BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Quality Management System (QMS)

Menurut (Selvamuthu & Das, 2018), *Statistical Quality Control* (SQC) merupakan penerapan penting dari teknik statistik dalam industri manufaktur. Umumnya, industri manufaktur mendapatkan bahan baku dari *vendor*. Untuk memastikan kualitasnya, penting untuk memeriksa bahan baku sebelum mengambil keputusan untuk menerima atau menolaknya. Dalam banyak kasus, bahan baku tersedia dalam jumlah besar atau *batch*, sehingga memeriksa setiap *item* secara individu tidak mungkin dilakukan. Oleh karena itu, sampel dipilih secara acak dari batch untuk diperiksa. Hal ini seringkali memunculkan pertanyaan tentang berapa banyak item yang harus diambil sebagai sampel dan berapa banyak *item* cacat yang ditemukan dalam sampel tersebut sebelum *batch* ditolak. Untuk menjawab pertanyaan tersebut, dapat menggunakan teknik *sampling* penerimaan, yakni teknik statistik untuk pengambilan keputusan. Jika bahan baku tidak memenuhi standar, maka bisa dikembalikan kepada *vendor*, namun jika memenuhi standar, bisa diproses melalui proses manufaktur dan diubah menjadi sebuah produk. Proses manufaktur harus dijaga dengan baik untuk mengurangi variasi yang dapat diatribusikan, seperti bahan baku cacat, peralatan rusak, penanganan mesin yang tidak tepat, kelalaian operator, dan staf teknis yang tidak terampil. Untuk mengetahui apakah proses manufaktur terkendali atau tidak, dapat dilakukan melalui teknik yang dikenal sebagai diagram kontrol. Setelah produk selesai

diproduksi, mereka akan diperiksa menggunakan teknik statistik untuk memutuskan apakah akan menerima atau menolak produk tersebut.

2.1.2 ISO 9000

Definisi dari Standar ISO 9000 untuk sistem manajemen kualitas (*Quality Management System*, QMS) menurut (Gaspersz, 2001) adalah sebuah kerangka kerja yang mencakup struktur organisasi, tanggung jawab, prosedur, proses, dan sumber daya yang digunakan untuk menerapkan praktik-praktik manajemen kualitas. QMS terdiri dari serangkaian prosedur terdokumentasi dan standar praktik yang bertujuan memastikan bahwa proses atau produk (baik barang maupun jasa) sesuai dengan kebutuhan atau persyaratan yang telah ditetapkan oleh pelanggan dan organisasi.

Menurut (Milovanović dkk., 2023), ISO 9001 adalah standar referensi yang diterima secara internasional untuk sistem manajemen mutu (QMS), membantu perusahaan meningkatkan kualitas internal mereka untuk mencapai kepuasan pelanggan. Pada akhir tahun 2020, terdapat sekitar 916.842 sertifikat ISO 9001 yang valid. ISO 9001:2015 menetapkan persyaratan untuk QMS ketika sebuah organisasi perlu menunjukkan kemampuannya dalam menyediakan produk dan layanan yang memenuhi persyaratan pelanggan, undang-undang dan peraturanyang berlaku, serta bertujuan untuk meningkatkan kepuasan pelanggan melalui penerapan sistem efektif dan proses perbaikan. ISO 9000:2015 menekankan pentingnya organisasi mengakui baik pelanggan langsung maupun tidak langsung sebagai penerima nilai dari organisasi, dengan membedakan antara pelangganeksternal dan internal, yakni penerima produk dan layanan perusahaan, serta

karyawan perusahaan yang kebutuhan dan harapannya juga harus dipenuhi untuk meningkatkan kepuasan pelanggan eksternal. Mengacu pada 7 prinsip dari *Quality Management*, yang terdapat pada Gambar 2.1.

Gambar 2.1 Tujuh prinsip manajemen mutu (Al-rub & Shibhab, 2020).

Menurut (Abuhav, 2017), pemahaman terhadap konteks sebuah organisasi kini merupakan standar persyaratan. Meskipun bukan konsep baru, hal ini secara resmi diadopsi oleh Standar ISO 9001:2015. Ketika mengembangkan *Quality Management System*, penting untuk mengenali, menganalisis, dan memahami lingkungan bisnis di mana organisasi beroperasi dan menghasilkan produknya. Setiap organisasi terdiri dari berbagai entitas bisnis yang berbeda, baik internal maupun eksternal, yang saling terkait dan berinteraksi. Konteks organisasi mencerminkan interaksi dan integrasi entitas bisnis tersebut, serta merupakan rangkaian fungsi, faktor, proses, *input* dan *output*, serta kondisi dan batasan yang

menciptakan lingkungan bisnis organisasi, termasuk masalah internal dan eksternal. Masalah-masalah ini mempengaruhi kemampuan organisasi dalam menyediakan produk dan dengan demikian mempengaruhi tujuan dan objektifnya. Oleh karena itu, saat merumuskan strategi bisnis dan menetapkan tujuan *Quality Management System*, penting untuk memahami, mempertimbangkan, dan merujuk pada aspek-aspek ini. Memahami konteks organisasi adalah kunci untuk merancang strategi bisnis dan strategi kualitas yang tepat.

2.2 Machine Learning

Menurut (Khan dkk., 2018), *Machine Learning* adalah bagian dari *Artificial Intelligence* (AI) yang memungkinkan komputer untuk mengasimilasi informasi dari data tanpa memerlukan program yang spesifik. Artinya, tujuan dari *Artificial Intelligence* adalah untuk mengembangkan metode yang secara otomatis melakukan observasi terhadap dunia nyata atau biasa disebut "*training data*", tanpakebutuhan aturan atau logika eksplisit yang didefinisikan oleh manusia (pengajar/*supervisor*). Dalam hal ini, *Machine Learning* dapat dianggap sebagai proses pemrograman berdasarkan sampel data. Secara keseluruhan, *Machine Learning* membahas tentang bagaimana belajar untuk meningkatkan kinerja di masa mendatang berdasarkan pengalaman di masa lalu.

Menurut (Singh, 2019) , *Machine Learning* bisa digambarkan secara luas sebagai teknik komputasi yang memanfaatkan pengalaman untuk meningkatkan kinerja atau membuat prediksi yang akurat. Pengalaman ini mengacu pada data masa lalu yang tersedia bagi pembelajar, yang biasanya berbentuk data elektronik yang terkumpul dan siap untuk dianalisis. Data ini bisa berupa kumpulan *data*

training manusia yang sudah diubah menjadi format digital, atau informasi lain yang diperoleh melalui interaksi dengan lingkungan sekitar.

Proses *Machine Learning* melibatkan rancangan algoritma prediksi yang efisien dan akurat. Seperti dalam disiplin ilmu komputer lainnya, beberapa *matrix* penting untuk mengukur kualitas algoritma tersebut adalah kompleksitas waktu dan ruang. Namun, dalam konteks *Machine Learning*, kita juga perlu memperhatikan kompleksitas sampel untuk mengevaluasi seberapa besar sampel yang dibutuhkan agar algoritma dapat memahami konsep yang hendak dipelajari. Secara keseluruhan, keberhasilan pembelajaran teoritis suatu algoritma sangat bergantung pada kompleksitas kelas konsep yang sedang dipelajari dan ukuran sampel pelatihan yang tersedia.

2.2.1 Deep Learning

Deep Learning merupakan cabang dari *Artificial Intelligence* yang berfokus pada pembuatan model neural network yang besar dan mampu mengambil keputusan yang akurat berdasarkan data (Kelleher, 2019). *Deep Learning* sangat sesuai untuk digunakan dalam situasi di mana data yang dihadapi bersifat kompleks dan terdapat kumpulan data dalam jumlah yang besar. *Deep Learning* berfungsi untuk membuat prediksi sesuai dengan input yang masuk ke dalam sistem. Cara kerja *Deep Learning* mirip sekali dengan cara kerja otak manusia. Terdapat *hidden layer* yang berfungsi seperti hubungan antar saraf otak. Lapisan ini berguna untuk pembelajaran pada mesin, *Deep Learning* memiliki dua jenis algoritma seperti algoritma ANN (*Artificial Neural Network*) dan algoritma DNN (*Deep Neural*

Network). *Deep Learning* dapat diimplementasikan pada beberapa bidang, contohnya pada kamera pengawas lalu lintas yang mendeteksi plat nomor kendaraan yang kemudian data tersebut dicocokkan untuk membuat surat tilang kendaraan elektronik (Naufal Ihsan Dhuha dkk., 2021).

