BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Dalam pengerjaan proyek akhir ini, *review* penelitian terdahulu digunakan sebagai sumber masukan dan ide yang relevan dengan penelitian yang akan dilakukan. Terdapat 5 penelitian yang digunakan sebagai perbandingan dalam proyek akhir ini.

Penelitian pertama oleh Alidrus dkk. (2021) dengan judul "Deteksi Penyakit Pada Daun Tanaman Padi Menggunakan Metode *Convolutional Neural Network* (CNN)". Proses diagnosis dilakukan dengan klasifikasi CNN dengan input berupa citra daun tanaman padi sebanyak 900 sampel yang terdiri dari tiga kelas penyakit, yaitu *blast*, *brown spot*, dan *hispa*. Data citra melalui proses *preprocessing* seperti *resizing* dan *cropping* sebelum memasuki tahap pemodelan. Model CNN yang dibangun terdiri dari 3 *convolutional layers*, 3 *max pooling layers*, dan 2 *fully connected layers*. Pengujian dilakukan dengan menggunakan 720 data latih dan 180 data validasi. Hasil pengujian menunjukkan akurasi rata-rata sebesar 92% pada data latih dan 77% pada data validasi. Model ini masih menunjukkan gejala *overfitting* dalam mengklasifikasikan citra.

Penelitian kedua oleh Jinan dan Hayadi (2022) dengan judul "Klasifikasi Penyakit Tanaman Padi Menggunakan Metode *Convolutional Neural Network* Melalui Citra Daun (*Multilayer Perceptron*)". Penelitian ini bertujuan untuk mengklasifikasikan penyakit daun padi menggunakan metode CNN. Dataset yang digunakan terdiri dari 120 citra daun padi yang terbagi menjadi tiga kelas, yaitu *Bacterial Leaf Blight*, *Brown Spot*, dan *Leaf Spot*. Penelitian ini menggunakan skenario pengujian dengan *preprocessing* berupa normalisasi dan *resize* citra menjadi ukuran 100x100 piksel. Model CNN diuji dengan parameter seperti kernel 3x3, *learning rate* 0,01, *optimizer Adam*, *batch size* 30, dan 150 *epoch*. Hasil ujicoba menunjukkan akurasi sebesar 91,7%.

Penelitian ketiga oleh Sulistiyana dan Anardani (2023) dengan judul "Aplikasi Deteksi Penyakit Tanaman Jagung Dengan *Convolutional Neural Network* dan *Support Vector Machine*". Pada penelitian ini menggunakan dua metode yaitu CNN dan SVM dengan input berupa citra daun jagung yang telah melalui proses *preprocessing* berupa augmentasi data seperti *rescale*, *rotate*, *zoom*, dan *flip*. Dataset yang digunakan terdiri dari 3.732 citra yang terbagi ke dalam 4 kelas yaitu *blight*, *fungus*, *rust*, dan *healthy*. Model CNN diuji menggunakan 100

epoch dan menghasilkan akurasi tertinggi sebesar 98%, *precision* 98%, *recall* 98% dan *f1-score* 98%, sedangkan model SVM menghasilkan akurasi sebesar 87%, *precision* 87%, *recall* 87% dan *f1-score* 86%. Hasil penelitian menunjukkan bahwa CNN memiliki performa lebih baik dibandingkan SVM dalam mendeteksi penyakit tanaman jagung.

Penelitian keempat oleh Shinta dkk. (2023) dengan judul *"*Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur *VGG-19"*. Penelitian ini mendeteksi penyakit daun tanaman padi yang biasanya dilakukan secara manual, yang membutuhkan waktu relatif lama dan akurasi yang kurang konsisten. Hal tersebut diatasi dengan memanfaatkan metode CNN dengan arsitektur *VGG-19*. Penelitian ini menggunakan dataset sebanyak 440 citra asli dan 1320 citra hasil augmentasi yang terbagi menjadi empat kelas yaitu *blast*, *brown spot*, *leaf smut*, dan *healthy*. Hasil pengujian menunjukkan akurasi tertinggi sebesar 94,31% pada data augmentasi dengan *precision* sebesar 88.00%, *recall* sebesar 100.0%, dan *f1-score* sebesar 93.61%. Sedangkan, akurasi tertinggi tanpa augmentasi adalah 93,18%, *precision* 100% *recall* 81.81%, dan *f1-score* 89.99%.

