BAB 2
LANDASAN TEORI

2.1 Udang Vanamei

Udang vanamei (Litopenaeus vannamei) berasal dari daerah subtropis pantai barat Amerika, mulai dari teluk California di Mexico bagian utara sampai ke Pantai barat Guatemala, El Savador, Nicaragua, Kosta Rika di Amerika Tengah hingga Peru di Amerika Selatan. Udang vanamei resmi diizinkan masuk ke Indonesia melalui SK Menteri Kelautan dan Perikanan RI No.41/2001 [13].

Udang vaname memiliki sifat nocturnal yaitu aktif pada malam hari. Udang ini sering ditemukan memendamkan diri di dasar lumpur/pasir dasar kolam dan tidak mencari makan pada siang hari. Akan tetapi, jika siang hari tetap diberi pakan maka udang vaname akan tetap bergerak untuk mencari makan. Hal ini menujukkan bahwa sifat nocturnal pada udang vaname ini tidak mutlak. Udang vaname juga memiliki sifat kanibalisme, pemakan lambat dan terus-menerus[14].

Klasifikasi: Kerajaan Animalia, Filum Arthropoda, Kelas Malacostraca, Ordo Decapoda, Famili Penaeidae, Genus Litopenaeus, Spesies Litopenaeus vannamei. Habitat: Udang vaname hidup di perairan dangkal pesisir, laguna, dan muara dengan dasar berpasir atau berlumpur. Mereka lebih menyukai suhu air yang hangat (25-30°C) dan kadar garam yang relatif rendah (15-30 ppt) [15]

Gambar 2.1 Udang Vannamei [9]

2.2 WSSV (*White Spot Syndrome Virus*)

White Spot Syndrome Virus (WSSV) adalah patogen yang menyebabkan penyakit bintik putih (White Spot Disease) pada udang. Biasanya, Petambak lebih familiar dengan sebutan penyakit WSSV. WSSV pertama kali terdeteksi di Taiwan pada tahun 1992, kemudian menyebar ke wilayah Asia Tenggara hingga ke wilayah Amerika Latin. Awal penyebaran penyakit WSSV di Indonesia pada tahun 1994 di pesisir utara Jawa Timur, Jawa Tengah, Jawa barat, hingga Aceh Selatan[16]. Virus ini saat ini terdaftar oleh Komite Internasional Taksonomi Virus (ICTV) sebagai satu-satunya anggota genus Whispovirus dalam keluarga Nimaviridae. Beberapa isolat virus telah diurutkan genomnya, dan perbedaannya telah ditinjau lebih lanjut[17].

Gejala perilaku dan fisiologis akibat infeksi bervariasi dalam tingkat keparahan antar inang yang rentan, dan hewan yang terinfeksi secara laten masih dapat bertahan hidup tanpa menunjukkan gejala penyakit yang terlihat. Hewan yang sakit umumnya menunjukkan perilaku lesu, perubahan warna pada hepatopankreas, penurunan nafsu makan, dan kurangnya perawatan tubuh, yang mengakibatkan kelonggaran pada kutikula[17]. Pada beberapa spesies WSSV berbentuk bulat telur dan memiliki selubung virus dengan panjang 210-420 nm dan lebar 70-167 nm. WSSV dapat dengan mudah menyebar ke bagian tubuh udang melalui hemolimfa yang membentuk garis-garis horizontal dengan lebar 20 nm. Udang yang terserang WSSV akan mempunyai bintik putih dengan diameter 0,5-2 mm pada bagian kepalanya. Ketika infeksi makin parah, bintik putih ini bisa menyebar ke seluruh tubuh udang. Selain bintik putih, tubuh udang yang terinfeksi WSSV akan berubah menjadi pucat kemerahan[16].

Penyakit WSSV terjadi akibat adanya stres pada udang akibat faktor lingkungan,seperti Kualitas air yang buruk, termasuk kadar oksigen yang rendah, kadar amonia dan nitrit yang tinggi, hingga fluktuasi pH, dapat menjadi penyebab stresnya udang vaname saat budidaya berlangsung,penanganan dan pemindahan udang yang kasar dan tidak sesuai dengan standar juga dapat menyebabkan stres pada udang. Hal ini karena udang sangat sensitif terhadap stimulan dari luar, seperti goncangan, paparan suhu yang terlalu tinggi, dan kepadatan yang berlebihan, Udang sensitif terhadap perubahan lingkungan yang mendadak. Misalnya termasuk perubahan cahaya, suhu, dan salinitas dalam waktu yang cepat. Perubahan mendadak ini dapat menjadi salah satu faktor penyebab stres pada udang vaname, pakan dan nutrisi yang tidak sesuai, Kualitas pakan yang buruk atau tidak memadai dapat menyebabkan stres pada udang. Sebab, pakan dengan kualitas buruk dapat

menyebabkan udang mengalami ketidakseimbangan nutrisi, vitamin, dan mineral yang didapat[18].

WSSV menimbulkan bintik-bintik putih pada karapas udang yang berasal dari penumpukan garam-garam kalsium abnormal. Tanda-tanda penyakit WSSV dapat meliputi:

1. Udang bergerak ke tepi kolam atau permukaan
2. Penurunan nafsu makan
3. Renang tidak teratur
4. Lesu
5. Tingkat variasi warna yang tinggi
6. Bercak berwarna putih pucat
7. Tingkat kesakitan atau kematian yang tinggi[19]

Perkembangan WSSV yang mencakup tiga tahap, sebagai berikut:

Inkubasi (Tahap Awal):

1. Udang yang terinfeksi WSSV akan menunjukkan atau tidak menunjukkan gejala klinis
2. Udang yang terinfeksi mungkin masih menunjukkan aktivitas renang yang normal

Transisi: Muncul bintik putih pada karapas dan muncul gejala klinis lainnya

Onset (Tahap Akhir): Udang yang terinfeksi akan mati dalam hitungan beberapa jam hingga hari[16]

Udang yang terinfeksi WSSV akan mengalami perubahan tingkah laku seperti menurunnya aktivitas berenang, berenang tidak terarah, dan sering kali berenang pada satu sisi tambak saja. Selain itu, udang akan cenderung bergerombol di tepi tambak dan berenang di permukaan.WSSV dapat menular melalui jalur vertikal, yaitu menyebar melalui induk ke benur udang. Penularan lainnya melalui jalur horizontal, yaitu melalui air dan terjadi kontak langsung dengan udang yang terinfeksi WSSV. Selain itu, WSSV dapat menular melalui perantara burung dari tambak satu ke tambak lain. Cara penularannya adalah burung memakan udang sakit yang berenang di permukaan tambak dan sisa udang yang tidak termakan oleh burung jatuh ke tambak lainnya [16].

Gambar 2.2 Udang Terkena Penyakit WSSV [9]

2.3 Deteksi Objek

Deteksi objek adalah tugas dalam bidang visi komputer yang bertujuan untuk menemukan letak objek dalam gambar digital. Oleh karena itu, deteksi objek merupakan salah satu bentuk dari kecerdasan buatan yang melatih komputer agar dapat "melihat" seperti halnya manusia, khususnya dalam mengenali dan mengklasifikasikan objek berdasarkan kategori semantik tertentu. Lokalisasi objek adalah teknik untuk menentukan lokasi spesifik suatu objek dalam gambar dengan cara menandai objek tersebut menggunakan bounding box (kotak pembatas). Sementara itu, klasifikasi objek adalah teknik yang menentukan ke dalam kategori apa objek yang terdeteksi tersebut termasuk. Tugas deteksi objek menggabungkan dua sub-tugas utama, yaitu lokalisasi objek dan klasifikasi objek, untuk secara bersamaan memperkirakan lokasi dan jenis objek dalam satu atau beberapa gambar [20].

Deteksi objek merupakan salah satu bidang paling penting dalam tugas visi komputer, yang juga merupakan cabang utama dari kecerdasan buatan. Tujuan dari deteksi objek adalah menemukan dan menentukan posisi objek secara akurat dalam gambar atau video yang diberikan. Dengan berkembangnya teknik pembelajaran mendalam (deep learning), algoritma yang lebih kuat dan andal telah muncul untuk menangani objek dalam berbagai skala dan fitur tingkat tinggi, sehingga dapat mengatasi keterbatasan dari alur kerja deteksi objek tradisional [21].

Deteksi objek terbagi menjadi dua kategori yaitu "*Two-stage Detection*" dan "*One-stage Detection*". Deteksi objek "*Two-stage Detection*" mempunyai dua stages (tahapan) dalam melakukan proses deteksi objek, tahap pertama yaitu memprediksi kandidat object proposal atau bounding box (kotak pembatas untuk mengidentifikasi posisi dan tipe objek) pada citra dan selanjutnya pada tahap kedua dilakukan proses klasifikasi dan regresi terhadap kandidat bounding box yang nantinya akan menghasilkan bounding box yang menunjukkan letak

serta tipe objek sebaliknya deteksi objek "*One-stage Detection*" hanya membutuhkan satu stage (tahapan) untuk melakukan proses deteksi objek [22].