2.2.2 Visi Komputer (*Computer Vision*)

Menurut (Khan dkk., 2018), Visi komputer (*Computer Vision*) adalah bidang penelitian yang mempelajari cara komputer memproses gambar dan memahami apa yang ada di dalamnya. Ini mencakup penggunaan jaringan saraf komputer untuk melakukan berbagai tugas seperti mengklasifikasikan objek, segmentasi, mendeteksi objek di dalam gambar, dan pemahaman *scene*. Sebagian besar arsitektur *Convolutional Neural Network* (CNN) sering digunakan dalam visi komputer untuk masalah-masalah seperti mengidentifikasi wajah atau objek dalam gambar, menandai objek dalam gambar dengan kotak pembatas di sekitar setiap objek, segmentasi (misalnya, memberi label pada *pixel* gambar masukan), dan generasi gambar (misalnya, mengubah gambar beresolusi rendah menjadi yang beresolusi tinggi).

Visi komputer (*Computer Vision*) bertujuan untuk memberikan kemampuan kepada komputer yang setidaknya sebanding dengan kemampuan visual manusia, bahkan mungkin lebih baik. Lebih tepatnya, Visi Komputer (*Computer Vision*) berusaha untuk mengembangkan metode yang dapat meniru kemampuan menakjubkan dari sistem visual manusia, yakni mengenali karakteristik dunia nyata 3D secara langsung dengan menggunakan cahaya yang terpantul dari berbagai

objek. Meskipun demikian, memahami struktur 3D dari dunia nyata berdasarkan gambar 2D yang diambil oleh kamera merupakan tantangan tersendiri.

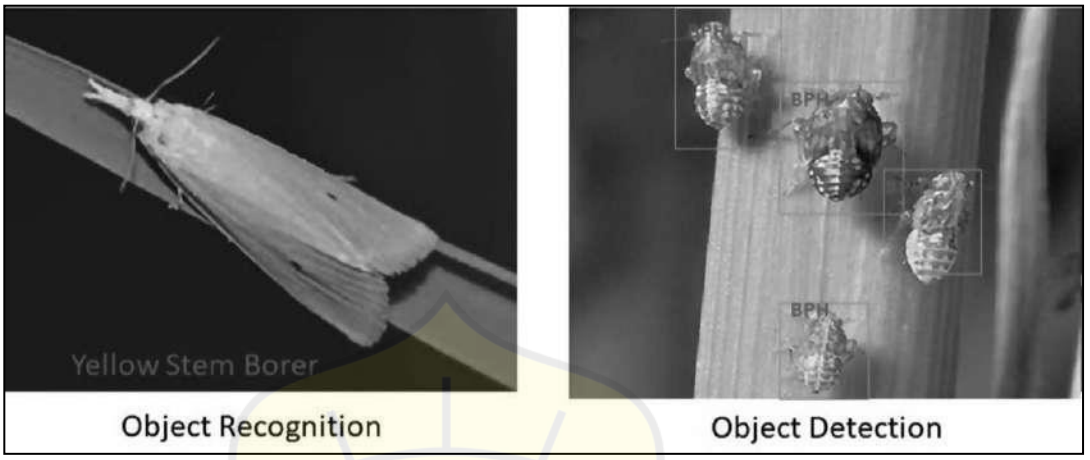
Secara garis besar, *Computer Vision* terkait dan memiliki aplikasi di bidang lain seperti kecerdasan buatan (AI), robotika, otomasi industri, pemrosesan sinyal, optik fisik, dan neurobiologi (Dompeipen dkk., 2021). Visi computer (*Computer Vision*) berfokus pada persepsi visual, memahami dunia melalui gambar dan video dengan membangun model fisik (Elgendy, 2020).

2.2.3 Object Detection

Menurut (Poonkuntran dkk., 2022), Deteksi Objek (*Object Detection*) adalah tugas utama dalam bidang visi komputer (*Computer Vision*) yang menangani identifikasi lokasi objek di dalam sebuah gambar. Sebagian besar tugas visi komputer, termasuk pelacakan objek otomatis, segmentasi instan, penjelasan gambar, dan deteksi anomali dan lain sebagainya, sangat bergantung pada deteksi objek. Sementara itu, pengenalan objek fokus pada identifikasi objek yang ada dalam sebuah gambar, sementara deteksi objek juga bertujuan untuk menemukan lokasi objek tersebut. Perbedaan antara kedua konsep tersebut dapat diilustrasikan dengan contoh di dalam Gambar 2.2, dimana pengenalan objek mengidentifikasi hama penggerek batang kuning dalam gambar, sedangkan deteksi objek tidak hanya mengidentifikasi hama tetapi juga menemukan koordinat lokasi hama tersebut.

Penelitian dalam deteksi objek dapat dibagi menjadi dua kategori utama, yaitu "deteksi objek umum" yang bertujuan untuk mengembangkan teknik yang dapat mendeteksi contoh objek dari kategori tertentu, serta "aplikasi yang

menggunakan deteksi objek" yang menerapkan metode deteksi objek dalam konteks tertentu seperti deteksi wajah atau pejalan kaki.



Gambar 2.2 *Object recognition vs. object detection* (Poonkuntran dkk., 2022).

2.2.4 Convolutional Neural Network (CNN)

Menurut (Khan dkk., 2018), *Convolutional Neural Network* (CNN) adalah salah satu jenis jaringan saraf yang paling populer, terutama untuk data yang memiliki dimensi tinggi seperti gambar dan video. Meskipun cara kerja CNN mirip dengan jaringan saraf biasa, perbedaan utamanya terletak pada setiap unit dalam lapisan CNN yang berfungsi sebagai *filter* 2D atau berdimensi tinggi yang mengalami konvolusi dengan input dari lapisan tersebut. Hal ini sangat penting terutama ketika kita ingin menganalisis pola dari data berdimensi tinggi seperti gambar atau video. *Filter* dalam CNN menggabungkan konteks spasial dengan memiliki bentuk spasial yang serupa dengan media input, namun ukurannya lebih kecil, dan menggunakan pembagian parameter untuk mengurangi jumlah variabel yang perlu dipelajari.

Menurut (Ketkar & Moolayil, 2021), *Convolutional Neural Network* (CNN) pada dasarnya adalah jenis jaringan saraf yang menggunakan operasi konvolusi sebagai bagian penting dalam salah satu lapisannya, berbeda dengan penggunaan *fully connected layer*. Teknologi CNN telah mencapai kesuksesan luar biasa dan telah diterapkan pada berbagai masalah di mana data masukan memiliki struktur grid yang sudah diketahui, seperti deret waktu (*grid 1-D*) atau gambar (*grid 2-D*). Kemunculan CNN telah membawa pembelajaran mendalam ke era modern, mengatasi salah satu tantangan komputasi paling signifikan dalam visi komputer di era digital. Dengan popularitasnya, CNN telah mendorong peningkatan dalam penelitian tentang pembelajaran mendalam yang masih terus berlanjut hingga saat ini.

CNN merupakan kelas model yang berguna untuk *supervised* dan *unsupervised learning*. Dalam *supervised learning*, model mempelajari pemetaan antara input ke sistem dan *output* yang diinginkan yang sudah diketahui, sedangkan dalam *unsupervised learning*, model berusaha untuk memperkirakan distribusi mendasar dari sampel data input tanpa memerlukan label yang sebenarnya. CNN belajar untuk memetakan gambar yang diberikan ke kategori yang sesuai dengan mendeteksi sejumlah representasi fitur abstrak, mulai dari yang sederhana hingga kompleks, dan fitur-fitur tersebut digunakan untuk memprediksi kategori yang tepat dari gambar *input* dalam jaringan.

CNN telah terbukti berhasil dalam bidang pengenalan dan klasifikasi gambar. Sebagai algoritma *Deep Learning,* CNN mengolah masukan berupa video/gambar, menetapkan bobot dan bias untuk berbagai aspek dalam gambar tersebut, kemudian membedakannya satu sama lain. Tujuannya adalah untuk

memanfaatkan informasi spasial di antara *pixel-pixel* dalam suatu gambar. Oleh karena itu, CNN didasarkan pada konvolusi diskrit. Bagian ini memberikan gambaran singkat tentang beberapa penelitian CNN yang sudah ada dan aplikasinya dalam deteksi dan pengenalan objek (Dhillon & Verma, 2020).

