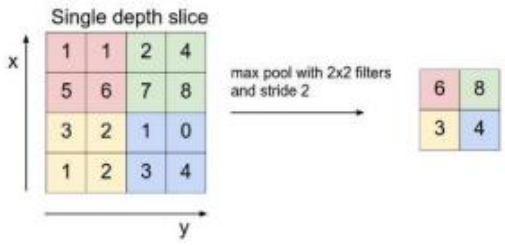
Operasi *Convolutional Layer* terdiri dari neuron yang membentuk sebuah filter dengan panjang dan tinggi (dalam piksel). Misalnya, lapisan pertama pada lapisan ekstraksi fitur umumnya merupakan Convolutional Layer yang berukuran 5x5x3, yang artinya memiliki panjang 5 piksel, tinggi 5 piksel, dan 3 saluran (*channel*) tergantung dengan gambar *input*. Ketiga filter ini akan digeser melalui seluruh bagian gambar. Setiap pergeseran akan melibatkan operasi "titik" antara nilai masukan dan nilai filter tersebut, menghasilkan keluaran yang sering disebut peta aktivasi atau peta fitur. Seperti pada gambar 2.5 berikut:



Gambar 2.5. *Convolution Layer* (Medium Samuel Sena, 2017)

2.5.2 Operasi *pooling*

Pooling adalah proses mengurangi ukuran matriks dengan menggunakan operasi *pooling*. Layer *pooling* umumnya ditempatkan setelah layer konvolusional. Pada dasarnya, layer *pooling* terdiri dari sebuah filter dengan ukuran dan langkah tertentu yang bergeser secara bergantian di dalam seluruh area *feature-map*. Terdapat dua jenis *pooling* yang umum digunakan, yaitu *average pooling* dan *max pooling*. Pada *average pooling*, nilai yang diambil adalah nilai rata-rata, sedangkan pada *max pooling*, nilai maksimal yang akan diambil. Dengan menempatkan lapisan *pooling* secara berulang di antara lapisan konvolusi dalam arsitektur model CNN ukuran volume keluaran peta fitur akan berkurang secara bertahap, sehingga akan mengurangi jumlah parameter dan komputasi dalam jaringan untuk mengontrol *overfitting*. Layer *pooling* pada umumnya menggunakan filter berukuran 2x2 yang diterapkan dengan langkah sebanyak dua dan beroperasi pada tiap segmen masukan. Berikut merupakan contoh gambar operasi max *pooling*:

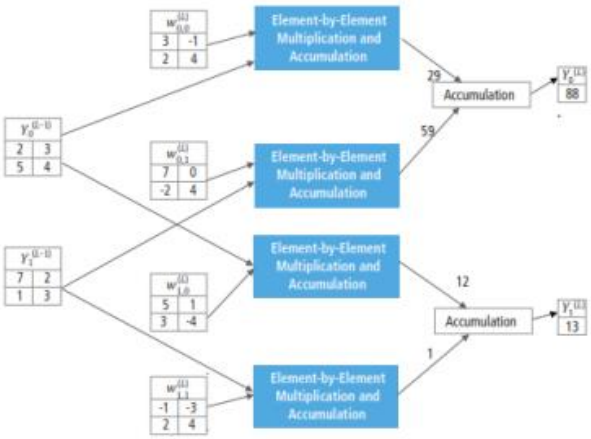


Gambar 2.6 Operasi Max-*Pooling* (Medium Samuel Sena, 2017)

Seperti terlihat pada gambar 2.6, gambar tersebut menunjukan proses dari max-*pooling*. Hasil dari proses *pooling* berupa sebuah matriks dengan dimensi yang lebih kecil dari dengan citra aslinya. Lapisan *pooling* di atas memproses tiap irisan kedalaman volume masukan secara bergantian. Jika dilihat pada gambar di atas, operasi max-*pooling* menggunakan filter berukuran 2x2. *Input* pada proses ini berukuran 4x4, kemudian dari masing-masing angka pada *input* 4x4, operasi tersebut akan mengambil nilai maksimumnya, setelahnya dilanjutkan untuk menciptakan matriks keluaran baru dengan ukuran 2x2.