Penelitian kelima oleh Bukhari (2024) dengan judul *"*Implementasi Metode *Convolutional Neural Network* (CNN) untuk Diagnosa Penyakit Tanaman Cabai pada Citra Daun*"*. Penelitian ini bertujuan untuk mendiagnosa penyakit tanaman cabai menggunakan dataset citra daun cabai dengan metode CNN berarsitektur *AlexNet*. Penelitian ini menggunakan 5 kelas, yaitu *leaf curl*, *leaf spot*, *whitefly*, *yellowish*, dan kategori sehat. Proses penelitian melibatkan 3 tahap, yaitu tahap *preprocessing* dengan penyesuaian ukuran citra menjadi 128x128 piksel, tahap ekstraksi fitur dengan 3 layer yaitu *convolutional layer*, *pooling layer*, dan f*ully connected layer*, serta pelatihan model. Fungsi aktivasi yang digunakan adalah *Rectification Linear Unit* (ReLU). Hasil akurasi dari penelitian ini menunjukkan nilai tertinggi sebesar 100% untuk *leaf curl* dan *leaf spot*, 90% untuk *whitefly*, 80% untuk kategori sehat, dan 70% untuk *yellowish.*

Tabel 2.1 Penelitian Terdahulu

Penelitian	Dataset	Algoritma		Output
Deteksi Penyakit Pada Daun Tanaman Padi Menggunakan Metode	Citra daun padi	CNN		Pada tahapan implement asi CNN mengguna

Penelitian	Dataset	Algoritma		Output
Convolutional Neural Network (CNN) (Alidrus dkk., 2021)				kan *optimizer RMSprop*, *drop out rate* 0,4 dan 50 *epoch* dengan rata-rata akurasi yang didapatkan sebesar 92% pada data latih dan 77% pada data validasi
Klasifikasi Penyakit Tanaman Padi Menggunakan Metode *Convolutional Neural Network* (CNN) Melalui Citra Daun (Jinan & Hayadi, 2022)	Citra daun padi	CNN		Dengan menggunakan *optimizer Adam*, *learning rate* 0,01 dan 150 *epoch* dengan nilai akurasi yang didapatkan sebesar 91,7% dan *loss* sebesar 8,8%.
Aplikasi Deteksi Penyakit Tanaman Jagung Dengan *Convolutional Neural Network* dan *Support Vector Machine* (Sulistiyana & Anardani, 2023)	Citra daun jagung	CNN dan SVM		Model CNN mencapai akurasi 98%, *precision* 98%, *recall* 98% dan *f1-score* 98%, sementara SVM mencapai akurasi 87%, *precision* 87%, *recall* 87% dan *f1-score* 86%.
Klasifikasi Citra Penyakit Daun Tanaman	Citra daun padi	CNN berarsitektur VGG-19		Akurasi tertinggi dengan

Penelitian	Dataset	Algoritma		Output
Padi Menggunakan CNN dengan Arsitektur VGG-19 (Shinta dkk., 2023)				augmentasi 94,31%, *precision* 88.00%, *recall* 100.0%, dan *f1-score* 93.61%, sedangkan tanpa augmentasi akurasi sebesar 93,18%; *precision* 100% *recall* 81.81%, dan *f1-score* 89.99%.
Implementasi Metode *Convolutional Neural Network* (CNN) untuk Diagnosa Penyakit Tanaman Cabai pada Citra Daun (Bukhari, 2024)	Citra daun cabai	CNN berarsitek-tur *AlexNet*		Akurasi validasi masing-masing kategori dimana *leaf curl* dan *leaf spot* (100%), *whitefly* (90%), *healthy* (80%), dan *yellowish* (70%), model menunjukkan perlu peningkatan pada kategori sehat dan *yellowish*.

Pada penelitian terdahulu, terdapat pengembangan yang perlu mendapatkan perhatian dalam deteksi penyakit tanaman padi, khususnya pada metode berbasis CNN. Penelitian sebelumnya telah menunjukkan efektivitas CNN dalam mendeteksi penyakit tanaman, tetapi terdapat keterbatasan seperti jumlah data yang relatif kecil dan kelas penyakit yang terbatas. Variasi latar belakang dan pencahayaan citra sering

menjadi kendala yang mempengaruhi performa model. Selain itu, penelitian sebelumnya tidak melanjutkan ke tahap integrasi pada *platform mobile* berbasis *android*. Penelitian ini akan menggunakan dataset dengan jumlah data yang lebih besar dan mencakup lebih banyak kategori, seperti *bacterial blight*, *blast, brown spot*, dan *tungro*, untuk mengetahui kemampuan model dalam mengenali pola gejala yang lebih kompleks. Dengan data yang lebih beragam, model diharapkan mampu menangkap fitur penting dari setiap penyakit. Aplikasi *mobile* pendeteksi penyakit padi dikembangkan untuk memberi kemudahan akses bagi petani di lapangan. Dengan memperhatikan gap-gap tersebut, penelitian ini diharapkan dapat lebih efektif dalam mengimplementasikan deteksi penyakit tanaman padi.