Lapisan konvolusi menghitung output nya dengan mengkonvolusi *input*, menambahkan bias β, dan melewatkan setiap hasil melalui fungsi aktivasi a[•]. Dengan ukuran kernel tiga, langkah satu, dan tingkat dilasi satu, unit tersembunyi ke-i^{-th} h^i akan dihitung sebagai:

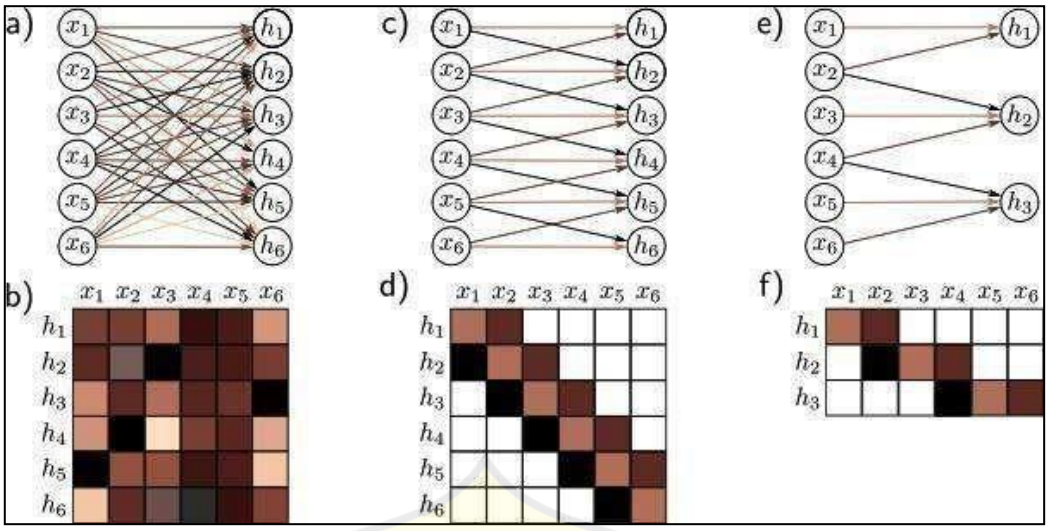
$$h^i = a[\beta + w_1 x_{i-1} + w_2 x_i + w_3 x_{i+1}]$$

$$= a\left[\beta_i + \sum_{j=1}^{D} w_{ij} x_j\right] \quad (1)$$

Di mana bias β dan bobot kernel $\omega_1, \omega_2, \omega_3$ adalah parameter yang dapat dilatih, dan (dengan *padding* nol) memperlakukan *input* x sebagai nol ketika berada di luar rentang yang valid. Ini adalah kasus khusus dari lapisan sepenuhnya terhubung yang menghitung unit tersembunyi ke-i^{-th} sebagai:

$$h^i = a\left[\beta_i + \sum_{j=1}^{D} w_{ij} x_j\right] \quad (2)$$

Jika ada D input $x_\bullet$ dan D unit tersembunyi $h_\bullet$, lapisan *fully connected* akan memiliki D^2 bobot $\omega_{\bullet\bullet}$ dan D bias $\beta_\bullet$. Sementara itu, lapisan konvolusi hanya membutuhkan tiga bobot dan satu bias. Lapisan *fully connected* dapat meniru fungsi ini dengan presisi jika sebagian besar bobotnya diatur ke nol dan sisanya diatur agar identik. Hal ini ditunjukkan pada gambar 2.3 di bawah ini. (Prince, 2023)



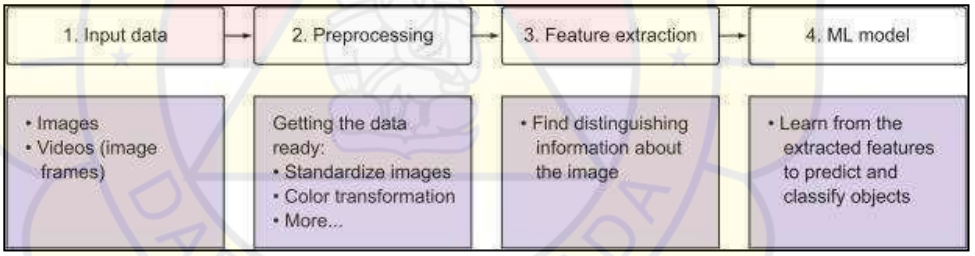
Gambar 2.0.3 Fully Connected vs. Convolutional Layers (Prince, 2023).

Pada gambar a) dapat dijelaskan bahwa sebuah lapisan *fully connected* memiliki bobot yang menghubungkan setiap *input* x dengan setiap unittersembunyi h yang dilambangkan dengan panah berwarna dan bias untuk setiap unit tersembunyi (tidak ditampilkan). Pada gambar b) dapat dijelaskan bahwa matriks bobot terkait Ω berisi 36 bobot yang menghubungkan enam input dengan enam unit tersembunyi. Pada gambar c) dapat dijelaskan bahwa lapisan konvolusi dengan ukuran kernel tiga menghitung setiap unit tersembunyi sebagai jumlah tertimbang yang sama dari tiga input tetangga (dilambangkan dengan panah) ditambah bias (tidak ditampilkan). Pada gambar d) dapat dijelaskan bahwa matriksbobot ini adalah contoh khusus dari matriks *fully connected* di mana banyak bobot bernilai nol dan lainnya berulang (warna yang sama menunjukkan nilai yang sama, warna putih menunjukkan bobot nol). Pada gambar e) dapat dijelaskan bahwa lapisan konvolusi dengan ukuran kernel tiga dan langkah dua menghitung jumlah tertimbang pada setiap posisi lainnya. Pada gambar f) dapat dijelaskan bahwa

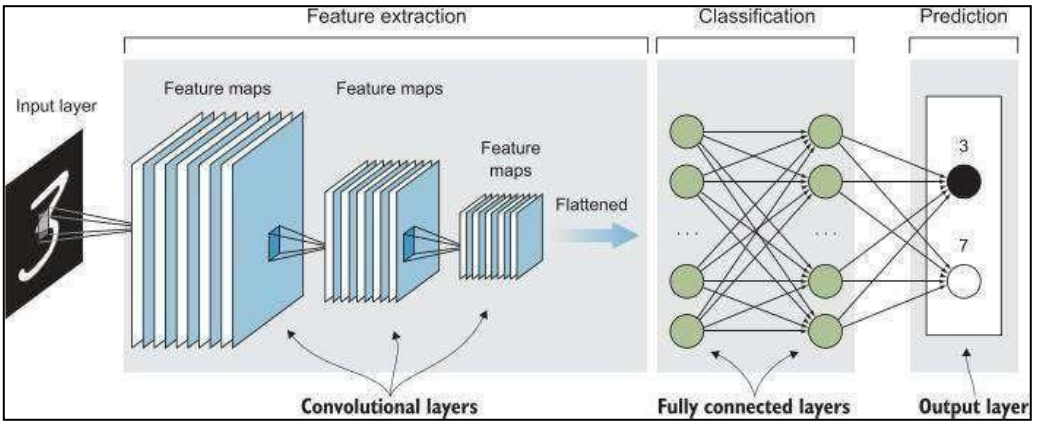
gambar tersebut juga merupakan contoh khusus dari jaringan *fully connected* dengan struktur bobot jarang yang berbeda (Prince, 2023).

2.2.4.1 Arsitektur *Convolutional Neural Network* (CNN)

Menurut (Elgendy, 2020), Pada umumnya, fitur-fitur dari CNN akan diekstrak secara manual dari gambar, kemudian vektor dari fitur yang dihasilkan akan diberikan kepada *classifier* (algoritma *Machine Learning* seperti SVM). Dengan kemampuan yang diberikan oleh *neural network*, *feature extraction* dapat digantikan dengan jaringan saraf (*Multi Layer Perceptron* (MLP) atau CNN) untuk melakukan pembelajaran fitur dan klasifikasi, seperti yang terdapat pada *feature extraction* dan ML model yang tertera pada Gambar 2.4.



Gambar 2.4 *Pipeline image classification* (Elgendy, 2020)



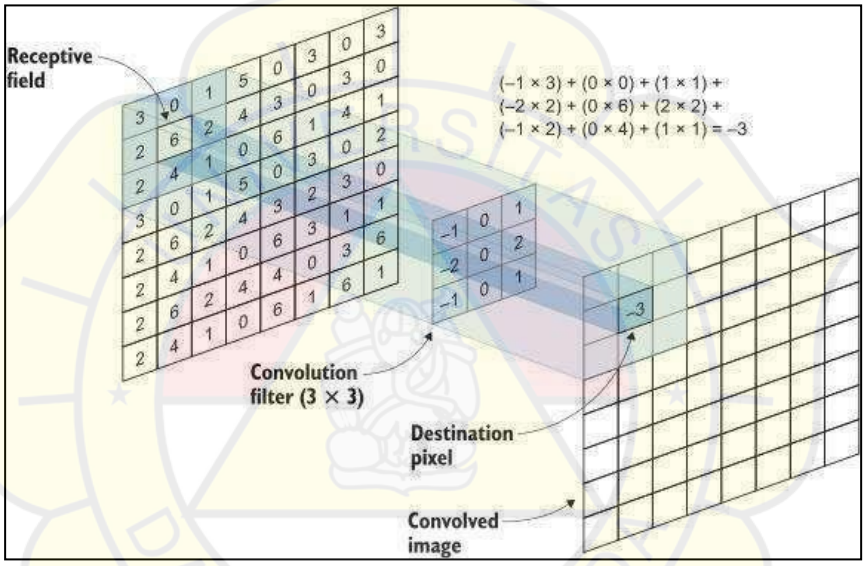
Gambar 2.5 Arsitektur CNN (Elgendy, 2020).