2.4 Preprocessing

Preprocessing adalah proses pengolahan data awal yang digunakan untuk mengubah data mentah yang diperoleh dari berbagai sumber menjadi informasi yang lebih bersih dan dapat digunakan untuk analisis lebih lanjut [23]. Namun terdapat beberapa proses juga dalam preprocessing seperti membersihkan, mengintegrasikan, mentransformasikan dan mereduksi data [24].

Melalui preprocessing, memungkinkan proses mining akan berjalan dengan lebih efektif dan efisien. Karena data yang telah melalui Pra-pemrosesan data, merupakan data yang sudah melalui beberapa tahap pembersihan [24]. Sehingga tujuan utama dari praproses data adalah untuk meningkatkan kualitas data. Tahapan ini membantu dalam menangani nilai yang hilang, menghapus data duplikat, serta melakukan normalisasi data. Dengan demikian, praproses data memastikan akurasi dan konsistensi dari himpunan data yang digunakan [25].

2.4.1 Resize

Gambar digital merupakan representasi visual yang tersusun atas elemen-elemen terkecil yang disebut piksel, yaitu titik-titik berwarna yang berpadu membentuk citra secara keseluruhan. Setiap gambar memiliki jumlah piksel yang tetap saat pertama kali dibuat. Oleh karena itu, proses perubahan ukuran gambar (*resizing*) pada dasarnya merupakan penambahan atau pengurangan jumlah piksel, yang secara langsung memengaruhi dimensi, resolusi, serta ukuran berkas gambar. Tindakan ini umumnya dilakukan dengan berbagai tujuan, seperti mengurangi ukuran file untuk menghemat ruang penyimpanan, mempermudah proses pengiriman atau unggahan, maupun menyesuaikan dimensi dan resolusi gambar agar sesuai dengan kebutuhan tertentu, misalnya untuk keperluan pencetakan atau publikasi pada media daring[26].

Resize image (juga disebut penskalaan atau penyampelan ulang) adalah proses mengubah dimensi gambar digital dengan meningkatkan (upscaling) atau mengurangi (downscaling) ukurannya. Ini merupakan operasi mendasar dalam pemrosesan citra dan visi komputer. Dalam proses perubahan ukuran gambar, dimensi gambar diubah untuk memenuhi persyaratan spesifik dalam tugas pemrosesan citra dan visi komputer. Perubahan ukuran gambar melibatkan perubahan jumlah piksel dalam gambar, yang memengaruhi baik ukuran visualnya maupun jumlah data yang diperlukan untuk merepresentasikannya[27].

Ketika mengubah ukuran gambar, secara efektif mengubah jumlah piksel yang digunakan untuk merepresentasikan gambar tersebut. Baik saat memperbesar (upsampling) maupun memperkecil (downsampling) gambar, proses ini dapat memengaruhi kualitas keseluruhan. Upsampling menambahkan piksel ke gambar, yang sering kali menghasilkan gambar yang lebih buram karena perangkat lunak harus memperkirakan warna piksel baru berdasarkan piksel di sekitarnya. Di sisi lain, downsampling menghapus piksel, yang dapat menyebabkan hilangnya detail, terutama pada area gambar yang rumit[28].

Pra-pemrosesan ini dilakukan menggunakan platform Roboflow, yang meliputi beberapa tahap:

a. *Auto Orient*: memastikan orientasi gambar tetap konsisten untuk menghindari kesalahan dalam pemrosesan lebih lanjut. Beberapa gambar memiliki metadata rotasi yang tersembunyi akibat cara pengambilan gambar dengan perangkat yang berbeda, seperti kamera atau ponsel [29]
b. *Static Crop*: memotong area 37–75% dari region horizontal dan vertikal guna mempertahankan bagian gambar yang paling informatif [30]
c. *Resize*: mengubah ukuran gambar menjadi 800×800 piksel agar seragam dan sesuai dengan kebutuhan model [30]

Rumus teori menghitung resize gambar

$$scale = min\left(\frac{Wo}{Wt}, \frac{Ho}{Ht}\right) \quad (1)$$

Keterangan:

Wo, Ho = lebar dari tinggi gambar asli

Wt, Ht = lebar dan tinggi gambar target

2.4.2 Augmentasi Data

Augmentasi Data menggambarkan sekumpulan algoritma yang membangun data sintetis dari kumpulan data yang tersedia. Data sintetis ini biasanya mengandung perubahan kecil pada data yang seharusnya tidak mempengaruhi prediksi model. Selain itu, data sintetis juga dapat merepresentasikan kombinasi antara contoh yang jauh berbeda, yang sebaliknya akan sangat sulit untuk disimpulkan. Augmentasi Data adalah salah satu metode paling berguna untuk mempengaruhi pelatihan Jaringan Saraf Dalam (Deep Neural Networks). Hal ini terutama disebabkan oleh sifat transformasi yang dapat diinterpretasikan serta kemampuannya untuk memberikan wawasan tentang bagaimana model mengalami kegagalan [31].

Augmentasi data digambarkan sebagai strategi untuk mencegah overfitting melalui regularisasi. Dengan memperoleh data yang benar atau akurat, yang paling berharga untuk penelitian dan eksperimen, merupakan tantangan yang sulit, meskipun informasi tersedia secara luas. Data harus cukup beragam dalam berbagai ukuran, posisi, warna, dan kondisi pencahayaan agar model dapat bekerja dengan lebih baik. Banyak prosedur augmentasi data diterapkan untuk mengatasi masalah keterbatasan jumlah data. Metode-metode ini membantu mengekstrak informasi dari data yang sudah ada [32].

Inti dari augmentasi data adalah meningkatkan kecukupan dan keberagaman data pelatihan dengan menghasilkan kumpulan data sintetis. Data yang telah diperluas ini dapat dianggap berasal dari distribusi yang mendekati distribusi data asli. Dengan demikian, kumpulan data yang telah diperluas dapat mewakili karakteristik yang lebih komprehensif. Namun, masih ada beberapa tantangan penelitian dalam metode augmentasi data untuk citra. Pertama, teknik augmentasi data citra dapat diterapkan dalam berbagai tugas visi komputer, seperti deteksi objek, segmentasi semantic, dan klasifikasi gambar. Namun, tantangannya adalah bahwa metode augmentasi data bersifat tidak spesifik terhadap tugas tertentu. Hal ini terjadi karena operasi augmentasi dilakukan secara bersamaan pada data citra dan label, sedangkan jenis label berbeda untuk setiap tugas. Sebagai contoh, metode augmentasi yang digunakan untuk tugas deteksi objek tidak dapat langsung diterapkan pada tugas segmentasi semantik. Akibatnya, efisiensi berkurang dan skalabilitas menjadi rendah [33].

Berikut merupakan beberapa jenis metode augmentasi data yang diimplementasikan dalam deteksi dan klasifikasi penyakit udang vaname:

1. Rotasi: merupakan salah satu teknik augmentasi data yang umum digunakan. Teknik ini melibatkan rotasi suatu gambar secara acak, baik searah maupun berlawanan arah jarum jam, dengan sudut tertentu. Rotasi ini mengubah posisi objek dalam citra sehingga meningkatkan variasi data pelatihan [34].
2. *Flipping*: Membalik gambar secara horizontal atau vertikal untuk meningkatkan model terhadap orientasi objek. Teknik ini bertujuan untuk meningkatkan kemampuan model dalam mengenali objek dengan berbagai orientasi tanpa mengubah karakteristik aslinya [35].
3. Crop: Teknik augmentasi data yang dilakukan dengan mengambil bagian acak dari suatu gambar asli untuk membentuk citra baru. Teknik ini membantu model dalam melakukan generalisasi dengan lebih baik, karena objek yang ingin dikenali tidak selalu terlihat secara utuh atau dalam skala yang sama pada seluruh data pelatihan[36]

4. Brightness Adjustment: teknik yang digunakan untuk menyesuaikan tingkat kecerahan citra dalam proses pelatihan model. Teknik ini memungkinkan perubahan terhadap intensitas cahaya dalam gambar, sehingga model dapat lebih tahan terhadap variasi pencahayaan di dunia nyata [37].
5. Exposure: teknik yang digunakan untuk menyesuaikan tingkat eksposur suatu citra, yaitu seberapa terang atau gelap gambar akibat pengaturan cahaya. Teknik ini membantu model deep learning beradaptasi dengan berbagai kondisi pencahayaan dalam data pelatihan [37].
6. Noise: Noise adalah gangguan pada citra yang terjadi akibat perubahan nilai piksel dari representasi aslinya. Noise dapat muncul secara sengaja maupun tidak sengaja, mengubah tampilan gambar dengan menambahkan variasi acak pada piksel [38].