2.2 Landasan Teori

2.2.1 Penyakit Tanaman Padi

Tanaman padi adalah salah satu komoditas pertanian paling penting terutama di negara-negara Asia seperti Indonesia, yang bergantung pada padi sebagai sumber pangan utama. Sebagai makanan utama bagi sebagian besar penduduk, keberhasilan panen padi sangat menentukan ketahanan pangan suatu negara. Namun, produktivitas tanaman padi sering kali menghadapi ancaman serius dari berbagai penyakit yang dapat menyebabkan kerugian besar baik secara kuantitas maupun kualitas hasil panen (Mohidem dkk., 2022). Beberapa penyakit umum yang menyerang tanaman padi, yaitu *bacterial blight*, *blast, brown spot*, dan *tungro*.

1) *Bacterial Blight*

Gambar 2.1 *Bacterial Blight*

Bacterial blight atau hawar bakteri, merupakan salah satu penyakit utama pada tanaman padi yang disebabkan oleh bakteri *Xanthomonas oryzae*. Penyakit ini biasanya ditandai

dengan munculnya lesi memanjang berwarna kuning hingga coklat pada daun padi. Seiring waktu, lesi tersebut dapat menyebar ke seluruh daun menyebabkan layu hingga kematian tanaman. Faktor lingkungan seperti kelembaban tinggi sering memperburuk penyebaran penyakit ini. Metode pengendalian yang umum digunakan meliputi penanaman varietas tahan penyakit, penerapan praktek agronomi yang baik, dan penggunaan agen pengendali hayati yang ramah lingkungan (Nugroho dkk., 2024).

2) *Blast*

Gambar 2.2 *Blast*

Blast adalah penyakit yang disebabkan oleh jamur *Magnaporthe oryzae* yang sering dianggap sebagai salah satu penyakit paling merusak pada tanaman padi. Penyakit ini menyerang berbagai bagian tanaman, seperti daun, batang, hingga leher malai, dengan gejala khas berupa lesi berbentuk berlian dengan tepi coklat dan pusat abu-abu. Penyakit ini berkembang pesat dalam kondisi lingkungan yang lembab dengan suhu sedang, terutama di wilayah tropis seperti Indonesia. Dampak penyakit ini bisa sangat signifikan, termasuk kerugian hasil panen secara besar-besaran. Pengendalian penyakit *blast* biasanya dilakukan dengan menggunakan varietas padi tahan penyakit, rotasi tanaman untuk memutus siklus patogen, dan aplikasi fungisida sesuai dosis yang direkomendasikan (Kumar dkk., 2021).

3) *Brown Spot*

Gambar 2.3 *Brown Spot*

Brown spot atau bercak coklat adalah penyakit yang disebabkan oleh jamur *Bipolaris oryzae*. Gejala utama penyakit ini adalah munculnya bercak bulat hingga oval berwarna coklat pada daun. Pada serangan yang parah, bercak-bercak ini dapat menyatu menyebabkan kerusakan besar pada permukaan daun hingga mengakibatkan nekrosis. Penyakit ini sering terjadi pada tanaman padi yang kekurangan nutrisi, khususnya nitrogen, atau pada kondisi lingkungan yang kurang mendukung pertumbuhan tanaman. Pengelolaan penyakit ini melibatkan strategi seperti pemberian pupuk nitrogen yang cukup, pengelolaan air yang optimal, serta penggunaan varietas padi yang tahan terhadap jamur ini (Kumar dkk., 2021).

4) *Tungro*

Gambar 2.4 *Tungro*

Tungro disebabkan oleh infeksi gabungan dua jenis virus, yaitu *rice tungro bacilliform virus* (RTBV) dan *rice tungro spherical virus* (RTSV), yang ditularkan melalui perantara serangga wereng hijau (*Nephotettix virescens*). Gejala umum

tungro dapat diketahui dari pertumbuhan tanaman yang terhambat, daun menguning dari ujung ke pangkal, serta malai yang pendek dan tidak berisi. Penyakit ini dapat menurunkan hasil panen yang banyak terutama pada serangan berat. Pengendalian *tungro* dapat dilakukan melalui penanaman varietas tahan, pengelolaan populasi wereng secara terpadu, serta penggunaan benih sehat (Rahmawati dkk., 2022).