Gambar 2.5 diatas merupakan arsitektur dari *Convolutional Neural Network* (CNN). Beberapa tahapan utama dalam proses pengolahan data pada *Convolutional Neural Network* adalah *Input Layer* yang bertindak sebagai titik awal di mana data gambar dimasukkan ke dalam jaringan, *Convolutional Layers* yang berperan dalam mengekstraksi fitur-fitur penting dari gambar melalui proses konvolusi, *Fully Connected Layers* yang berperan untuk menyatukan fitur-fitur yang diekstraksi ke dalam representasi yang lebih abstrak, dan *Output Prediction Layer* yang berperan untuk memproduksi hasil berupa prediksi kelas atau nilai yang diinginkan berdasarkan input yang diberikan.

2.2.4.2 Convolutional Layer

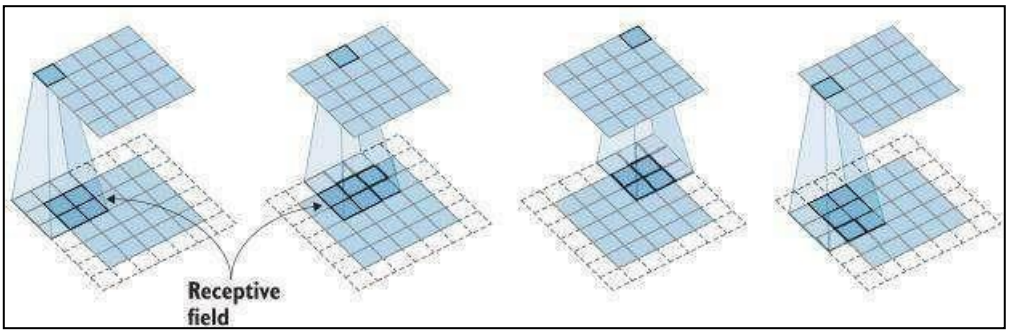
Menurut (Elgendy, 2020), *Convolutional Layer* merupakan komponen inti dari *convolutional neural network*. Fungsinya mirip dengan jendela pencari fitur yang bergerak di atas gambar, *pixel* per *pixel*, untuk mengekstraksi fitur-fitur yang penting dalam mengidentifikasi objek pada gambar. Dalam matematika, konvolusi adalah operasi yang melibatkan dua fungsi untuk menghasilkan fungsi yang telah

dimodifikasi. Dalam CNN, fungsi pertama adalah *input image*, sementara fungsi kedua adalah *convolutional filte*r yang akan memproses gambar untuk menghasilkan nilai *pixel* baru. Dalam Gambar 2.5, kita dapat melihat bagaimana *convolutional layer* pertama memproses gambar. Dengan menggunakan *filter* konvolusi pada *input image*, jaringan akan memecah gambar menjadi bagian-bagian kecil dan memproses setiap bagian secara terpisah untuk menyusun gambaryang dimodifikasi, yang disebut *feature map.*



Gambar 2.6 Filter konvolusi berukuran 3 × 3 (Elgendy, 2020).

Matriks kecil berukuran 3 × 3 yang berada di tengah adalah *filter* konvolusi, atau juga disebut sebagai kernel. Kernel tersebut bergeser di atas gambar asli *pixel* per *pixel* dan melakukan perhitungan matematika untuk mendapatkan nilai gambar "di konvolusi" yang baru di layer berikutnya. Area gambar yang diolah oleh *filter* dapat dilihat pada Gambar 2.6 (Elgendy, 2020).



Gambar 2.7 Proses Konvolusi dan Receptive Field dalam CNN (Elgendy, 2020)

Pada Gambar 2.7, kernel bergerak di seluruh gambar. Setiap elemen *pixel* yang cocok dikalikan dan hasilnya dijumlah untuk membentuk gambar baru dengan nilai *pixel* yang baru. Gambar hasil konvolusi ini dikenal sebagai *feature map* atau *activation map* (Elgendy, 2020).

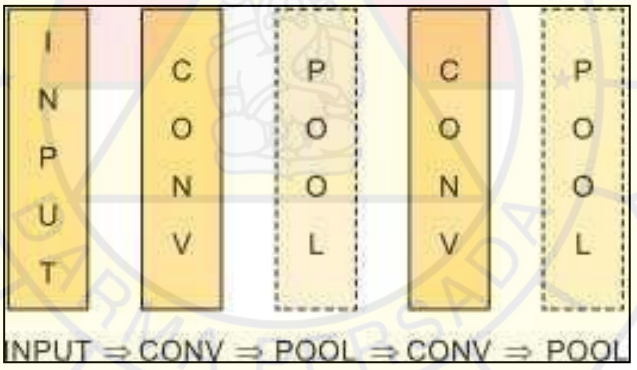
Gambar 2.8 Proses Konvolusi dalam CNN. (Elgendy, 2020)

2.2.4.3 Pooling Layer atau *Subsampling*

Menambahkan lebih banyak *convolutional layer* akan meningkatkan kedalaman *output layer*, sehingga meningkatkan jumlah parameter yang harus disesuaikan oleh jaringan. Saat menambahkan beberapa *convolutional layer*, akan ada peningkatan yang signifikan dalam jumlah parameter. Hal ini akan membuat

operasi matematika yang terlibat dalam pembelajaran menjadi lebih kompleks, baik dari segi waktu maupun ruang. Oleh karena itu, *pooling layer* berperan penting dalam membantu mengurangi ukuran jaringan dengan mengurangi jumlah parameter yang diteruskan ke *layer* berikutnya. *Pooling* merubah ukuran *input* dengan menggunakan fungsi statistik, seperti maksimum atau rata-rata, untuk mengurangi jumlah parameter secara keseluruhan yang diteruskan ke *layer* selanjutnya.

Tujuan dari *pooling layer* adalah untuk menurunkan ukuran *feature map* yang dihasilkan oleh *convolutional layer* sehingga mengurangi kompleksitas komputasi. Biasanya, *pooling layer* ditambahkan setelah satu atau dua *convolutional layer* dalam arsitektur CNN seperti pada Gambar 2.8 (Elgendy, 2020).



Gambar 2.9 Penambahan *Pooling Layer* dalam Arsitektur CNN (Elgendy, 2020).

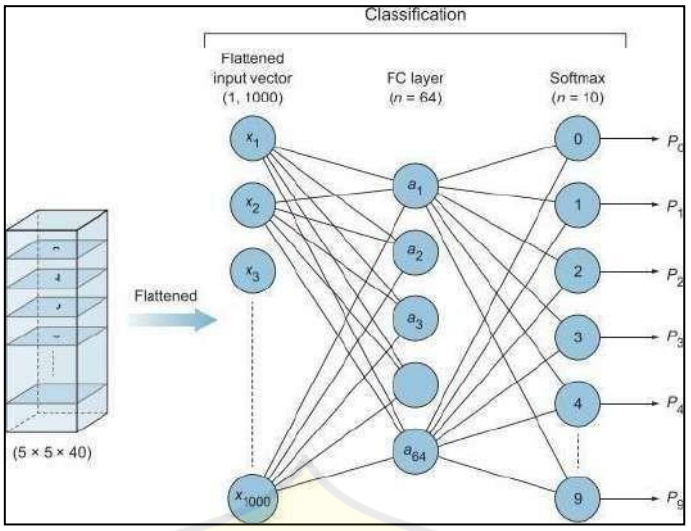
Terdapat dua jenis utama *pooling layer*, yaitu *max pooling* dan *average pooling*. Seperti kernel konvolusi, kernel *max pooling* adalah jendela dengan ukuran tertentu dan nilai langkah yang bergeser di atas gambar. Perbedaan dengan *max pooling* adalah bahwa jendela-jendela tersebut tidak memiliki bobot atau nilaiapa pun. Yang dilakukannya hanyalah bergeser di atas *feature map* yang dibuat oleh *convolutional layer* sebelumnya dan memilih nilai *pixel* maksimum untuk

dilewatkan ke *layer* berikutnya, tanpa memperhatikan nilai-nilai yang tersisa. Pada Gambar 2.10 terlihat sebuah *filter pooling* dengan ukuran 2 × 2 dan langkah sebesar 2 (*filter* melompat 2 *pixel* saat bergeser di atas gambar). *Pooling layer* akan mengurangi ukuran *feature map* dari 4 × 4 menjadi 2 × 2 (Elgendy, 2020).