Rumus untuk menentukan jumlah gambar yang diaugmentasi menggunakan berbagai metode augmentasi data adalah sebagai berikut [39]

$$Jumlah\ augmentasi\ per\ teknik = \frac{Total\ gambar\ tambahan}{Jumlah\ teknik\ augmentasi} \quad \text{.....(2)}$$

2.4.3 Normalisasi

Normalisasi data adalah suatu proses transformasi data agar memenuhi kondisi tertentu atau mengikuti skala tertentu, sehingga memudahkan perbandingan atau analisis antar data yang memiliki karakteristik atau rentang yang berbeda. Tujuan normalisasi termasuk penyelarasan skala atribut, menghindari dominasi atribut dengan nilai besar terhadap atribut dengan nilai kecil, dan meningkatkan performa algoritma machine learning yang sensitif terhadap skala. Beberapa metode normalisasi data yang umum digunakan melibatkan transformasi nilai-nilai atribut agar sesuai dengan skala tertentu[40]Salah satu metode normalisasi yang sering digunakan dalam melakukan normalisasi data adalah Metode decimal scaling yang merupakan metode transformasi data dengan normalisasi untuk menyamakan rentang nilai pada setiap atribut dengan skala tertentu dengan menggerakkan nilai desimal dari data ke arah yang diinginkan. Agar dapat menghasilkan data yang lebih baik [41].

Proses normalisasi dilakukan dengan mengonversi nilai piksel setiap gambar ke dalam rentang tertentu. Biasanya, nilai piksel dikonversi ke dalam rentang 0 hingga 1 dengan membagi nilai piksel dengan 255. Jika model pretrained seperti ResNet digunakan dalam *Faster* R-CNN, normalisasi dilakukan dengan mengurangi nilai *mean* sebesar [0.485, 0.456, 0.406] dan membaginya dengan standar deviasi [0.229, 0.224, 0.225] dari *dataset* ImageNet.

Langkah ini membantu menyamakan distribusi nilai piksel dengan data pelatihan model sebelumnya, sehingga model dapat mengenali pola lebih baik [42]

Penjelasan dari teori diatas menyatakan bahwa terdapat rumus normalisasi untuk menghitung rentang 0 hingga 1:

$$X' = \frac{X}{255} \quad \text{.....(3)}$$

Selanjutnya terdapat rumus Normalisasi Z-Score dalam perhitungan normalisasi dari penjelasan diatas:

Rumus Normalisasi Z-Score:

$$X_{norm} = \frac{X - \mu}{\sigma} \quad \text{.....(4)}$$

2.4.4 *Bounding Box*

Bounding Box adalah sebuah kotak imajiner yang digunakan untuk mengelilingi objek yang telah terdeteksi. *Bounding Box* memiliki bentuk kotak dan ukurannya sama dengan ukuran objek yang terdeteksi [43]. Berdasarkan pernyataan diatas *bounding box* dan label lebih presisi terhadap object yang akan dideteksi sehingga pengujian terhadap *bounding box* dan label bertujuan untuk meningkatkan akurasi dan presisi deteksi objek. Pengujian ini mencakup evaluasi ukuran dan posisi *bounding box* yang mengelilingi objek dalam gambar, serta memverifikasi ketepatan label yang diberikan. Proses ini membantu memastikan bahwa setiap objek terdeteksi dengan benar dan label yang disematkan sesuai dengan kategori objek yang sebenarnya. Dengan melakukan penyesuaian dan perbaikan berdasarkan hasil pengujian, model deteksi dapat mencapai tingkat presisi yang lebih tinggi dan meminimalkan kesalahan deteksi atau pelabelan [44].

Terdapat dua skenario utama dalam penggunaan *bounding box* pada proyek pengolahan citra, yaitu:

1. Labelling, yaitu ketika menentukan dan menandai objek dalam gambar sebagai bagian dari proses pelatihan model.
2. Inferensi, yaitu ketika model yang telah dilatih digunakan untuk mendeteksi dan mengenali objek dalam gambar [45].

Pelabelan *bounding box* merupakan tahap penting dalam proyek deteksi objek. Pada tahap ini, *bounding box* digambar di sekitar objek yang menjadi fokus untuk menandai area yang akan dikenali oleh model. *Bounding box* ini kemudian digunakan sebagai data pelatihan bersama dengan gambar masukan untuk membangun model visi komputer. Untuk melakukan pelabelan *bounding box*, diperlukan alat pelabelan khusus yang mendukung proses penandaan objek dalam gambar. Alat-alat ini umumnya memiliki berbagai fitur yang

membantu mempercepat dan mempermudah proses pelabelan. Model visi komputer menghasilkan koordinat yang menunjukkan lokasi suatu objek yang telah dilatih untuk dikenali oleh model. Koordinat ini dapat disajikan dalam berbagai format, dengan dua format yang paling umum digunakan, yaitu:

1. **(x_0, y_0, x_1, y_1) atau xyxy** – Format ini mendefinisikan *bounding box* berdasarkan dua titik sudutnya, yaitu sudut kiri atas *(x_0, y_0)* dan sudut kanan bawah *(x_1, y_1)*.
2. **(x, y, width, height) atau xywh** – Format ini mendefinisikan *bounding box* berdasarkan titik pusat *(x, y)* serta lebar *(width)* dan tinggi *(height)* dari kotak tersebut. [45].
3. Cara kerja *bounding box* pada *Faster* R-CNN

$$A = (xmax - xmin) \times (ymax - ymin) \quad (5)$$

2.5 Deep Learning

Deep Learning adalah cabang dari machine learning yang didasarkan pada arsitektur jaringan saraf tiruan (artificial neural network atau ANN). Jaringan saraf tiruan ini menggunakan lapisan-lapisan node yang saling terhubung, yang disebut neuron, yang bekerja bersama untuk memproses dan belajar dari data input. Dalam jaringan saraf deep yang sepenuhnya terhubung, terdapat lapisan input dan satu atau lebih lapisan tersembunyi yang terhubung secara berurutan. Setiap neuron menerima input dari neuron pada lapisan sebelumnya atau dari lapisan input, dan output dari satu neuron menjadi input bagi neuron lain pada lapisan berikutnya. Proses ini berlangsung hingga lapisan akhir menghasilkan output dari jaringan tersebut. Transformasi data input melalui lapisan-lapisan jaringan saraf ini dilakukan dengan serangkaian transformasi non-linear yang memungkinkan jaringan untuk mempelajari representasi kompleks dari data tersebut [46]

Arsitektur dasar deep learning terinspirasi dari struktur otak manusia, yang bertujuan untuk menciptakan sistem yang mampu belajar seperti manusia. Oleh karena itu, beberapa terminologi dalam deep learning dapat ditelusuri kembali ke bidang neurologi. Mirip dengan bagaimana neuron membentuk blok bangunan utama otak, arsitektur deep learning memiliki unit komputasi yang memungkinkan pemodelan fungsi non-linear, yang disebut perseptron. Perseptron berfungsi seperti "neuron" dalam otak manusia, yang mengirimkan impuls listrik melalui sistem saraf kita; perseptron menerima serangkaian sinyal input dan mengubahnya menjadi sinyal output.

Perseptron memiliki tujuan untuk memahami representasi data dengan menumpuk banyak lapisan, di mana setiap lapisan bertanggung jawab untuk memahami bagian tertentu

dari input. Lapisan ini dapat dianggap sebagai kumpulan unit komputasi yang belajar mendeteksi kemunculan berulang dari nilai-nilai tertentu. Setiap lapisan perseptron bertugas untuk menafsirkan pola tertentu dalam data. Jaringan perseptron ini meniru cara neuron di otak membentuk jaringan, sehingga arsitektur ini disebut jaringan saraf atau jaringan saraf tiruan [47]

Gambar 2.3 Deep Learning [30]

Melalui prinsip dan arsitektur ini, deep learning memungkinkan pembelajaran otomatis dari representasi data yang sangat kompleks, yang pada akhirnya membuka jalan bagi mesin untuk melakukan tugas-tugas yang sebelumnya dianggap hanya dapat dilakukan oleh manusia, seperti pengenalan gambar, pemrosesan bahasa alami, dan pengambilan keputusan berbasis data yang rumit.

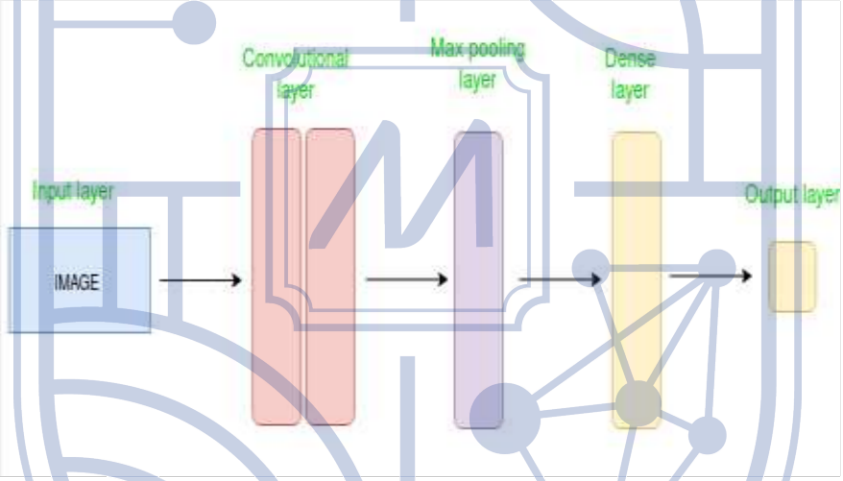
2.6 CNN

Convolutional Neural Network (CNN), atau yang juga dikenal sebagai ConvNet, adalah jenis algoritma pembelajaran mendalam yang khusus dirancang untuk tugas-tugas yang membutuhkan pengenalan objek, termasuk klasifikasi gambar, deteksi, dan segmentasi. CNN digunakan dalam berbagai skenario praktis, seperti kendaraan otonom, sistem kamera keamanan, dan lainnya [48].