2.2.2 *Preprocessing*

Preprocessing merupakan langkah penting dalam pemrosesan citra sebelum data diumpankan ke model pembelajaran mesin atau deep learning. Tujuan dari *preprocessing* adalah untuk meningkatkan kualitas data sehingga model dapat bekerja secara lebih efektif dan efisien. Menurut Setiawan & Hartono (2023) beberapa operasi *preprocessing* yang biasa dilakukan sebagai berikut:

1) Normalisasi

Normalisasi adalah metode yang digunakan untuk mengubah rentang nilai piksel citra menjadi standar yang lebih seragam, biasanya dalam kisaran 0 hingga 1. Normalisasi membantu mempercepat proses pelatihan model dengan memastikan bahwa semua fitur citra memiliki bobot yang seimbang. Hal ini penting karena jika nilai-nilai piksel bervariasi secara drastis, model bisa cenderung lebih fokus pada fitur dengan nilai yang lebih besar dan mengabaikan fitur yang lebih kecil.

2) *Contrast Limited Adaptive Histogram Equalization* (CLAHE)

CLAHE adalah metode peningkatan kontras yang digunakan untuk meningkatkan kualitas visual citra dengan cara membagi citra menjadi beberapa bagian dan kemudian menerapkan *histogram equalization* pada masing-masing bagian. Berbeda dengan metode peningkatan kontras biasa, CLAHE membatasi peningkatan kontras sehingga area gambar yang terang tidak menjadi terlalu terang. Metode ini sangat bermanfaat

ketika citra memiliki kontras rendah atau pencahayaan yang buruk, misalnya pada gambar medis atau citra satelit.

3) Augmentasi

Augmentasi adalah proses memperbanyak variasi data *train* dengan mengubah gambar yang sudah ada. Tujuannya agar model dapat mengenali objek dalam kondisi yang lebih beragam, seperti sudut pandang yang berbeda atau pencahayaan yang berubah. Metode ini sangat bermanfaat dalam *deep learning* terutama jika jumlah data terbatas. Beberapa jenis augmentasi yang sering digunakan seperti rotasi, *zoom*, *flipping* (membalik gambar), serta perubahan kecerahan atau posisi. Dengan demikian, augmentasi membantu meningkatkan kemampuan generalisasi model dan mengurangi risiko *overfitting* selama proses pelatihan.

2.2.3 *Deep Learning (DL)*

Deep learning adalah bagian dari *machine learning* yang bekerja dengan jaringan saraf buatan yang terdiri dari banyak lapisan. Model ini dirancang untuk memahami pola data secara bertahap, mulai dari fitur sederhana hingga yang lebih kompleks. Salah satu keunggulan DL adalah kemampuannya untuk mempelajari pola data tanpa membutuhkan proses manual untuk mengekstrak fiturnya, yang membuatnya efektif untuk berbagai aplikasi seperti pengenalan gambar, analitis, dan pengenalan gambar (Sarker, 2021).

DL memiliki kelebihan dalam menangani data berukuran besar dan kompleks karena didukung oleh perangkat keras kontemporer seperti GPU dan TPU yang dapat mempercepat proses pelatihan model. DL dapat membangun representasi data secara hierarkis dari fitur dasar hingga fitur tingkat tinggi yang lebih kompleks, berkat banyak lapisan arsitekturnya. Kemampuan ini memungkinkan DL untuk membuat prediksi yang akurat dan memahami data secara menyeluruh (Alzubaidi dkk., 2021).

DL dapat diterapkan pada berbagai jenis pembelajaran, termasuk *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Dalam s*upervised learning*, DL membantu menghubungkan input dengan *output* yang ditargetkan, sedangkan pada *unsupervised learning* membantu menemukan pola tersembunyi dalam data. Dalam *reinforcement learning*, DL membantu meningkatkan pengambilan

keputusan melalui pengalaman yang diperoleh dari interaksi (Sarker, 2021).

Menurut Alzubaidi dkk (2021) beberapa varian arsitektur DL telah digunakan secara luas di berbagai bidang.

1) *Deep Belief Networks* (DBN)

DBN adalah arsitektur jaringan saraf tiruan yang terdiri dari beberapa lapisan Restricted Boltzmann Machines (RBM), di mana setiap lapisan belajar untuk menangkap fitur-fitur penting dari data masukan. Lapisan-lapisan ini disusun secara bertahap, memungkinkan DBN untuk menghasilkan representasi fitur yang lebih abstrak. DBN banyak digunakan dalam aplikasi pengenalan pola, pemrosesan citra, dan analisis data medis karena kemampuannya untuk mengidentifikasi struktur tersembunyi dalam data. Dalam proses pelatihan DBN, metode pra-pelatihan tanpa pengawasan digunakan untuk memaksimalkan akurasi klasifikasi (Nurbaiti & Suryanto, 2023).