Gambar 2.10 Pooling 2x2 Mengurangi *Feature Map* (Elgendy, 2020).

2.2.4.4 Fully Connected Layer

Setelah melewati proses pembelajaran fitur menggunakan *convolutional layer* dan *pooling*, semua fitur telah diekstrak dan ditempatkan dalam tabung panjang. Saatnya untuk menggunakan fitur-fitur yang diekstrak ini untuk mengklasifikasikan gambar. *Multi Layer Perceptron* (MLP) bekerja sangat baik dalam masalah klasifikasi. Alasan penggunaan *convolutional layer* adalah MLP kehilangan banyak informasi berharga saat mengekstrak fitur dari gambar-gambar harus diratakan sebelum diberikan kepada jaringan, sedangkan *convolutional layer* dapat memproses gambar secara langsung. Setelah setelah fitur-fitur tersebut diekstrak, dan setelah diratakan, MLP dapat digunakan untuk mengklasifikasikannya. Untuk menegaskan, dapat dilihat pada Gambar 2.11 yang merupakan *fully connected layer*.



Gambar 2.11 *Fully connected layers* untuk *Multi Layer Perceptron (MLP)* (Elgendy, 2020).

2.2.5 Fungsi Aktivasi dan Metrik

2.2.5.1 Sigmoid

Sigmoid adalah fungsi aktivasi yang digunakan dalam jaringan saraf tiruan, termasuk *Convolutional Neural Network* (CNN). Fungsi ini mengubah nilai *input* menjadi nilai antara 0 dan 1, membuatnya sangat cocok untuk tugas klasifikasi biner. Dengan perhitungan rumus sebagai berikut:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Dimana:

a. $\sigma(x)$ adalah *output* dari fungsi *sigmoid*.

b. x adalah *input* ke fungsi *sigmoid*.

c. e adalah bilangan *Euler*, yang kira-kira sama dengan 2.71828.

2.2.5.2 Binary Cross Entropy (BCE)

Menurut (Dr.A, 2020), *Binary Cross Entropy (*BCE) adalah fungsi *loss* yang umum digunakan dalam tugas klasifikasi biner, termasuk klasifikasi gambar dengan *Convolutional Neural Networks* (CNN). BCE mengukur perbedaan antaradistribusi probabilitas yang diprediksi dan distribusi aktual. Fungsi ini sangat cocok untuk melatih model dalam menyelesaikan banyak masalah klasifikasi sekaligus, jika setiap klasifikasi dapat direduksi menjadi pilihan biner. Dengan perhitungan rumus sebagai berikut:

$$L = \frac{1}{N}\sum_{i=1}^{N} [y_i \cdot \log(\hat{y}_l) + (1 - \hat{y}_l) > \log(1 - \hat{y}_l)] \quad (4)$$

Dimana:

a. *N* adalah jumlah sampel

b. $\hat{y}_i$ adalah probabilitas prediksi untuk sampai ke-*i* berada di kelas 1.

c. log adalah logaritma natural.

2.2.5.3 Akurasi

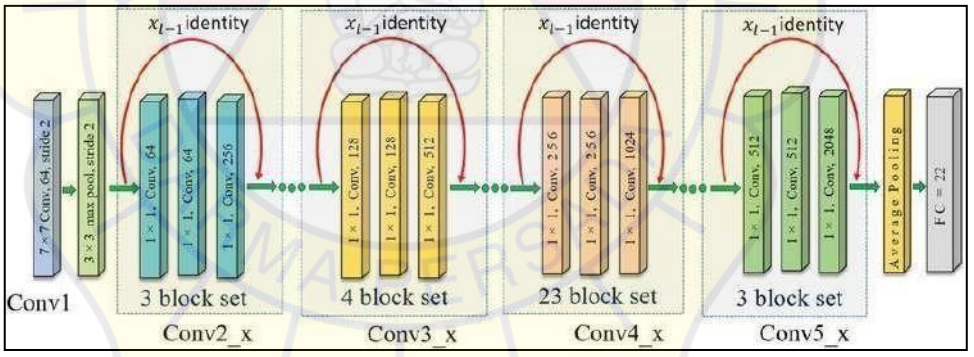
Akurasi adalah ukuran kinerja yang sering digunakan dalam evaluasi model klasifikasi. Akurasi menunjukkan seberapa sering model klasifikasi membuat prediksi yang benar dibandingkan dengan total prediksi yang dilakukan. Secara matematis, akurasi dihitung sebagai rasio jumlah prediksi benar terhadap jumlah total sampel yang diuji. Secara umum, rumus untuk menghitung akurasi adalah sebagai berikut:

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Jumlah\ Total\ Sampel} \quad (5)$$

2.2.6 Arsitektur Model

2.2.6.1 Residual Network (ResNet)

Residual Network (ResNet) adalah tipe jaringan saraf tiruan yang dirancang berdasarkan struktur sel piramidal yang terdapat di korteks serebral. (Harahap dkk., 2022) *ResNet* adalah model yang telah dilatih sebelumnya sehingga tidak memerlukan konfigurasi untuk pengaturan *layer*nya. Prinsip kerja *ResNet* adalah membangun jaringan yang lebih dalam dibandingkan jaringan konvensional, sambil menemukan jumlah lapisan optimal untuk mengatasi masalah gradien yang menghilang (Ridhovan & Suharso, 2022).



Gambar 2.12 Arsitektur Model ResNet 101 (Tiwari dkk., 2022)

Pada Gambar 2.12 adalah arsitektur model *ResNet 101* dengan penjelasan sebagai berikut:

1. Conv1
    a. 7x7 Conv, 64, *stride* 2

Lapisan konvolusi dengan kernel 7x7, 64 *filter*, dan *stride* 2.

b. 3x3 *max pool*, *stride* 2

Lapisan *pooling* maksimum dengan kernel 3x3 dan *stride* 2.

2. Conv2_x

A. 3 *block* set terdiri dari tiga blok residual. Setiap *block* terdiri dari:

a. Konvolusi 1x1 yang mengurangi dimensi (*bottleneck*).

b. Konvolusi 3x3.

c. Konvolusi 1x1 yang mengembalikan dimensi ke ukuran semula.

B. *Shortcut connection* adalah koneksi langsung dari *input* ke *output* setiap *block* untuk memastikan gradien dapat mengalir tanpa hambatan.

3. Conv3_x

A. 4 *block* set terdiri dari empat *block* residual dengan struktur yang sama seperti pada Conv2_x, tetapi dengan peningkatan jumlah filter sebagai berikut:

a. Konvolusi 1x1 yang mengurangi dimensi.

b. Konvolusi 3x3.

c. Konvolusi 1x1 yang mengembalikan dimensi ke ukuran semula.

B. Shortcut Connection.

4. Conv4_x

    A. 23 *block* set terdiri dari dua puluh tiga *block* residual dengan struktur yang sama seperti pada Conv2_x dan Conv3_x, tetapi dengan peningkatan jumlah filter yang lebih besar, dengan penjelasan sebagai berikut:

        a. Konvolusi 1x1 yang mengurangi dimensi.

        b. Konvolusi 3x3.

        c. Konvolusi 1x1 yang mengembalikan dimensi ke ukuran semula.

    B. Shortcut Connection.

5. Conv5_x

    A. 3 *block* set terdiri dari tiga *block* residual dengan struktur yang sama seperti pada Conv2_x dan Conv3_x, tetapi dengan jumlah filter yang lebih besar lagi, sebagai berikut:

        a. Konvolusi 1x1 yang mengurangi dimensi.

        b. Konvolusi 3x3.

        c. Konvolusi 1x1 yang mengembalikan dimensi ke ukuran semula.

6. Average Pooling

Lapisan *pooling* rata-rata mengurangi dimensi spasial (tinggi dan lebar) dari *feature map*.