CNN dilatih menggunakan *dataset* besar yang diberi label, di mana jaringan belajar mengenali pola dan fitur yang berkaitan dengan objek atau kelas tertentu. Terbukti sangat efektif dalam tugas-tugas terkait gambar, CNN mencapai performa state-of-the-art dalam

berbagai aplikasi computer vision. Kemampuan CNN untuk secara otomatis mempelajari representasi hierarkis fitur membuatnya sangat cocok untuk tugas-tugas di mana hubungan spasial dan pola dalam data sangat penting untuk prediksi yang akurat. CNN banyak digunakan dalam klasifikasi gambar, deteksi objek, pengenalan wajah, dan analisis citra medis[49].

Lapisan konvolusi adalah komponen utama dari CNN, di mana filter diterapkan pada gambar input untuk mengekstrak fitur seperti tepi, tekstur, dan bentuk. Output dari lapisan konvolusi kemudian diteruskan ke lapisan pooling, yang digunakan untuk mengurangi dimensi spasial feature map, mengurangi kompleksitas spasial namun tetap mempertahankan informasi yang paling penting. Output dari lapisan pooling kemudian diteruskan ke satu atau lebih lapisan fully connected, yang digunakan untuk membuat prediksi atau mengklasifikasikan gambar [50].



Gambar 2.4 Arsitektur CNN [32]

2.7 R-CNN

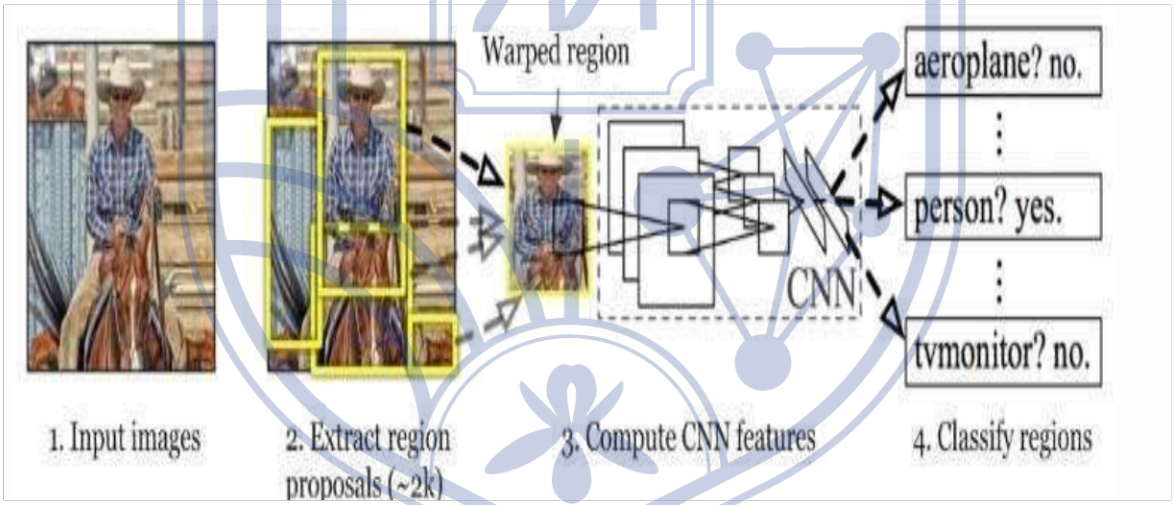
Region-based Convolutional Neural Network (R-CNN) adalah jenis arsitektur pembelajaran mendalam yang digunakan untuk deteksi objek dalam tugas-tugas visi komputer. R-CNN merupakan salah satu model perintis yang membantu memajukan bidang deteksi objek dengan menggabungkan kekuatan jaringan saraf convolutional dan pendekatan berbasis wilayah[51]. Untuk mengatasi tantangan dalam deteksi objek, Ross Girshick memperkenalkan R-CNN. Pendekatan ini memanfaatkan algoritma selective search untuk menghasilkan sekitar 2.000 proposal wilayah, yang kemudian diproses melalui *Convolutional Neural Network* (CNN) untuk mengekstrak fitur. Fitur-fitur ini kemudian

diklasifikasikan menggunakan *Support Vector Machine* (SVM), sementara *bounding box* regressor digunakan untuk meningkatkan akurasi lokalisasi[52].

R-CNN mengidentifikasi dan melokalisasi objek dalam gambar dengan mengusulkan *Regions of Interest* (RoI) dan mengklasifikasikannya melalui CNN. Kerangka kerja deteksi objek dimulai dengan gambar input yang mengandung potensi objek dan menggunakan *Region Proposal Network* (RPN), seperti Selective Search, untuk menghasilkan *bounding boxes* yang mungkin mengandung objek[52].

Setiap wilayah yang diusulkan diubah ukurannya dan dimasukkan ke dalam CNN yang sudah dilatih sebelumnya, seperti AlexNet atau VGG16, untuk mengekstrak representasi fitur. Fitur-fitur ini kemudian diklasifikasikan oleh SVM ke dalam kategori yang telah ditentukan atau ditandai sebagai latar belakang. Untuk memperbaiki lokalisasi lebih lanjut, model regresi *bounding box* menyesuaikan koordinat setiap kotak, menyelaraskannya lebih dekat dengan batas objek yang sebenarnya[52].

Proses sistematis ini secara efektif menggabungkan generasi proposal, ekstraksi fitur, klasifikasi, dan penyempurnaan *bounding box*, memungkinkan deteksi objek yang akurat [52].



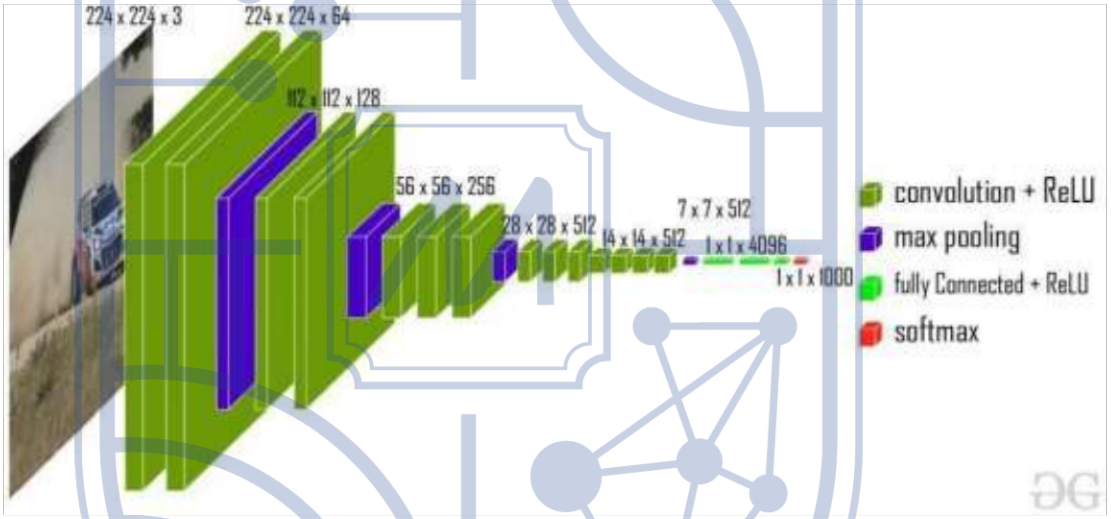
Gambar 2.5 R-CNN Architecture [35]

2.8 *Fast* R-CNN

Diperkenalkan oleh Ross Girshick pada tahun 2015, *Fast* R-CNN mengoptimalkan arsitektur R-CNN dengan berbagi komputasi di antara proposal region. Pertama, proposal region dihasilkan menggunakan algoritma selective search, yang dapat menghasilkan hingga sekitar 2000 proposal region. Proposal region ini (projeksi RoI) kemudian dikombinasikan

dengan gambar input dan diteruskan ke jaringan CNN, yang selanjutnya menghasilkan peta fitur konvolusi sebagai output[53]. Perbaikan utama meliputi:

1. Single Stage Processing: Alih-alih mengekstraksi fitur untuk setiap proposal wilayah secara terpisah, *Fast* R-CNN memproses seluruh gambar satu kali melalui CNN untuk menghasilkan feature map. Proposal wilayah kemudian diekstrak dari feature map bersama ini, sehingga mengurangi redundansi komputasi.
2. Softmax Classifier: *Fast* R-CNN menggantikan SVM dengan softmax classifier, yang memungkinkan pelatihan jaringan secara end-to-end.
3. Improved *Bounding Box* Regression: *Fast* R-CNN meningkatkan proses regresi *bounding box*, sehingga memberikan akurasi lokalisasi yang lebih baik [52].