2) *Recurrent Neural Networks* (RNN)

RNN adalah jaringan saraf tiruan yang dirancang untuk memproses data sekuensial atau deret waktu. Berbeda dengan jaringan saraf konvensional, RNN memiliki umpan balik internal yang memungkinkan mereka menyimpan informasi dari satu langkah ke langkah berikutnya. Hal ini membuat RNN sangat berguna untuk tugas-tugas seperti pemrosesan bahasa alami, prediksi deret waktu, dan pengenalan ucapan. Namun, masalah seperti "vanishing gradient" dapat mengganggu kinerja RNN ketika memproses ketergantungan jangka panjang, yang diatasi dengan menggunakan LSTM atau GRU (Rahmawati & Widodo, 2024).

3) *Generative Adversarial Networks* (GANs)

GAN terdiri dari dua jaringan saraf yang bersaing: generator dan diskriminator. Generator bertujuan untuk menghasilkan data yang menyerupai data nyata, sedangkan diskriminator bertugas membedakan antara data nyata dan palsu. Proses pelatihan GAN ini berlangsung sebagai kompetisi antara dua jaringan, sehingga menghasilkan model yang mampu menciptakan data sintetis yang realistis. Di Indonesia, GAN telah

digunakan dalam pengembangan citra sintetis untuk keperluan pengenalan wajah dan peningkatan resolusi gambar medis (Sudibyo & Permana, 2023).

4) *Deep Reinforcement Learning* (DRL)

DRL adalah kombinasi dari pembelajaran penguatan dan deep learning, yang memungkinkan agen untuk belajar dari interaksi dengan lingkungan melalui eksplorasi dan eksploitasi tindakan yang memberikan reward maksimal. DRL telah diterapkan dalam berbagai aplikasi seperti robotika, kontrol otomatis, dan permainan video. Salah satu keunggulan utama DRL adalah kemampuannya untuk menangani lingkungan yang kompleks dan dinamis dengan memanfaatkan jaringan saraf dalam untuk menghasilkan kebijakan yang optimal (Smith & Lee, 2024).

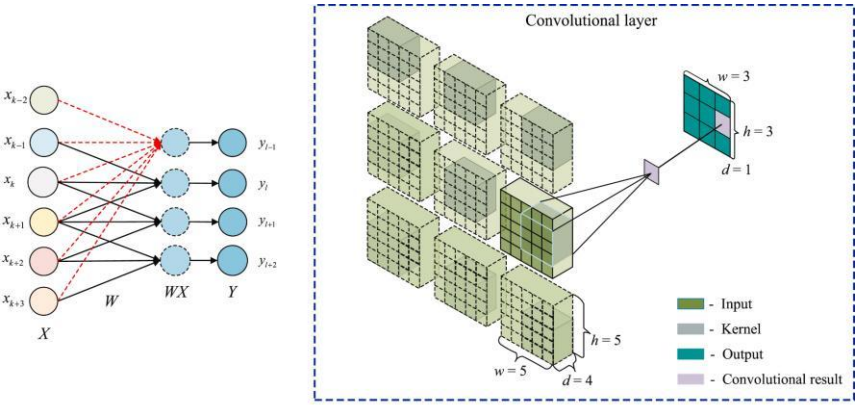
5) *Stacked Denoising Autoencoders* (SDA)

SDA adalah varian dari *autoencoder* yang dilatih dengan menambahkan noise pada input data. Tujuan dari penambahan noise ini adalah untuk mempelajari representasi yang lebih robust dan dapat memperbaiki data yang terganggu oleh noise. SDA telah digunakan secara luas dalam reduksi dimensi, deteksi anomali, dan pemrosesan citra. Penggunaan SDA memungkinkan jaringan untuk mempelajari fitur penting dari data meskipun data tersebut terkontaminasi oleh gangguan (Dewi & Saputra, 2023).

2.2.4 *Convolutional Neural Network* (CNN)