2.5.3 Fully-Conected Layer

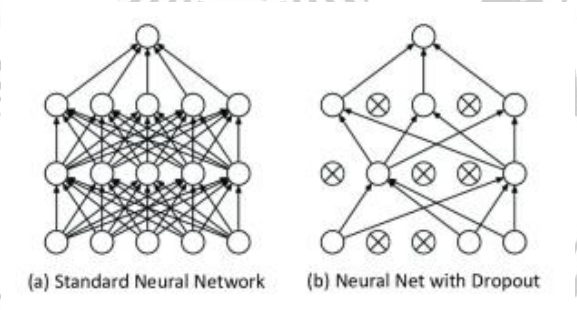
Fully-Connected Layer adalah lapisan di mana setiap neuron aktivasi dari lapisan sebelumnya terhubung ke setiap neuron di lapisan berikutnya, serupa dengan Neural Network biasanya. Lapisan ini umumnya digunakan dalam MLP (*Multi-Layer Perceptrons*) untuk melakukan transformasi data dimensional sehingga data dapat diklasifikasikan secara linier. Perbedaan utama antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah neuron dalam lapisan konvolusi hanya terhubung ke wilayah tertentu pada masukan, sedangkan lapisan *Fully-Connected* memiliki koneksi keseluruhan. Namun keduanya tetap mengoperasikan produk ‘titik’, sehingga tidak terlalu banyak perbedaan dalam fungsinya. Berikut adalah langkah-langkah proses lapisan terhubung penuh:



Gambar 2.7. Processing of a Fully-Connected Layer (Medium Samuel Sena, 2017)

2.5.4 Droput Regulation

*Dropou*t adalah sebuah teknik regulasi jaringan saraf yang memiliki tujuan untuk memilih beberapa neuron secara acak dan mengabaikannya selama proses pelatihan. Dengan kata lain, beberapa neuron ini dinonaktifkan secara acak. Artinya kontribusi neuron yang dinonaktifkan dalam jaringan akan ditangguhkan dan tidak ada bobot baru yang diterapkan selama backpropagation. Berikut adalah ilustrasi dari proses dropout:



Gambar 2.8. *Dropout Regulation*[4]

Berdasarkan gambar 2.8 di atas, pada bagian (a) memiliki jaringan saraf biasa dengan dua *hidden layer*. Bagian (b), disisi lain, menunjukkan jaringan saraf yang menggunakan dropout. Pada gambar tersebut, tampak beberapa neuron aktivasi yang tidak lagi digunakan. Teknik ini sangat mudah untuk

diterapkan pada model CNN dan mempengaruhi performa model selama pelatihan serta dapat mengurangi *overfitting* [16].

Dalam jaringan saraf tiruan normal, yl dapat diambil sebagai nilai keluaran dari suatu lapisan l, dan zl sebagai nilai masukan pada lapisan l, serta Wl dan bl sebagai bobot dan bias dari lapisan l. Untuk unit ke-i, proses feedforward dapat dihitung dengan fungsi aktivasi f seperti pada persamaan (2.5).

$$z_i^{(l+1)} = W_i^{(l+1)} y^l + b_i^{(l+1)}$$

$$y_i^{l+1} = f\left(z_i^{(l+1)}\right) \tag{2.5}$$

Sementara pada jaringan yang menerapkan teknik dropout, variabel rl melambangkan vektor sepanjang j yang menyimpan nilai yang diambil dari distribusi Bernoulli. Pemrosesan feedforward dilakukan dengan menggunakan persamaan (2.6).

$$y^{\sim l} = r_j^l * y^l$$

$$Z_i^{l+1} = W_i^{(l+1)} y^l + b_i^{(l+1)}$$

$$y_i^{l+1} = f\left(z_i^{(l+1)}\right) \tag{2.6}$$

2.5.5 Sigmoid Classiefer

"*Sigmoid Classifier*" dapat merujuk pada jenis pengklasifikasi biner yang menggunakan fungsi *sigmoid* (fungsi logistik) untuk membuat prediksi. Fungsi *sigmoid* mengonversi nilai masukan ke dalam rentang antara 0 dan 1, yang dapat diinterpretasikan sebagai probabilitas. Fungsi *sigmoid*, sering ditulis sebagai σ(z), diberikan oleh rumus:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

Di sini, "z" mewakili kombinasi linear antara fitur masukan dan bobot yang terkait. Fungsi sigmoid memampatkan masukan ke dalam rentang (0, 1), yang dapat diinterpretasikan sebagai probabilitas bahwa suatu masukan termasuk ke dalam kelas positif. Selama fase pelatihan, model mempelajari bobot optimal yang meminimalkan fungsi kerugian yang sesuai, biasanya berupa kerugian logistik atau kerugian entropi silang. Ini melibatkan gradien turun atau teknik optimisasi lainnya. Pengklasifikasi sigmoid menggunakan batas keputusan, dimana jika *σ(z)* lebih besar atau sama dengan 0,5, masukan diklasifikasikan sebagai kelas positif, dan jika *σ(z)* kurang dari 0,5, masukan diklasifikasikan sebagai kelas negatif. Saat melakukan prediksi, Anda menghitung *σ(z)* untuk masukan yang diberikan, dan berdasarkan hasilnya, Anda mengklasifikasikan masukan ke salah satu dari dua kelas.