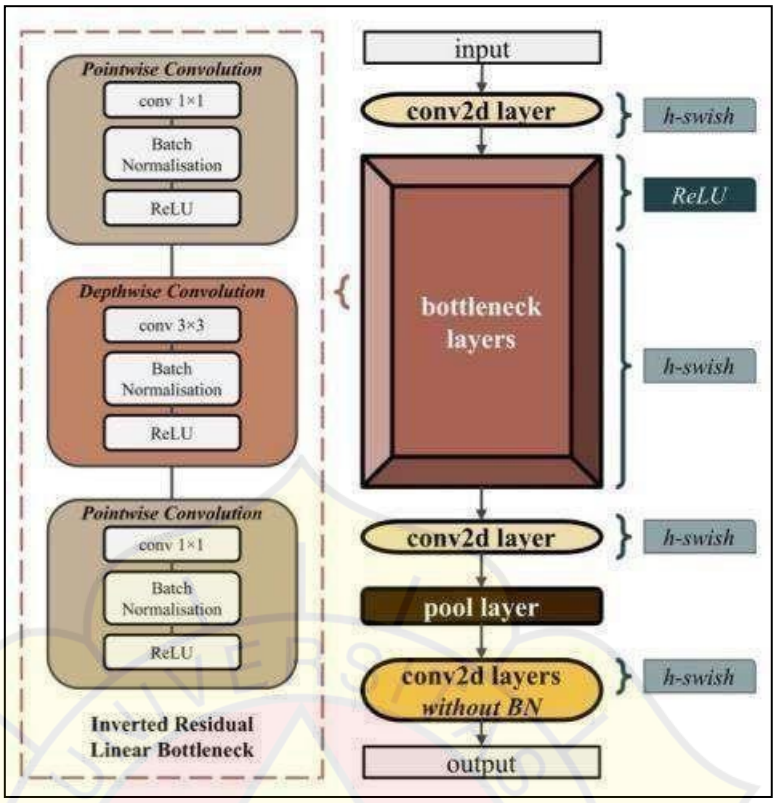
7. *FC (Fully Connected) layer*

   Lapisan terhubung penuh dengan jumlah *neuron* yang sesuai dengan jumlah kelas (C = 22 dalam gambar ini).

Arsitektur *ResNet 101* ini memungkinkan pelatihan jaringan yang sangat dalam dengan total 101 lapisan, karena *shortcut connections* memudahkan *backpropagation* dengan mengurangi masalah gradien yang memudar. Hasilnya adalah jaringan yang lebih dalam dan lebih kuat untuk tugas-tugas klasifikasi gambar dan visi komputer lainnya.

2.2.6.2 MobileNet

MobileNet adalah model dengan latensi rendah dan konsumsi daya minimal, yang dirancang untuk mengatasi keterbatasan sumber daya dalam berbagai skenario penggunaan (Fahcruroji dkk., 2024). *MobileNet* menggunakan *depthwise separable convolutions*, yang membagi *convolutions* standar menjadi *depthwise convolution* (satu filter per saluran *input*) dan *pointwise convolution* (*convolutions* 1×1 untuk penggabungan hasil). Ini memisahkan proses penyaringan dan penggabungan yang biasanya dilakukan dalam satu langkah pada *convolutions* standar (Howard dkk., 2017).



Gambar 2.13 Arsitektur Model MobileNet V3 (Qian, 2022).

Pada Gambar 2.13 diatas adalah arsitektur model *MobileNet V3* dengan penjelasan sebagai berikut:

1. *Input* dan Lapisan Awal

   a. Input

   Data gambar masuk ke jaringan melalui lapisan *input.*

   b. Conv2d *layer*

   Lapisan konvolusi 2D awal dengan aktivasi *h-swish*, yang diikuti oleh *batch normalization.*

2. *Bottleneck Layers*

   Bagian utama dari arsitektur terdiri dari beberapa *bottleneck layers* yang menggunakan *block Inverted Residual Linear Bottleneck. Block* ini terdiri dari:

   a. *Pointwise Convolution* (conv 1x1)

      Konvolusi 1x1 untuk mengubah dimensi saluran *(channel).* Ini diikuti oleh *batch normalization* dan aktivasi *ReLU*.

   b. *Depthwise Convolution* (conv 3x3)

      Konvolusi 3x3 yang diterapkan secara terpisah pada setiap saluran, diikuti oleh *batch normalization* dan aktivasi *ReLU*.

   c. *Pointwise Convolution* (conv 1x1)

      Konvolusi 1x1 kedua untuk mengembalikan dimensi saluran ke ukuran semula, diikuti oleh *batch normalization* dan aktivasi *ReLU.*

3. Lapisan Setelah *Bottleneck*

   Setelah bottleneck layers, terdapat lapisan-lapisan, sebagai berikut:

   a. *Convd layer*

      Lapisan konvolusi 2D dengan aktivasi *h-swish* dan *batch normalization.*

   b. *Pool layer*

      Lapisan pooling untuk mengurangi dimensi spasial dari *feature map,* yang diikuti oleh aktivasi *h-swish.*

4. Lapisan Akhir

   Convd layers without BN adalah serangkaian lapisan konvolusi 2D tanpa batch *normalization,* yang diikuti oleh aktivasi *h-swish.*

5. *Output*

   Lapisan output menghasilkan hasil akhir dari jaringan yang dapat digunakan untuk klasifikasi atau tugas lainnya.

   Aktivasi Normalisasi

   a. *ReLU* dan *h-swish*

      Dua fungsi aktivasi yang digunakan dalam jaringan ini. *h-swish* adalah varian dari fungsi *swish* yang lebih efisien dan sering digunakan dalam *MobileNet V3.*

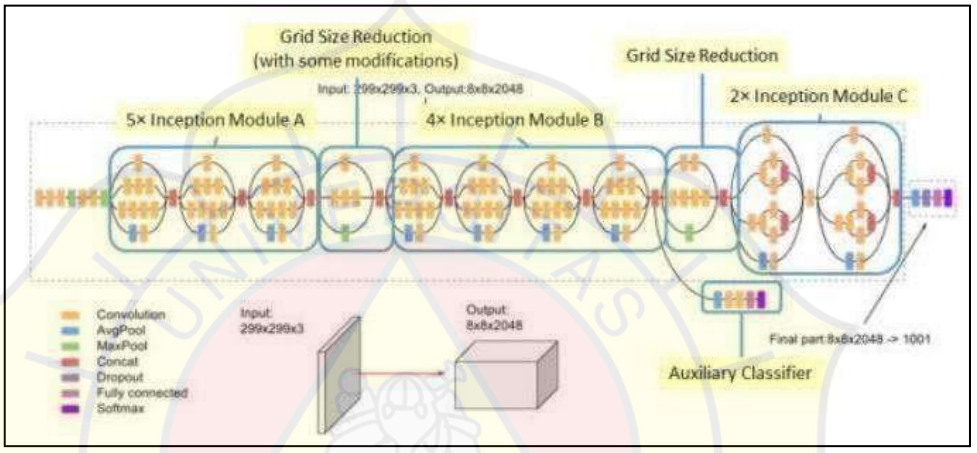
   b. *Batch Normalization*

      Normalisasi *batch* digunakan setelah setiap konvolusi untuk menstabilkan dan mempercepat pelatihan jaringan.

Arsitektur ini menggabungkan efisiensi dari *depthwise separable convolutions* dengan *block inverted residual linear bottleneck* yang dioptimalkan. Penggunaan *h-swish* sebagai fungsi aktivasi, yang lebih efisien dibandingkan *ReLU* dalam konteks *MobileNet V3,* juga membantu dalam meningkatkan kinerja keseluruhan jaringan. Pendekatan ini memungkinkan *MobileNet V3* menjadi sangat efisien dalam hal ukuran model dan kecepatan inferensi, sambil tetap mempertahankan akurasi yang tinggi.

2.2.6.3 Inception

Inception V3 adalah jenis jaringan saraf konvolusi yang dikembangkan oleh tim peneliti Google pada tahun 2015. Model ini merupakan pengembangan dari versi sebelumnya, yaitu *Inception V1* dan *Inception V2*, dengan tujuan untuk meningkatkan kecepatan komputasi dan kinerja dalam mengenali gambar (Kosman dkk., 2024).



Gambar 2.14 Arsitektur Model Inception V3 (Pratiwi & Pardede, 2022).

Pada Gambar 2.14 diatas adalah arsitektur model *Inception V3* dengan penjelasan sebagai berikut:

1. *Input* dan *Grid Size Reduction*
   a. *Input*

      Data gambar dengan ukuran 299 x 299 x 3 (tinggi x lebar x kanal warna) masuk ke jaringan.

   b. *Grid Size Reduction (with some modifications)*

      Bagian ini mengurangi ukuran *grid* (dimensi spasial) dari *input*, kemungkinan melalui serangkaian lapisan konvolusi dan *pooling*.