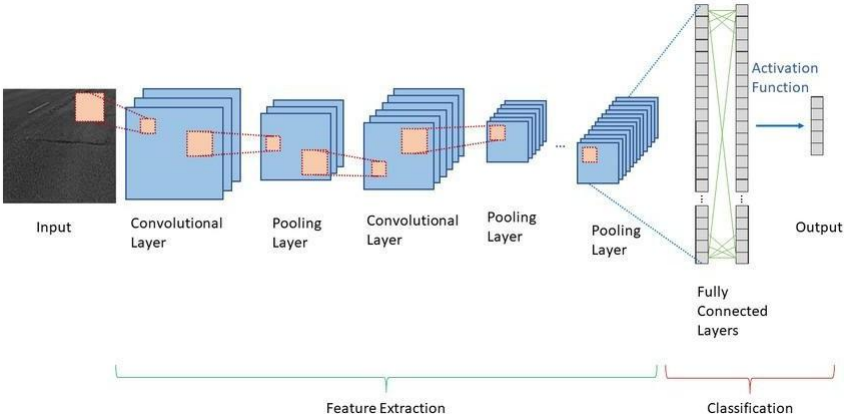
Menurut Rahayu dkk. (2024), metode DL semakin banyak diterapkan dalam pengolahan citra karena kemampuannya memberikan akurasi yang sangat tinggi. Salah satu metode DL yang sering digunakan untuk klasifikasi citra adalah *Convolutional Neural Network* (CNN). CNN terdiri dari tiga jenis proses atau lapisan, yaitu *convolution layer*, *pooling layer*, dan *fully connected layer* (*dense layer*). Pada *convolution layer*, dilakukan serangkaian operasi konvolusi 2D dengan fungsi aktivasi *Rectified Linear Unit* (ReLU). Proses ini menggunakan kernel dengan ukuran tertentu (biasanya 3 × 3 piksel), menghasilkan beberapa *feature map* pada setiap lapisan konvolusi. Proses ini melibatkan *filtering* antara setiap input *channel* dengan kernel tertentu, di mana hasilnya

dijumlahkan untuk menghasilkan *output channel*. Ilustrasi mengenai proses ini sering digunakan untuk menjelaskan bagaimana *feature map* terbentuk.



Gambar 2.5 (a) Skematik Koneksi Lapisan Konvolusi (b) Operasi Konvolusi dalam CNN

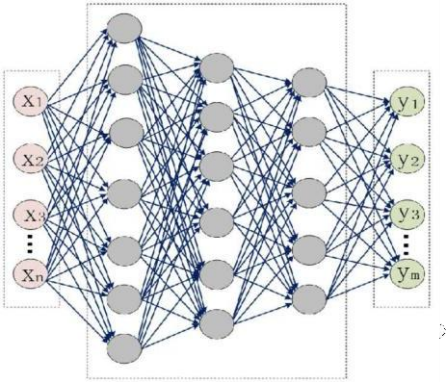
dari input, seperti yang dijelaskan oleh Adiwijaya et al. (2021). Model CNN terdiri dari *convolution layer* yang bertugas mengekstraksi fitur lokal, yang kemudian dilanjutkan dengan *pooling layer* untuk menyederhanakan representasi fitur dan mengurangi dimensi data. *Pooling layer* biasanya ditempatkan setelah lapisan konvolusi untuk mempertahankan fitur penting sambil mengurangi komputasi yang dibutuhkan. Lapisan terakhir adalah *fully connected layer*, yang mengintegrasikan fitur-fitur yang telah dipelajari dan melakukan klasifikasi menggunakan fungsi *softmax*. CNN dilatih menggunakan metode *backpropagation*, biasanya dengan algoritma *Stochastic Gradient Descent* (SGD), untuk mengoptimalkan bobot dan bias sehingga meminimalkan fungsi kerugian (Albelwi & Mahmood, 2017).



Gambar 2.6 Arsitektur CNN

Pooling layer dalam CNN memiliki peran penting dalam mengurangi dimensi data sekaligus mempertahankan fitur-fitur penting. *Pooling* dilakukan dengan membagi *feature map* menjadi area kecil (*window*), biasanya berukuran 2 × 2 piksel. Dari setiap area, satu nilai statistik tertentu diambil untuk merepresentasikan area tersebut. Pada *max pooling*, nilai maksimum diambil untuk mempertahankan fitur dominan, sementara pada *average pooling*, nilai rata-rata dari setiap *window* dihitung untuk menghasilkan representasi yang lebih halus. Proses *pooling* tidak hanya mengurangi ukuran citra, tetapi juga membuat model menjadi lebih tahan terhadap variasi seperti rotasi atau translasi pada data input sehingga meningkatkan generalisasi model.

Fully connected layer adalah bagian akhir dalam arsitektur CNN, di mana semua neuron pada lapisan sebelumnya terhubung secara penuh dengan neuron pada lapisan ini. *Fully connected layer* bertugas mengintegrasikan fitur-fitur yang telah dipelajari dari *pooling* dan *convolution layer* untuk menentukan kelas target yang paling sesuai. Output dari *fully connected layer* biasanya berupa probabilitas untuk setiap kategori, yang dihitung menggunakan fungsi *softmax*. Probabilitas ini membantu menentukan prediksi akhir dari model CNN.