Gambar 2.6 *Fast* R-CNN Architecture[36]

Didalam *Fast* R-CNN juga terdapat Teknik baru yaitu RoI (Region of Interest) pooling. Tujuannya adalah untuk menghasilkan peta fitur dengan ukuran tetap dari input yang bervariasi (RoI). RoI pooling menerima dua nilai sebagai input:

1. Peta fitur yang diperoleh dari lapisan CNN sebelumnya (misalnya, 14 × 14 × 512 pada VGG-16).
2. Matriks N × 4 yang mewakili region of interest (RoI), di mana **N** adalah jumlah RoI, dua nilai pertama menunjukkan koordinat sudut kiri atas RoI, dan dua nilai lainnya menunjukkan tinggi dan lebar RoI yang dinyatakan sebagai (r, c, h, w)[53].

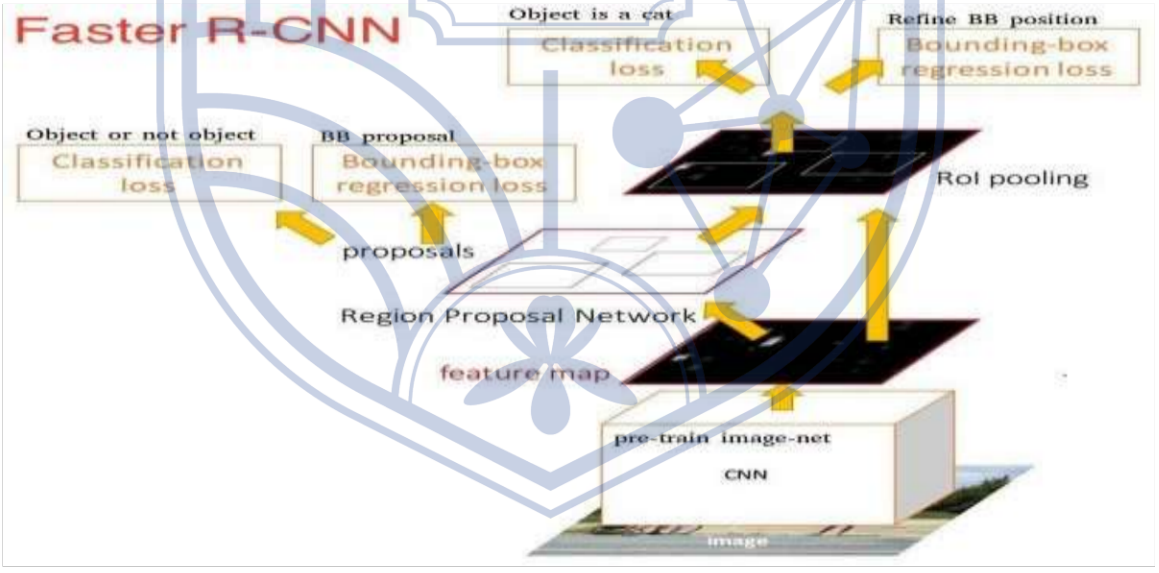
2.9 *Faster* R-CNN

Faster R-CNN, singkatan dari "*Faster Region-Convolutional Neural Network*," adalah arsitektur deteksi objek state-of-the-art dari keluarga R-CNN, yang diperkenalkan

oleh Shaoqing Ren, Kaiming He, Ross B. Girshick, dan Jian Sun pada tahun 2015. Tujuan utama dari jaringan *Faster* R-CNN adalah mengembangkan arsitektur yang tidak hanya mendeteksi objek dalam gambar tetapi juga menentukan lokasinya dengan tepat[54].

Convolutional Neural Networks (CNN) pertama-tama digunakan untuk mengklasifikasikan wilayah gambar input, dan *Faster* R-CNN mendeteksi wilayah-wilayah yang kemungkinan besar berisi objek yang akan dideteksi. Hal ini bertujuan untuk mengurangi biaya komputasi sekaligus meningkatkan akurasi. Selain itu, banyak proposal wilayah (region proposals) dihasilkan untuk meningkatkan kinerja dalam mendeteksi output yang diinginkan. Dalam artikel ini, detektor objek *Faster* R-CNN digunakan untuk mendeteksi suatu objek tertentu. Keuntungan utama dari penggunaan gambar pelatihan adalah untuk mengurangi waktu pemrosesan [54].

Faster R-CNN menggabungkan manfaat pembelajaran mendalam, *Convolutional Neural Networks* (CNNs), dan *Region Proposal Networks* (RPNs) ke dalam satu jaringan terpadu, yang secara signifikan meningkatkan kecepatan dan akurasi model. Melalui integrasi ini, *Faster* R-CNN mampu melakukan deteksi objek secara lebih cepat dan akurat dibandingkan arsitektur pendahulunya [55]. Berikut ini adalah gambaran dari komponen arsitekturnya.

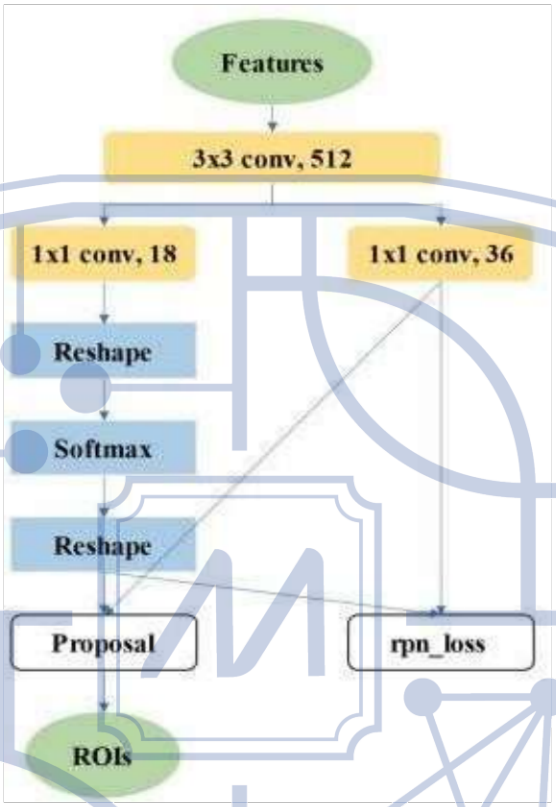


Gambar 2.7 Arsitektur *Faster* R-CNN [39]

2.9.1 Region Proposal Network (RPN)

RPN terdiri dari dua bagian yaitu bagian pertama menggunakan fungsi Softmax untuk mengklasifikasikan jangkar (anchor) sebagai positif atau negatif. Bagian kedua menghitung

regresi kotak pembatas (*bounding box* regression) untuk mendapatkan solusi yang lebih akurat. Setelah itu, lapisan proposal menggabungkan jangkar positif dengan offset regresi kotak pembatas terkait untuk menghasilkan rekomendasi dan proposal yang terlalu kecil atau berada di luar batas dihilangkan [56].



Gambar 2.8 The Network Structure Diagram of RPN [39]

Secara singkat, digunakan untuk menghasilkan kandidat daerah yang berpotensi mengandung objek. RPN membantu model dalam menentukan lokasi yang perlu dianalisis lebih lanjut. RPN mengurangi latensi hingga 10 kali lipat, memungkinkan deteksi objek secara real-time tanpa menambah beban komputasi yang signifikan [57].

Dari pernyataan diatas terdapat rumus yang dipakai dalam perhitungan Region Proposal Network (RPN) sebagai berikut:

$Jumlah\ anchor\ =\ H\ x\ W\ x\ k$ ..... (6)