Hasilnya adalah *output* dengan ukuran *grid* yang lebih kecil, siap untuk masuk ke modul *Inception.*

2. *Inception Modules*

Arsitektur *Inception* terdiri dari beberapa jenis modul *Inception*, yang merupakan *block* utama dari jaringan ini. Modul-modul ini menjalankan berbagai operasi konvolusi dan *pooling* secara paralel, kemudian menggabungkan hasilnya:

A. 5x *Inception Module A*

a. Terdiri dari lima modul *Inception A*, yang mungkin mencakup berbagai operasi konvolusi dan *pooling* untuk menangkap fitur pada berbagai skala.
b. *Output* dari modul ini tetap diproses lebih lanjut dengan mengurangi ukuran *grid*.

B. 4x *Inception Module B*

Diikuti oleh empat modul *Inception B*, yang melanjutkan pemrosesan dengan operasi yang lebih kompleks untuk menangkap fitur yang lebih mendalam.

3. *Grid Size Reduction*

Setelah modul *Inception B*, ada lagi pengurangan ukuran *grid* untuk memperkecil dimensi spasial dari *feature map* lebih lanjut sebelum masuk ke modul berikutnya.

4. *Inception Module C*

   A. 2x *Inception Module C*

      Dua modul *Inception C* digunakan untuk pemrosesan akhir fitur sebelum masuk ke klasifikasi.

5. *Auxiliary Classifier*

   Auxiliary Classifier (klasifikator tambahan) digunakan sebagai mekanisme regularisasi untuk membantu pelatihan. Ini adalah sub-jaringan kecil yang terhubung di tengah arsitektur untuk memberikan tambahan gradien ke lapisan awal jaringan. Ini membantu jaringanutama untuk konvergensi lebih cepat dan mengurangi *overfitting*.

6. *Output*

   A. *Final part*

      a. Setelah semua modul *Inception, feature map* akhir dikompres dan diproses untuk menghasilkan *output* akhir dengan dimensi 8x8x2048.

      b. AvgPool

         Lapisan *pooling* rata-rata digunakan untuk mengurangi dimensi spasial lebih lanjut.

      c. Dropout

         Teknik regularisasi *dropout* diterapkan untuk mencegah *overfitting.*

d. Fully Connected

Lapisan *fully connected* digunakan untuk klasifikasi akhir.

e. Softmax

Fungsi aktivasi *softmax* menghasilkan probabilitas untuk setiap kelas *output*, menghasilkan prediksi akhir.

Arsitektur *Inception (Inception V3)* dirancang untuk menangkap berbagai tingkat fitur dari gambar melalui paralelisasi operasi konvolusi dan *pooling* dalam modul *Inception*. Dengan menggunakan berbagai ukuran filter dan *pooling* dalam setiap modul, jaringan dapat menangkap fitur pada skala yang berbeda secara efisien. Selain itu, penggunaan *auxiliary classifier* membantu meningkatkan stabilitas dan kinerja selama pelatihan. Arsitektur ini sangat efisien dalam hal komputasi dan sangat efektif dalam tugas-tugas klasifikasi gambar.

2.2.7 Website

Menurut (Abdulloh, 2018), *website* adalah kumpulan halaman yang memuat berbagai informasi digital seperti teks, gambar, animasi, suara, dan video yang dapat diakses oleh siapapun melalui internet. Halaman *website* dibangun dengan menggunakan bahasa standar yaitu HTML, yang kemudian diterjemahkan oleh *web browser* agar dapat ditampilkan kepada pengguna dalam format yang dapat dibaca.

Menurut (Vossen dkk., 2017), Sejak awal tahun 1990-an, *World Wide Web* atau yang sering disebut sebagai *website* telah mengalami perubahan besar dalam kehidupan pribadi dan profesional. Dalam 25 tahun terakhir, dapat dilihat bagaimana *web* berkembang menjadi sumber informasi utama yang diakses melalui

mesin pencari (*search engine)* dan portal, serta sebagai *platform* untuk berbagai media yang memfasilitasi penyimpanan dan berbagi konten gratis.

2.3 Software dan *Tools*

2.3.1 Google Drive

Menurut (Agus dkk., 2019), *Google Drive* adalah sebuah layanan penyimpanan data *online* yang dimulai dari *Google Docs* dan sekarang dimiliki oleh Google Inc. Layanan ini diluncurkan pada 24 April 2012 dan menyediakan pengguna dengan 15 *gigabyte* (GB) ruang penyimpanan gratis. *Google Drive* merupakan salah satu solusi penyimpanan *cloud* terbaik. Seperti layanan penyimpanan cloud lainnya, *Google Drive* memiliki aplikasi untuk iOS dan Android yang memungkinkan pengguna untuk mengakses dan mengelola file dari perangkat seluler. Fitur yang mencolok dari *Google Drive* adalah kemampuannya untuk mengedit dokumen, spreadsheet, dan presentasi secara langsung. *Google Drive* juga terintegrasi dengan *Google* Foto, yang memungkinkan pengguna untuk mengatur foto dalam album secara *online*. Pengguna dapat mengakses *Google Foto*melalui aplikasi *Google Drive* tanpa perlu mengunduh aplikasi *Google* Foto terpisah. Fitur menarik lainnya adalah kemampuan untuk mengunggah *file* secara otomatis dengan menyeret dan meletakkannya ke situs *web Google Drive*. Pengguna juga dapat melihat pratinjau lampiran *email* dari *Gmail* langsung di *Google Drive* dan menyimpannya ke *cloud*. Meskipun menggunakan *Google Drive* memerlukan sedikit persiapan jika pengguna sudah memiliki akun Google, aplikasiini sangat mudah digunakan, terutama untuk pekerja kantor atau siapa pun yang

membutuhkan fasilitas *office* dengan penyimpanan *cloud* (Safitri & Nasution, 2023).

2.3.2 Google Colab

Menurut (Dolinay, 2023), G*oogle Colab* adalah *platform* yang memungkinkan untuk menulis, menjalankan, dan berbagi kode *Python* langsung dari *browser*. Ini merupakan versi dari *Jupyter Notebook* yang terintegrasi dalam rangkaian alat Google. *Jupyter Notebook* dan *Google Colab* memungkinkan untuk membuat dokumen yang berisi kode yang dapat dieksekusi, serta teks, gambar, HTML, dan lainnya. Dokumen yang dibuat di *Google Colab* juga dapat disimpan di *Google Drive* dan dibagikan untuk pengeditan, komentar, maupun tampilan.

2.3.3 Tensorflow

Menurut (Zaccone & Karim, 2018), *TensorFlow* merupakan sebuah *software* matematika dan *framework open source* untuk *deep learning* yang dikembangkan oleh Tim Google Brain pada tahun 2011. Tujuan utama *TensorFlow* adalah untuk menganalisis data guna memprediksi hasil bisnis yang efektif. Meskipun awalnya dikembangkan untuk melakukan penelitian dalam bidang *Machine Learning* dan *Deep Neural Networks* (DNNs), *TensorFlow* memiliki fleksibilitas yang memungkinkannya digunakan untuk berbagai jenis algoritma klasik *machine learning* seperti *Support Vector Machine* (SVM), *logisticregression*, *decision trees*, dan *random forest*.

TensorFlow adalah sebuah *framework open source* yang dikembangkan oleh Google, digunakan untuk melakukan komputasi ilmiah dan numerik dengan menggunakan grafik aliran data sebagai model eksekusinya. Grafik aliran data yang dipakai dalam *TensorFlow* membantu para ahli dalam bidang *Machine Learning* untuk melakukan pelatihan yang lebih lanjut dan intensif pada data mereka, yang bertujuan untuk mengembangkan model analitik *Deep Learning* dan prediktif.

Sesuai dengan namanya, *TensorFlow* mencakup berbagai operasi yang dilakukan oleh *neural networks* pada data *multidimensional*, atau *tensor*. Node-node dalam grafik aliran tersebut merepresentasikan operasi-operasi matematika seperti penjumlahan, perkalian, faktorisasi matriks, dan lain-lain, sementara itu, tepian-tepian menghubungkan tensor-tensor tersebut untuk memastikan terjadi komunikasi antara node dan tepian, yang mengatur aliran data dan kontrol. Dengan demikian, *TensorFlow* menyediakan berbagai model linear dan algoritma *Deep Learning* yang populer dan diimplementasikan dengan kokoh.

2.3.4 Python

Menurut (Guttag, 2021), *Python* adalah bahasa pemrograman umum yang efektif digunakan untuk membuat hampir semua jenis program yang tidak memerlukan akses langsung ke perangkat keras komputer. Sejak diperkenalkan oleh Guido von Rossum pada tahun 1990, bahasa ini telah mengalami banyak perubahan. Selama dekade pertama eksistensinya, *Python* adalah bahasa yangkurang dikenal dan kurang digunakan. Namun, situasi tersebut berubah seiring kedatangan *Python 2.0* pada tahun 2000. Selain memasukkan perbaikan penting

pada bahasa itu sendiri, ini menandai perubahan jalur evolusi bahasa tersebut. *Python 3.0* dirilis pada akhir tahun 2008. Versi *Python* ini membersihkan banyak inkonsistensi dalam desain *Python 2*. Namun, *Python 3* tidak kompatibel ke belakang. Artinya, sebagian besar program dan pustaka yang ditulis untuk versi *Python* sebelumnya tidak dapat dijalankan menggunakan implementasi *Python 3*.