2.5.6 Adam Optimizer

Algorthtms and Adaptive Moment Estimation (Adam) adalah suatu optimizer yang diperuntukkan untuk mempelajari dependensasi sementara dan mengkestaksi kesalahan fitur [17] Adam optimizer adalah suatu algoritma yang menampilkan secara bertahap akan optimasi suatu fungsi objek, dengan gradien dan parameternya adalah

$$\theta t + 1 = \theta t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \hat{m}_t, \tag{2.8}$$

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

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

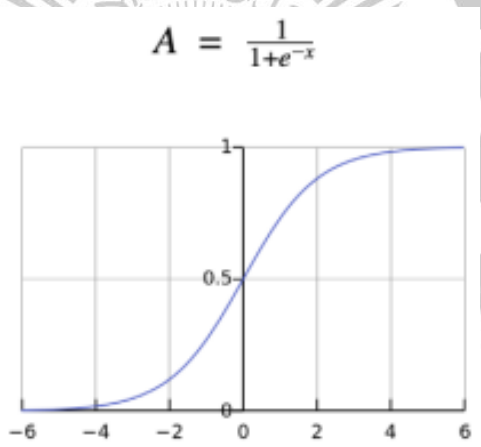
Yang dimana θ_t, θ_{t+1} merupakan fungsi dari objek, t merupakan parameter waktu, $\beta_1, \beta_2 \in [0,1)$ merepresentasikan pergerakan dari perubahan indeks, η menampilkan tingkat pembelajaran, ε adalah parameter konstan yang miliki yaitu 0.9999, sementara $\hat{m}_t$ dan $\hat{v}_t$ adalah order pertama dan order kedua dari estimasi moment setelah perubahan dan pembacaan yang dilakukan oleh algoritma adam

2.6 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi yang menggambarkan hubungan antara tingkat aktivitas internal (fungsi penjumlahan) dan dapat berbentuk linier ataupun *non-linier*. Fungsi ini digunakan untuk mengetahui apakah suatu neuron diaktifkan atau tidak. Menurut Samuel Sena dalam artikel yang diunggah ke website Medium (Sena, Pengenalan *Deep learning Part* 1: Jaringan Saraf, 2018) dijelaskan bahwa ada beberapa fungsi aktivasi yang umum digunkan dalam jaringan saraf yaitu sebagai berikut:

2.6.1 Fungsi Aktivasi *Sigmoid*

Fungsi aktivasi *sigmoid* merupakan fungsi *non-linier*. Masukan dari fungsi aktivasi ini merupakan bilangan nyata atau *real*, dan keluaran dari fungsi tersebut berkisar dari 0 (nol) hingga 1 (satu). Berikut adalah grafik fungsi aktivasi *sigmoid*.:



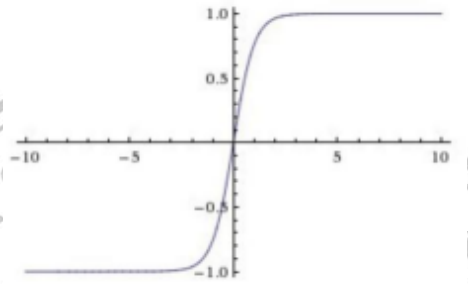
Gambar 2.9.Fungsi Aktivasi Sigmoid

Jika masukan dari suatu node pada jaringan saraf bernilai negatif maka keluaran yang diperoleh adalah 0, namun jika masukannya positif maka keluarannya adalah 1. Akan tetapi fungsi ini memiliki kelemahan, yaitu gradien dapat dimatikan oleh sigmoid ketika aktivasi neuron menghasilkan nilai pada rentang 0 atau 1 dan gradiennya cenderung mendekati nol. Dalam hal ini, keluaran dari sigmoid tidak berpusat pada nol (*zero-centered*).