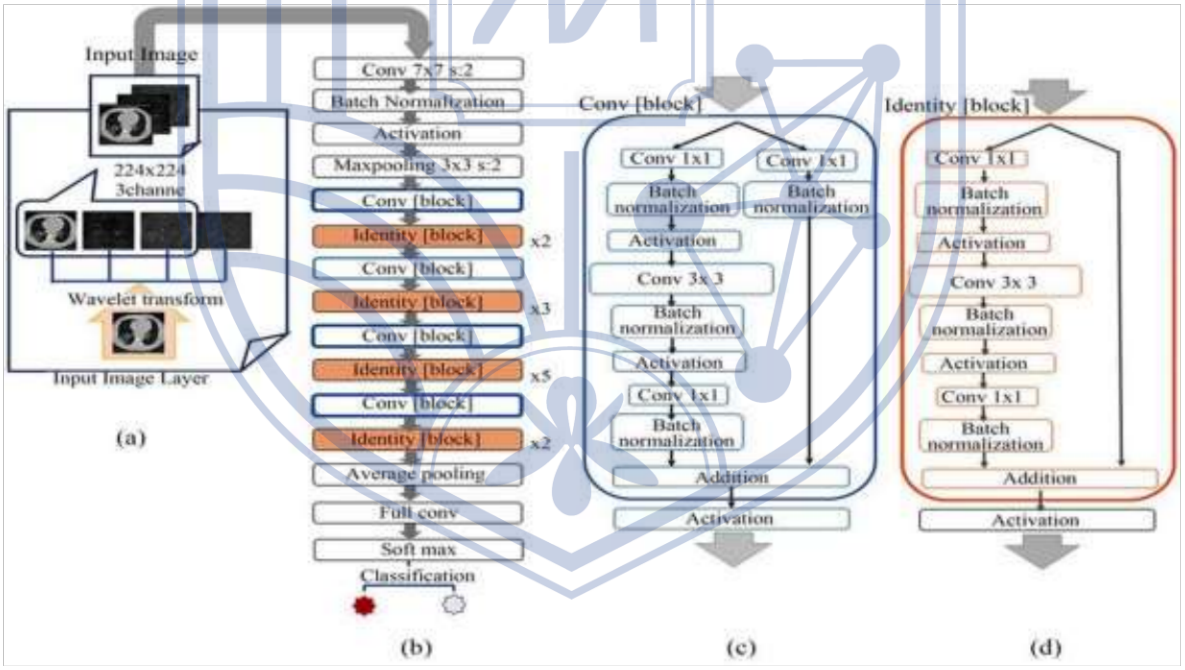
2.9.2 ResNet-50

ResNet-50 adalah arsitektur jaringan saraf konvolusi dalam (*deep convolutional neural network*) yang dirancang untuk mengatasi masalah degradasi kinerja pada jaringan yang sangat dalam. Model ini memperkenalkan *residual connections* untuk mempelajari *residual mappings* dan mengekstraksi fitur spasial tingkat tinggi dari gambar. ResNet-50

terdiri dari serangkaian lapisan konvolusi, lapisan normalisasi batch (*batch normalization*), lapisan *pooling*, dan lapisan *fully connected* [58].

Komponen inti dari ResNet-50 adalah *residual block*. Setiap *residual block* terdiri dari dua atau tiga lapisan konvolusi. Pada CNN tradisional, setiap lapisan konvolusi mentransformasikan input dan meneruskannya ke lapisan berikutnya. Namun, pada ResNet-50, setiap *residual block* memperkenalkan *skip connection* yang memungkinkan informasi diteruskan langsung antar lapisan yang berbeda. Secara spesifik, jika kita merepresentasikan input ke lapisan pertama sebagai x dan output setelah transformasi pada lapisan ke-n sebagai $H(x)$, maka *skip connection* menambahkan input x langsung ke output $H(x)$, menghasilkan residu $R(x) = H(x) + x$. =Residu $R(x)$ ini merepresentasikan perbedaan antara input dan output, yang merupakan bagian yang perlu dipelajari oleh jaringan. Dengan mempelajari residu, jaringan dapat lebih mudah dioptimalkan dan dilatih karena gradien dapat merambat lebih langsung [59]. Dari pernyataan diatas terdapat rumus yang dipakai dalam perhitungan ResNet-50 sebagai berikut:

$$W \sim N\ (0, \sqrt{\{\frac{2}{in\ feature}\}}) \quad (7)$$



Gambar 2.9 Arsitektur ResNet-50 [43]

2.9.3 Fungsi Loss

Fungsi Loss adalah jenis fungsi objektif, yang dalam konteks ilmu data mengacu pada fungsi apa pun yang meminimalkan atau memaksimalkan tujuan pelatihan model. Istilah

'fungsi loss', yang biasanya identik dengan fungsi biaya atau fungsi kesalahan, secara khusus mengacu pada situasi di mana minimalisasi adalah tujuan pelatihan untuk model machine learning. Secara sederhana, fungsi kerugian melacak tingkat kesalahan dalam hasil model kecerdasan buatan (AI). Hal ini dilakukan dengan mengukur perbedaan ('kerugian') antara nilai yang diprediksi-yaitu, hasil model-untuk input yang diberikan dan nilai aktual atau kebenaran dasar. Jika prediksi model akurat, kerugiannya kecil. Jika prediksinya tidak akurat, kerugiannya besar [60].

Fungsi loss dirancang untuk mencapai dua tujuan utama:

1. Mendekatkan elemen diagonal matriks cross-correlation ke 1, yang memastikan bahwa representasi dari dua tampilan sangat berkorelasi.
2. Mendekatkan elemen off-diagonal ke 0, yang mengurangi redundansi di antara komponen-komponen representasi yang dipelajari.

Metode ini tidak bergantung pada jumlah kelas yang tetap dan menghindari masalah ekspansi data, karena tidak memerlukan contoh negatif secara eksplisit. Namun, untuk mencapai performa yang baik, sering kali diperlukan dimensionalitas representasi akhir yang besar. Selain itu, ketahanan model dapat terpengaruh oleh distorsi tertentu pada input [61]. Berikut adalah jenis-jenis fungsi loss:

1. Cross Entropy Loss

Cross-entropy loss, yang juga dikenal sebagai log loss, adalah metrik yang digunakan dalam pembelajaran mesin untuk mengukur kinerja model klasifikasi. Nilainya berkisar antara 0 hingga 1, dengan nilai yang lebih rendah lebih baik. Nilai idealnya adalah 0. Tujuan dari sebuah pengoptimal (optimizer) yang bertugas melatih model klasifikasi dengan *cross-entropy loss* adalah membawa model sedekat mungkin ke 0. Dalam artikel ini, kita akan membahas *cross-entropy loss* untuk klasifikasi biner dan multikelas serta cara menginterpretasikannya [62]. Rumus Loss Cross-Entropy (CE):

$$H(y, y\hat{}) = -\frac{1}{N}\sum_{i=1}^{N} [y_i \cdot \log(y\hat{}_i) + (1 - y_i) \cdot \log(1 - y\hat{}_i)] \quad \text{.....(8)}$$

Penjelasan:

N = jumlah sampel dalam *dataset*.

y_i = label aktual dari sampel ke-i, dengan yi bernilai 0 atau 1.

$y\hat{}_i$ = nilai prediksi dari sampel ke-i, yang berkisar antara 0 dan 1.

2. Huber Loss

Huber loss menggabungkan sifat dari Mean Squared Error (MSE) dan *Mean Absolute Error* (MAE). Fungsi ini dirancang agar lebih tahan terhadap outlier dibandingkan

MSE, sambil tetap menjaga kelancaran dan dapat didiferensiasikan [61]. Rumus Huber Loss:

$$H_{\text{Huber}} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}(y_i - y_i)^2 & |y_i - y_i| < \delta \\ \frac{1}{N}\sum_{i=1}^{N} \delta\left(|y_i - y_i| - \frac{1}{2}\delta\right) & |y_i - y_i| > \delta \end{cases} \quad \text{.....(9)}$$

Penjelasan:

δ = hyperparameter yang menentukan transisi antara MSE dan MAE.

$\delta\ kecil$ = model akan lebih sering menggunakan MAE.

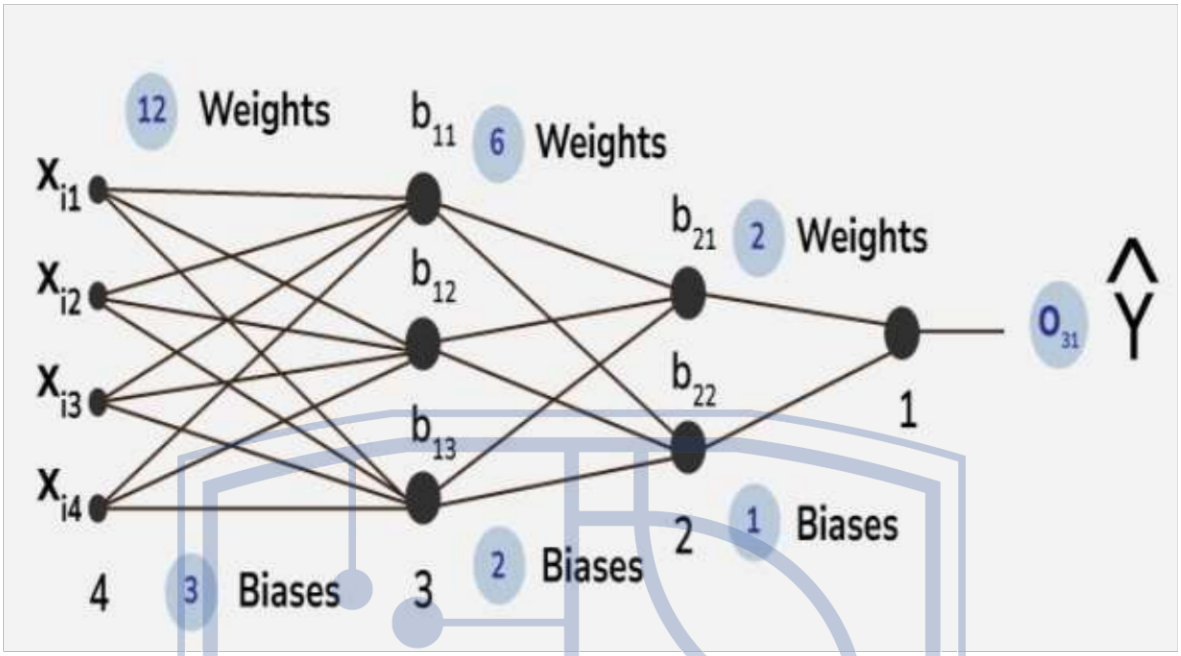
$\delta\ besar$ = model akan lebih sering menggunakan MSE.