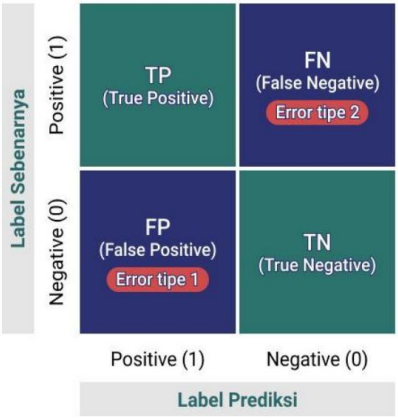


Gambar 2.7 *Fully Connected Layer* dengan 2 *Layer*

Softmax digunakan untuk kasus klasifikasi dengan lebih dari dua kelas (*multi-class*) dan ditempatkan pada lapisan output di tahap *fully connected* atau klasifikasi. *Softmax* berfungsi untuk menghitung probabilitas setiap kelas dari total seluruh kelas yang tersedia, sehingga dapat membantu menentukan kelas yang paling sesuai berdasarkan input yang diberikan. Kelebihan utama *softmax* adalah kemampuannya mengonversi output menjadi nilai probabilitas dalam rentang 0 hingga 1, dengan memastikan bahwa total probabilitas dari semua kelas selalu sama dengan 1. Hal ini memudahkan dalam memahami dan menginterpretasikan hasil klasifikasi (Raihan dkk., 2021).

2.2.5 *Confusion Matrix*

Menurut Raihan dkk (2021), *confusion matrix* adalah tabel berukuran N x N (N adalah jumlah kelas atau kategori) yang menunjukkan jumlah prediksi yang benar dan salah dari sebuah model klasifikasi. Matriks ini bertujuan untuk membandingkan nilai sebenar dengan nilai prediksi. Baris matriks menunjukkan kelas yang sebenarnya, sedangkan kolom menunjukkan kelas yang diprediksi. Nilai dalam *confusion matrix* dikelompokkan kedalam 4 kategori, seperti terlihat pada gambar 2.7.



Gambar 2.8 *Confusion Matrix*

1) *True Positive* (TP) : Data sebenar positif yang diprediksi positif.
2) *True Negative* (TN) : Data sebenar negatif yang diprediksi negatif.
3) *False Positive* (FP) : Data sebenar negatif yang diprediksi positif.
4) *False Negative* (FN) : Data sebenar negatif yang diprediksi negatif.

2.2.6 *Evaluation*

Hasil *confusion matrix* yang diperoleh akan dilakukan tahap *evaluation* untuk menganalisis kinerja atau performa dari model yang telah dikembangkan. Evaluasi dilakukan untuk memastikan model sesuai dengan tujuan dan kebutuhan spesifik yang ingin dicapai. Proses evaluasi dapat dilakukan dengan menentukan *accuracy*, *sensitivity*, *specificity*, AUC, dan *F1-score*.

1) *Accuracy* (Akurasi)

Akurasi menghitung ketepatan suatu model dalam melakukan klasifikasi secara keseluruhan.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

2) *Sensitivity* (Sensitifitas)

Sensitifitas menghitung jumlah kelas positif yang berhasil diprediksi oleh model.

$$Sensitivity = \frac{TP}{TP + FN} \quad (2.2)$$

3) *Specificity* (Spesifisitas)

Spesifisitas menghitung jumlah kelas negatif yang berhasil diprediksi oleh model.

$$Specificity = \frac{TN}{TN + FP} \quad (2.3)$$

4) *F1-Score*

F1-Score menggabungkan *precision* dan *recall*, yang menunjukkan seberapa baik model CNN dapat membedakan antara prediksi yang benar dan salah. Nilai *F1-Score* berkisar dari 0 hingga 1, dengan 1 berarti model memiliki *Precision* dan *Recall* yang sempurna.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.4)$$

5) *Area Under the Curve* (AUC)

AUC mengukur ketepatan model prediksi dengan menghitung nilai *sensitivity* dan *specificity*. Nilai AUC berkisar antara 0 dan 1, dimana nilai AUC yang lebih tinggi menunjukkan bahwa pengukuran klasifikasi yang diteliti lebih baik. Menurut Frastian (2018) nilai AUC dapat dibagi menjadi berbagai kriteria, seperti yang ditunjukkan dalam tabel 2.2.

Tabel 2.2 Klasifikasi Berdasarkan Nilai AUC

Nilai AUC	Keterangan
0.90 – 1.00	*Excellent*
0.80 – 0.90	*Good*
0.70 – 0.80	*Fair*
0.60 – 0.70	*Poor*
0.50 – 0.60	*Failure*

2.2.7 *Use Case Diagram*

Use case diagram adalah representasi grafis yang menggambarkan interaksi antara pengguna (aktor) dengan sistem dalam sebuah perangkat lunak. Diagram ini menunjukkan bagaimana pengguna dapat menggunakan sistem melalui berbagai skenario atau *use case*, yang mewakili fungsi-fungsi utama dari sistem. *Use case diagram* terdiri dari

aktor, yaitu pengguna yang berinteraksi dengan sistem, serta *use case* yang merepresentasikan aksi yang diberikan oleh sistem kepada aktor. Pengembang dapat memahami secara lebih jelas kebutuhan pengguna dan memastikan sistem yang dikembangkan sesuai dengan ekspektasi pengguna.