2.6.2 Fungsi Aktivasi *Tanh*

Fungsi aktivasi *Tanh* adalah fungsi nonlinier. Masukan untuk fungsi aktivasi ini adalah bilangan *real*, dan keluaran dari fungsi ini berkisar antara -1 hingga 1. Berikut adalah grafik dari fungsi aktivasi *Tanh*.:

$$tanh(x) = 2\sigma(2x)-1$$

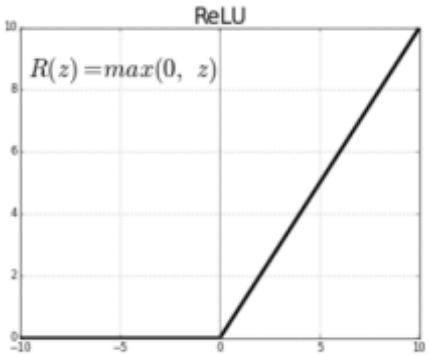


Gambar 2.10 Fungsi Aktivasi Tanh

Seperti fungsi *sigmoid*, fungsi ini juga memiliki kelemahan yaitu gradien dapat dimatikan. Namun fungsi ini mempunyai kelebihan, yaitu keluaran dari fungsi *Tanh* adalah berpusat pada nol (*zero-centered*). Fungsi *Tanh* sering kali merupakan pilihan yang lebih baik dibandingkan dengan fungsi *sigmoid* untuk aplikasi. Perlu diketahui bahwa fungsi *Tanh* merupakan pengembangan dari fungsi *sigmoid*.

2.6.3 Fungsi Aktivasi ReLU

Pada dasarnya fungsi ReLU (Rectified Linear Unit) menjalankan "*threshold*" dari 0 hingga tak terhingga. Fungsi ini telah menjadi salah satu fungsi terpopuler saat ini. Di bawah ini adalah grafik fungsi aktivasi ReLU:



Gambar 2.11 Fungsi aktivasi ReLU

Dalam fungsi ini, masukan neuron adalah bilangan negatif. Oleh karena itu, fungsi ini mengubah nilai ini menjadi nilai 0, dan jika masukannya bernilai positif, maka keluaran dari neuron akan menjadi nilai aktivasi itu sendiri. Fungsi aktivasi ini memiliki keunggulan berupa *Stochastic Gradient Descent* (SGD). Jika dibandingkan dengan fungsi *sigmoid* dan *tanh*, ReLU dapat mempercepat proses konfigurasi. Namun aktivasi ini juga memiliki kelemahan. Aktivasi ini bisa menjadi rentan selama proses pelatihan dan dapat mengakibatkan kematian unit.

2.7 *Library*

Perkembangan dibidang teknologi sangatlah membantu pekerjaan manusia sehari hari, termasuk *Deep Learning* (DL) yang saat ini dipermudah oleh banyaknya *library* dan *Application Program Interface* (API).

2.7.1 Listdir

Listdir adalah salah satu *library* yang digunakan dalam percobaan kali ini yang dimana *listdir* sendiri berguna untuk melakukan pembuatan direktori yang akan memindah dan memindai semua data yang akan digunakan menjadi satu direktori sehingga dapat mempermudah dalam pembacaan citra yang akan diklasifikasikan

2.7.2 Tensorflow

Tensorflow merupakan suatu antarmuka yang berguna untuk mengekspresikan algoritma pembelajaran mesin dan untuk mengeksekusi perintah dengan menggunakan informasi yang dimiliki oleh suatu objek serta dapat juga dengan baik membedakan perbedaan antar objek satu dengan objek lainnya. *Tensorflow* sendiri mempunyai fitu yang dapat dijalankan di GPU (*Graphic Processing Unit*) maupun CPU (*Central Processing Unit*) [18].

2.7.3 Numpy

Numpy adalah salah satu dari *library* utama yang dimiliki oleh Bahasa *pyhton*, yang dimana memainkan peran yang sangat penting di penelitian berbagai bidang seperti fisika, kimia maupun astronomi [19]. *NumPy* adalah suatu data struktur yang secara efisien menyimpan dan mengakses data secara multidimensi [20], sehingga menyimpan sementara hasil suatu data yang sedang digunakan dalam suatu prosess.

2.7.4 Seaborn

Seaborn adalah salah satu *open source* dari *python* yang berguna untuk menampilkan data menggunakan Bahasa *python*. *Seaborn* akan sangat bermanfaat Ketika memvisualisasikan suata data frame yang banyak untuk digunakan meneganalsis data dengan jumlah yang banyak [21].

2.7.5 OpenCV

OpenCV berawal dari intel pada tahun 1999 yang dikemukakan oleh Gary Bradsky yang pertama kali diluncurkan pada tahun 2000. *OpenCV* berguana untuk menjadi system pendukung algoritma yang berhubungan dengan *computer vision* dan *machine learning* [22].

2.7.6 Keras

Keras merupakan salah satu *library* pembelajaran *pyhton* yang dimana fokus utama keras adalah membantu pembuatan prototype dengan cepat serta melakukan percobaan secara cepat pula. Hal ini sangatlah membantu para

peneliti untuk melakukan peneleitian secara singkat dan mudah. Keras memiliki klaim terkuat untuk meenjadi pilihan *framework* untuk para pemula [23].