2.9.4 *Forward Pass*

Forward pass merupakan proses dalam jaringan saraf tiruan di mana data masukan diteruskan melalui lapisan-lapisan jaringan hingga menghasilkan keluaran akhir. Proses ini terdiri dari beberapa tahapan utama:

1. Input Layer: Data masukan diberikan ke jaringan melalui lapisan input, yang mewakili fitur-fitur yang akan diproses oleh model.
2. Hidden Layer: Data masukan diteruskan ke satu atau lebih lapisan tersembunyi, di mana setiap neuron menerima sinyal dari neuron pada lapisan sebelumnya. Setiap neuron menghitung jumlah bobot dari sinyal masuk, menerapkan fungsi aktivasi untuk menghasilkan nilai yang diteruskan ke lapisan berikutnya.
3. Output Layer: Setelah diproses oleh lapisan tersembunyi, data mencapai lapisan output. Lapisan ini menerapkan fungsi aktivasi yang sesuai dengan jenis tugas yang dilakukan, seperti fungsi softmax untuk klasifikasi atau linear untuk regresi.
4. Prediction: Keluaran dari lapisan output merupakan hasil akhir dari proses *forward pass*, yang dapat berupa prediksi atau klasifikasi berdasarkan data masukan yang diberikan.

Forward pass merupakan tahap esensial dalam pembuatan prediksi menggunakan jaringan saraf. Hasil keluaran dari *forward pass* dibandingkan dengan nilai target untuk menghitung loss function. Nilai loss ini akan digunakan dalam proses *backpropagation* untuk memperbarui bobot dan bias jaringan selama pelatihan [63].



Gambar 2.10 Mathematical Forward Propagation [49]

Berikut rumus *forward pass*:

$$y^{\wedge} = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (10)$$

$$F = Backbone(I) \quad (11)$$

$$(S, B) = RPN(F) \quad (12)$$

$$R = RoI\ Pooling(F, B) \quad (13)$$

$$(C, B') = Head(R) \quad (14)$$

Penjelasan:

$I = input\ gambar$

$F = feature\ map$

$S = objectness\ score$

$B = bounding\ box\ proposal$

$R = hasil\ RoI\ Pooling$

$C = class\ score$

$B' = bounding\ box\ final$

2.9.5 *Backpropagation*

Algoritma *backpropagation* adalah metode untuk menyempurnakan bobot jaringan saraf dengan mempertimbangkan tingkat kesalahan yang diperoleh pada iterasi atau epoch sebelumnya. Ini adalah metode standar dalam pelatihan jaringan saraf buatan. Anda dapat membayangkannya sebagai sistem umpan balik di mana, setelah setiap putaran pelatihan

atau 'epoch,' jaringan meninjau kinerjanya dalam menyelesaikan tugas. Jaringan menghitung selisih antara output yang dihasilkan dan jawaban yang benar, yang disebut sebagai error. Kemudian, jaringan menyesuaikan parameter internalnya, atau 'bobot,' untuk mengurangi kesalahan pada iterasi berikutnya. Metode ini sangat penting dalam menyempurnakan akurasi jaringan saraf dan merupakan strategi dasar dalam pembelajaran untuk membuat prediksi atau keputusan yang lebih baik [64].

Backpropagation memainkan peran krusial dalam bagaimana jaringan saraf terus berkembang dan meningkatkan kinerjanya seiring waktu. Berikut alasannya:

1. Pembaruan Bobot yang Efisien: Algoritma ini menghitung gradien dari fungsi loss terhadap setiap bobot menggunakan aturan rantai (chain rule), sehingga memungkinkan pembaruan bobot secara efisien.
2. Skalabilitas: *Backpropagation* dapat diskalakan dengan baik untuk jaringan dengan banyak lapisan dan arsitektur yang kompleks, yang menjadikannya dasar bagi deep learning.
3. Pembelajaran Otomatis: Dengan *backpropagation*, proses pembelajaran menjadi otomatis, memungkinkan model menyesuaikan dirinya sendiri untuk mengoptimalkan kinerjanya [65].
4. Rumus menghitung Loss output klasifikasi:

$$\frac{\partial L_{cls}}{\partial z_i} = y_i - y_i \quad (15)$$

5. Rumus menghitung Loss output regresi:

$$\frac{\partial L_{reg}}{\partial x} = \begin{cases} x, & |x| \leq 1 \\ \text{sign}(x), & |x| \geq 1 \end{cases} \quad (16)$$

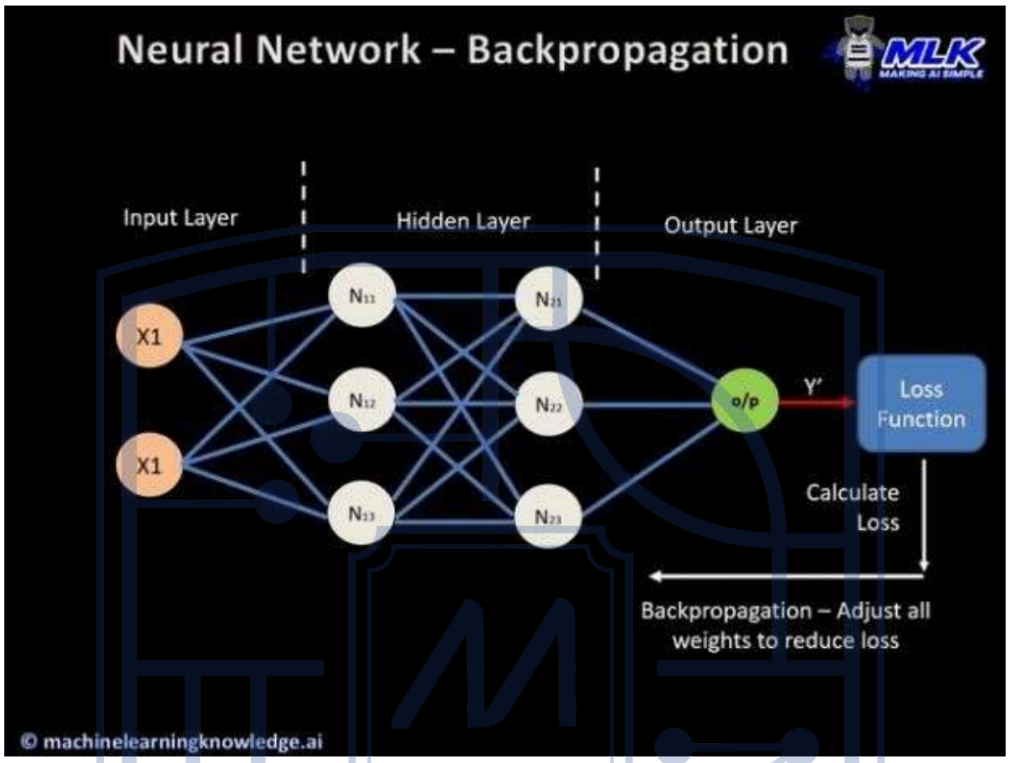
6. Rumus *update* parameter dengan Gradient Descent:

$$W_{\text{baru}} = W_{\text{lama}} - \alpha \frac{\partial L}{\partial W} \quad (17)$$

Meskipun *backpropagation* adalah algoritma yang kuat, ia menghadapi beberapa tantangan:

1. Masalah Vanishing Gradient: Dalam jaringan yang sangat dalam, gradien dapat menjadi sangat kecil selama proses *backpropagation*, sehingga menyulitkan jaringan untuk belajar. Hal ini sering terjadi saat menggunakan fungsi aktivasi seperti sigmoid atau tanh.
2. Gradien yang Meledak (Exploding Gradients): Sebaliknya, gradien juga bisa menjadi terlalu besar, menyebabkan jaringan mengalami divergensi selama pelatihan.

3. Overfitting: Jika jaringan terlalu kompleks, ia mungkin hanya menghafal data pelatihan daripada belajar pola yang lebih umum, sehingga kinerjanya menurun pada data baru [65].



Gambar 2.11 Ilustrasi *Backpropagation* [50]

2.9.6 *Fine Tuning*

Fine-tuning adalah proses menyesuaikan model yang telah dilatih sebelumnya (pre-trained model) agar lebih sesuai dengan data. Proses ini memungkinkan memanfaatkan data secara maksimal dan meningkatkan kinerja model [66].