Meskipun demikian, *Python* memiliki beberapa keunggulan dibandingkan dengan banyak bahasa pemrograman lainnya. Bahasa ini relatif mudah dipahami dan dipelajari. Karena Python dirancang untuk diinterpretasikan, ia dapat memberikan umpan balik secara langsung saat runtime yang sangat bermanfaat bagi pemula dalam pemrograman. Ada juga banyak pustaka yang tersedia secara gratis yang berinteraksi dengan *Python* dan menambahkan fungsionalitas tambahanyang bermanfaat.

2.3.5 Streamlit

Menurut (Khorasani dkk., 2022), *Streamlit* adalah sebuah *server side-web framework* yang sangat disarankan untuk tetap mengisolasi aspek-aspek sistem lainnya demi modularitas dan keamanan. *Streamlit* merupakan *framework* pertama yang sepenuhnya menggunakan *Python* yang dapat mempersingkat proses *development website*.

2.3.6 SQLite

Menurut (Kreibich, 2010), *SQLite* adalah *public-domain software package* yang menyediakan *relational database management system* (RDBMS). RDBMS digunakan untuk menyimpan catatan yang didefinisikan pengguna dalam tabel-tabel besar. Selain menyimpan dan mengelola data, mesin basis data juga dapat memproses perintah *query* yang kompleks dengan menggabungkan data dari berbagai tabel untuk membuat laporan dan ringkasan data. Istilah "*Lite*" dalam *SQLite* tidak mengacu pada kapabilitasnya, melainkan menggambarkan kesederhanaannya dalam hal *setup*, *overhead* administratif, dan penggunaan sumber daya.

2.4 Kajian Penelitian Terdahulu

Tabel 2.1 Tinjauan Kajian Penelitian Terdahulu

NO	PENULIS	TAHUN	METODE	JDUL	HASIL
1	Richo, Ryan Yudha Adhitya, Muhammad Khoirul Hasin, Mat Syai'in, Edy Setiawan	2023	*Convolutional Neural Network (CNN)*	Analisis Pengaruh *Optimizer* pada Model CNN untuk Identifikasi Cacat pada	Penelitian ini menunjukkan bahwa penambahan *optimizer* memiliki dampak signifikan terhadap hasil *training*.

				Perekat Kemasan	Berdasarkan proses *training* yang telah dilakukan, ditemukan bahwa model yang menggunakan *optimizer* tambahan menghasilkan training *accuracy* yang lebih optimal daripada model tanpa *optimizer*, dengan perbandingan masing-masing sebesar 95,57% dan 67,53%. Berdasarkan evaluasi seluruh parameter yang diujikan selama proses *training*,

					diketahui bahwa *optimizer* Adam menunjukkan kinerja yang lebih baik dibandingkan dengan *optimizer* lainnya, dengan *validation accuracy* mencapai 92,77%. Selain itu, hasil prediksi yang benar terhadap data latih sebanyak 98 data, dan parameter *confusion* mencapai 99%. Pada tahap testing, diperoleh bahwa keakuratan deteksi yang

					dicapai dengan menggunakan *Optimizer* Adam adalah yang tertinggi, di mana dari 30 data yang diuji, 28 di antaranya berhasil dideteksi dengan benar, mencapai tingkat keberhasilan pengujian sebesar 90,00%. Dapat disimpulkan bahwa *Optimizer* Adam merupakan pilihan terbaik untuk mengklasifikasikan hasil deteksi kemasan tepung pada model

					CNN.
2	Alfendio Alif Faudisyah, Kristoko Dwi Hartomo, Hindriyanto Dwi Purnomo	2023	*Convolutional Neural Network* (CNN)	Deteksi Cacat pada Isolasi Trafo Secara Visual menggunakan Algoritma *Convolutional Neural Network* (CNN)	Penelitian menggunakan algoritma *Convolutional Neural Network* (CNN) menunjukkan bahwa perbedaan dalam standarisasi data dapat mempengaruhi kinerja model. Penelitian ini menguji dua jenis standarisasi data, yaitu dengan menggunakan gambar berukuran 180 x 180 x 3 *pixel* dan 240 x 240 x 3 *pixel*. Dataset yang

					digunakan telah mengalami augmentasi dan terdiri dari 432 gambar, dengan pembagian 80% untuk *training* dan 20% untuk *testing*. Hasil pengujian dengan standarisasi data ukuran 180 x 180 x 3 *pixel* menunjukkan akurasi 0,9913 untuk *training*, 0,9884 untuk *testing*, dan 1,00 untuk evaluasi. Sementara itu, hasil pengujian dengan standarisasi data

					ukuran 240 x 240 x 3 *pixel* mencatat akurasi 0,9798 untuk *training*, 0,9651 untuk *testing*, dan 0,94 untuk evaluasi.
3	Krisna Prayoga, Rita Magdalena, Sofia Saidah	2023	*Convolutional Neural Network* (CNN)	Sistem Deteksi Kecacatan Ban Dengan *Convolutional Neural Network*	Penelitian menunjukkan hasil pengujian sistem dengan lima *layer* menghasilkan *F-1 Score* tertinggi sebesar 0,88 atau 88%. Hasil pengujian terbaik diperoleh dengan menggunakan parameter *pixel* 224x224, *Optimizer* adam,

					Learning Rate 0,0001, *Epoch* 80, dan *Batch size* 16. Keberhasilan sistem dalam mendeteksi kecacatan pada ban mencapai 88%, dan sebanyak 12% system mendeteksi salah.
4	Alan Antoni, Tatang Rohana,	2023	*Convolutional Neural Network* (CNN)	Implementasi Algoritma *Convolutional Neural Network* Untuk Klasifikasi Citra Kemasan Kardus *Defect* dan *No Defect*	Penelitian menunjukkan bahwa algoritma *Convolutional Neural Network* (CNN) efektif dalam mengklasifikasi citra kemasan kardus menjadi kategori *defect*

					dan *no defect*, dengan akurasi tertinggi mencapai 95,77% menggunakan *hyperparameter input size* 300x300, *epoch* 30, dan *learning rate* 0,001.
5	Yozika Arvio, Dine Tiara Kusuma, Iriansyah Sangadji, Erno Kurniawan Dewantara	2023	*Convolutional Neural Network* (CNN)	Penerapan Metode *Convolutional Neural Network* (CNN) Dalam Proses Pengolahan Citra Untuk Mendeteksi Cacat Produksi Pada	Penelitian menunjukkan implementasi arsitektur CNN mencapai akurasi yang tinggi, mencapai 99% dalam pengujian dengan data yang telah ada dan 96,4% dalam pengujian *real-*

				Produk Masker	*time* dengan 28 data. Hal ini Menunjukkan kemampuan arsitektur CNN dalam mengklasifikasi data secara efisien dan akurat.
6	Inggis Kurnia Trisiawan, Yuliza, Fina Supegina, Said Attamimi	2022	*Convolutional Neural Network* (CNN)	Penerapan *Multi-Label Image Classification* Menggunakan Metode *Convolutional Neural Network* (CNN) Untuk Sortir Botol Minuman	Penelitian ini menunjukkan bahwa model arsitektur *Convolutional Neural Network* (CNN) berhasil mengklasifikasikan gambar ke dalam enam kategori yang telah ditentukan, dengan akurasi pelatihan

					mencapai 95,02% dan *loss* sebesar 0,1064. Rata-rata akurasi prediksi pada pengujian menggunakan metode *10-fold cross-validation* adalah 97,71%, sementara pada pengujian dengan dataset baru mencapai 98,526%.
7	Rindi Kusumawardani, Putu Dana Karningsih	2021	*Convolutional Neural Network* (CNN)	Deteksi dan Klasifikasi Cacat Kemasan Kaleng Menggunakan *Convolutional*	Penelitian menunjukkan bahwa algoritma CNN efektif dalam membedakan jenis kerusakan

				Neural Network	pada kemasan kaleng. Pengujian menggunakan dataset citra yang telah disiapkan maupun pengujian pada tiap citra menunjukkan hasil akurasi pengujian di atas 90% untuk kelima model jaringan yang diuji yaitu *ShuffleNet, GoogleNet, ResNet18, ResNet50,* dan *ResNet101*. Ini menandakan bahwa