2.2.8 *Use Case Scenario*

Use case scenario adalah deskripsi rinci dari interaksi antara pengguna dengan sistem berdasarkan suatu skenario tertentu dalam *use case*. *Use case scenario* terdiri dari serangkaian langkah yang menjelaskan bagaimana sistem akan merespons setiap tindakan yang diambil oleh aktor. Skenario ini meliputi skenario utama yang menggambarkan jalur normal dari interaksi, serta skenario alternatif yang menggambarkan jalur lain jika terjadi situasi yang berbeda. *Use case scenario* membantu memperjelas alur kerja dan tanggapan sistem terhadap berbagai input, sehingga pengembang bisa merancang sistem dengan lebih tepat dan efisien.

2.2.9 *Black Box Testing*

Black box testing adalah jenis pengujian yang memfokuskan pada fungsionalitas dari perangkat lunak. Penguji tidak perlu memperhatikan struktur internal atau logika kode program. Sebaliknya, penguji menentukan sekumpulan kondisi input yang akan dicobakan dan melakukan pengujian terhadap program. Tujuan dari *black box testing* adalah untuk menemukan berbagai jenis kesalahan yang mungkin terjadi dan tidak sesuai dengan yang diharapkan (Pratama dkk., 2023). Terdapat beberapa teknik yang digunakan dalam *black box testing*, yaitu:

1) *Equivalence Partitioning*

Metode ini membagi domain input data menjadi beberapa kelas ekivalen yang diasumsikan akan diproses dengan cara yang sama oleh sistem. Dengan menguji satu nilai dari setiap kelas, penguji dapat mengurangi jumlah kasus uji yang diperlukan tanpa mengurangi cakupan pengujian.

2) *Boundary Value Analysis*

Metode ini fokus pada pengujian nilai-nilai batas dari kelas ekivalen, karena kesalahan sering terjadi pada batas-batas tersebut. Penguji akan menguji nilai tepat pada, di atas, dan di

bawah batas untuk memastikan sistem menangani batasan input dengan benar.

3) *Decision Table Testing*

Metode ini menggunakan tabel keputusan untuk menangkap berbagai kombinasi input dan kondisi yang mungkin terjadi, serta tindakan yang sesuai. Ini membantu memastikan bahwa semua kemungkinan kombinasi kondisi telah diuji.

4) *Use Case Testing*

Metode ini berfokus pada pengujian skenario penggunaan nyata yang menggambarkan interaksi antara pengguna dan sistem. Dalam metode ini, pengujian dilakukan berdasarkan *use case* atau skenario spesifik yang mendeskripsikan bagaimana pengguna seharusnya menggunakan fitur perangkat lunak untuk mencapai tujuan tertentu.

5) *State Transition Testing*

Metode ini digunakan ketika sistem memiliki status yang berbeda dan transisi antara status tersebut. Penguji akan memverifikasi bahwa transisi antara status terjadi dengan benar sesuai dengan spesifikasi.

2.2.10 *User Acceptance Testing* (UAT)

User acceptance testing merupakan tahap akhir dalam proses pengembangan produk, yang bertujuan memastikan sistem yang dibuat benar-benar sesuai dengan kebutuhan dan harapan pengguna akhir. Pada tahap ini, pengguna diberikan kesempatan untuk mencoba langsung perangkat lunak sebelum resmi diluncurkan. Dengan begitu, mereka dapat menemukan fitur yang mungkin belum sesuai atau adanya bug yang belum terdeteksi. UAT juga membantu mengidentifikasi masalah-masalah potensial yang mungkin terlewat pada proses pengujian sebelumnya.

Pelaksanaan UAT biasanya melibatkan berbagai pihak yang berkepentingan terhadap perangkat lunak, seperti *end user*, *business stakeholder*, tim QA, dan *developer*. Namun, peran utama dalam UAT ada pada *end user*, yaitu individu atau kelompok yang akan menggunakan system tersebut dalam aktivitas sehari-hari. Karena mereka memahami

kebutuhan dan ekspektasi terhadap aplikasi, *end user* menjadi pihak yang paling tepat untuk memastikan apakah perangkat lunak sudah benar-benar memenuhi kebutuhan mereka.