Fine tuning dapat dianggap sebagai bagian dari teknik pembelajaran transfer yang lebih luas: praktik memanfaatkan pengetahuan yang telah dipelajari oleh model yang sudah ada sebagai titik awal untuk mempelajari tugas-tugas baru. Dengan memanfaatkan pelatihan model sebelumnya melalui pembelajaran transfer, fine tuning dapat mengurangi jumlah daya komputasi yang mahal dan data berlabel yang diperlukan untuk mendapatkan model besar yang disesuaikan dengan kasus penggunaan dan kebutuhan bisnis. Misalnya, fine tuning dapat digunakan untuk sekadar menyesuaikan nada percakapan LLM yang telah dilatih sebelumnya atau gaya ilustrasi model pembuatan gambar yang telah dilatih sebelumnya; ini juga dapat digunakan untuk menambah pembelajaran dari *dataset* pelatihan asli model dengan data eksklusif atau pengetahuan khusus yang spesifik untuk domain tertentu [67].

fine tuning memerlukan teknik untuk melatih lebih lanjut model yang bobotnya telah diperbarui melalui pelatihan sebelumnya. Dengan menggunakan pengetahuan model dasar sebelumnya sebagai titik awal, penyempurnaan menyesuaikan model dengan melatihnya pada kumpulan data yang lebih kecil dan spesifik untuk tugas tertentu. Sementara kumpulan data khusus tugas secara teoretis dapat digunakan untuk pelatihan awal, melatih model besar dari awal pada kumpulan data kecil berisiko overfitting: model mungkin belajar untuk melakukan dengan baik pada contoh pelatihan, tetapi generalisasi buruk untuk data baru. Ini akan membuat model tidak sesuai dengan tugas yang diberikan dan mengalahkan tujuan pelatihan model [67].

Pada pernyataan diatas terdapat rumus sederhana untuk menghitung tingkat akurasi fine tuning untuk mengoptimalkan model *Faster* R-CNN:

1. Classification Report:

$$L_{\text{cls}} = \sum y_i \log(\hat{y_i}) \quad (18)$$

$$\frac{\partial L_{\text{cls}}}{\partial z_i} = \hat{y_\iota} - y_i \quad (19)$$

2. Regression Loss:

$$L_{\text{reg}} = \begin{cases} 0.5(b-\hat{b})^2, & |b-\hat{b}| < 1 \\ |b-\hat{b}| - 0.5, & |b-\hat{b}| \geq 1 \end{cases} \quad (20)$$

3. Optimizer Stochastic Gradient Descent (SGD):

$$W_{\text{baru}} = W_{\text{lama}} - \alpha \cdot \frac{\partial L}{\partial W} \quad (21)$$

Penjelasan:

a. $\alpha = learning\ rate$

b. $\frac{\partial L}{\partial W}$ $gradien\ loss\ terhadap\ bobot\ model$

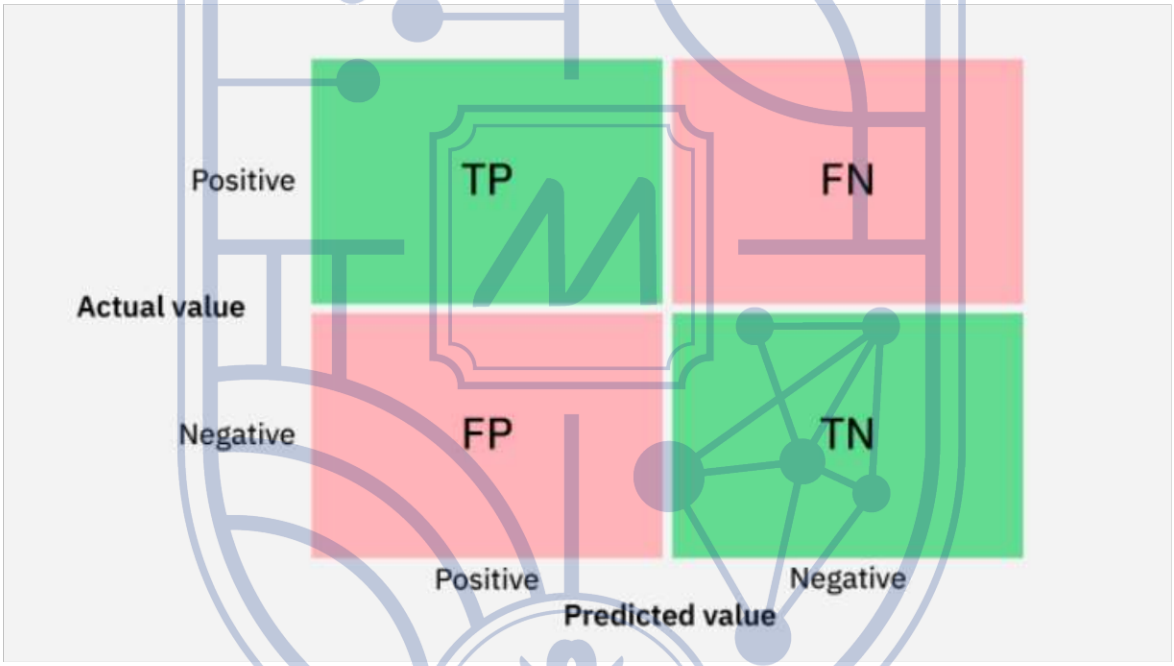
2.10 Confusion Matrix

Confusion matrix adalah metode visualisasi untuk hasil algoritma pengklasifikasi. Secara lebih spesifik, matriks ini berbentuk tabel yang menunjukkan jumlah instance ground truth (data aktual) dari suatu kelas tertentu dibandingkan dengan jumlah instance kelas yang diprediksi. Confusion matrix merupakan salah satu metrik evaluasi yang digunakan untuk mengukur kinerja model klasifikasi. Matriks ini juga dapat digunakan untuk menghitung metrik kinerja model lainnya, seperti presisi (*precision*) dan *recall*, serta metrik-metrik lainnya [68].

Saat menilai kinerja model klasifikasi, Confusion matrix sangatlah penting. Matriks ini menawarkan analisis menyeluruh terhadap prediksi positif benar, negatif benar, positif

salah, dan negatif salah, yang memfasilitasi pemahaman yang lebih mendalam tentang ingatan, akurasi, presisi, dan efektivitas keseluruhan model dalam pembedaan kelas. Saat terdapat distribusi kelas yang tidak merata dalam kumpulan data, matriks ini sangat membantu dalam mengevaluasi kinerja model di luar metrik akurasi dasar [69].

Dalam confusion matrix, kolom merepresentasikan nilai prediksi dari suatu kelas, sedangkan baris merepresentasikan nilai aktual (ground truth) dari kelas tersebut, atau sebaliknya [68]. Matriks ini merupakan cara untuk menampilkan jumlah kejadian akurat dan tidak akurat berdasarkan prediksi model. Matriks ini sering digunakan untuk mengukur kinerja model klasifikasi, yang bertujuan untuk memprediksi label kategoris untuk setiap kejadian input [69].



Gambar 2.12 Confusion Matriks [59]

Matriks menampilkan jumlah contoh yang diproduksi oleh model pada data uji.

1. Positif Sejati (TP): Model dengan tepat meramalkan hasil positif (hasil aktualnya positif).
2. Negatif Sejati (TN): Model dengan tepat meramalkan hasil negatif (hasil aktualnya negatif).
3. False Positive (FP): Model salah memprediksi hasil positif (hasil sebenarnya negatif). Dikenal juga sebagai kesalahan Tipe I.
4. Negatif Palsu (FN): Model secara keliru memprediksi hasil negatif (hasil sebenarnya positif). Dikenal juga sebagai kesalahan Tipe II [69].

Penjelasan masing-masing metrik dalam confusion matriks:

1. Accuracy: Akurasi mengukur seberapa sering prediksi model sesuai dengan nilai sebenarnya secara keseluruhan. Accuracy dihitung dengan rumus berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{..... (22)}$$

2. *Precision*: mengukur seberapa banyak prediksi yang benar untuk suatu kelas dibandingkan dengan total prediksi yang diberikan model untuk kelas tersebut. *Precision* dihitung dengan rumus berikut:

$$precision = \frac{TP}{TP+FP} \quad \text{..... (23)}$$

3. *Recall*, atau dikenal sebagai Sensitivity atau True Positive Rate, mengukur sejauh mana model dapat menemukan semua sampel dari suatu kelas dalam *dataset*. *Recall* dihitung dengan rumus berikut:

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (24)}$$

4. *F1-score* merupakan rata-rata harmonik antara *precision* dan *recall*, yang digunakan untuk memberikan keseimbangan antara kedua metrik tersebut. *F1-score* sangat berguna dalam kondisi class imbalance, di mana terdapat ketimpangan jumlah sampel antara kelas yang berbeda. Nilai *F1-score* dihitung menggunakan rumus:

$$F1 - Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad \text{.....(25)}$$

5. *Support* menunjukkan jumlah total sampel dalam setiap kelas yang terdapat dalam *dataset* aktual. Nilai ini diperoleh dengan menjumlahkan seluruh baris pada confusion matrix untuk kelas ke-i [70].