BAB II
TINJAUAN PUSTAKA
DAN DASAR TEORI

2.1 Studi Literatur

Kecerdasan buatan sedang mengalami kemajuan pesat, terutama dalam bidang *computer vision*. *Computer vision* bertujuan membuat mesin bisa melihat dunia seperti manusia, dengan tugas-tugas seperti deteksi, penandaan, dan pengenalan citra, serta analisis video dan pemrosesan bahasa alami (Bhatt, dkk, 2021). Beberapa penelitian terdahulu telah mengeksplorasi beragam teknik dalam mendeteksi objek, pengenalan wajah, klasifikasi, penandaan, dan analisis pada citra. Rincian perbandingan antara penelitian-penelitian terdahulu dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Penelitian terdahulu

No	Penulis	Metode	Tujuan	Hasil
1	(Wang, dkk, 2021)	SVM dan CNN	Membandingkan algoritm SVM dan CNN pada 4 jenis dataset yang berbeda	CNN mencapai akurasi 0,98 pada dataset yang besar, SVM hanya mencapai 0,88
2	(Reshi, dkk, 2021)	CNN, RF, GBM, SVC LR dan KNN	Mendiagnosis penyakit COVID-19 melalui analisis gambar X-ray dada	CNN mencapai akurasi 100% pada dataset uji dan 99,5% pada dataset validasi
3	(Piotrowski, Napiorkowski dan	DE dan PSO	Membandingkan teknik optimasi antara DE dan PSO	PSO kalah dari DE karena cenderung kurang stabil, mudah terjebak di solusi lokal,

Tabel 2. 1 Penelitian terdahulu (Lanjutan)

	Piotrowska, 2023)			dan sangat bergantung pada parameter serta struktur swarm. sedangkan, DE lebih adaptif, eksploratif, dan terbukti lebih unggul dalam banyak benchmark.
4	(Movassagh, dkk, 2023)	DE, GA, IWO	Membandingkan teknik optimasi dalam DE, GA dan IWO dalam mengoptimasi model DL	DE lebih baik dari GA dan IWO karena mampu menghindari solusi lokal, menjaga keseimbangan eksplorasi dan eksploitasi, memiliki struktur yang sederhana namun efektif, serta menunjukkan kinerja stabil dan adaptif di berbagai jenis permasalahan.
5	(Raiaan et al., 2024)	DE, GA, FA, GWO	Membandingkan teknik optimasi DE, GA dan IWO dalam mengoptimasi model CNN	Salah satu kelebihan utama DE adalah fleksibilitas dan kestabilannya dalam menghadapi berbagai jenis fungsi objektif, termasuk fungsi non-linear, non-*differentiable*, dan

Tabel 2. 1 Penelitian terdahulu (Lanjutan)

				multimodal yang sering muncul dalam pengolahan citra atau data medis.
6	(Huang, dkk, 2023)	DE-CNN	Merancang arsitektur CNN secara otomatis menggunakan DE	DE-CNN mencapai tingkat deteksi yang sangat tinggi dengan *Precision* 0,9995 dan *Recall* 0,9996 pada dataset SWaT serta *Precision* 0,9990 dan *Recall* 0,9950 pada dataset WADI
7	(Li, dkk, 2024)	DE-CNN	Mengoptimasi arsitektur CNN menggunakan DE dalam tugas pengenalan emosi musik	DE-CNN menunjukkan peningktan akurasi sebesar 4,83% pada dataset 1 dan 5,74% pada dataset 2 dibandingkan dengan CNN klasik.
8	(Kaliyapillai dan Krishnamurthy, 2020)	DE-RNN, DE-GRU dan DE-LSTM	Mengoptimasi penyetelan *hyperparameter* dari model *Deep Learning* untuk klasifikasi penyakit	DE dapat meningkatkan performa model *Deep Learning* dalam klasifikasi penyakit dengan akurasi rata-rata diatas 90%

Dalam penelitian-penelitian terdahulu telah banyak membahas tentang teknik-teknik yang dapat digunakan dalam mendeteksi objek, pengenalan wajah, klasifikasi, penandaan, dan analisis pada citra. *Convolutional Neural Network* (CNN) adalah metode yang paling banyak digunakan dalam melakukan klasifikasi citra. Sejak tahun 2012, CNN telah menjadi algoritma utama dalam tugas klasifikasi citra dan telah mencapai kinerja yang luar biasa pada tugas-tugas visual skala besar (Chen, dkk, 2021). Hal ini dipertegas dengan penelitian yang dilakukan (Wang, dkk, 2021) yang melakukan perbandingan antara *Support Vector Machine* (SVM) dan *Convolutional Neural Network* (CNN) dalam melakukan klasifikasi gambar. Studi yang dilakukan menemukan bahwa ketika menggunakan dataset besar seperti mnist, akurasi SVM adalah 0,88 sementara akurasi CNN adalah 0,98. Namun, ketika menggunakan dataset kecil seperti COREL1000, akurasi SVM adalah 0,86 dan akurasi CNN adalah 0,83, menunjukkan bahwa pembelajaran mesin tradisional seperti SVM memiliki efek solusi yang lebih baik pada dataset kecil, sedangkan kerangka kerja pembelajaran mendalam seperti CNN memiliki akurasi pengenalan yang lebih tinggi pada dataset besar.

Model *Deep Learning* (DL) seperti *Convolutional Neural Network* (CNN) unggul dibandingkan dengan beberapa model *Machine Learning* (ML) lainnya seperti *Random Forest* (RF), *Gradient Boosting Machine* (GBM), *Support Vector Classifier* (SVC), *Logistic Regression* (LR), dan *K-Nearest Neighbor* (KNN) dalam melakukan analisis gambar dan klasifikasi gambar. CNN berhasil mencapai akurasi 100% pada subset data uji dari dataset yang digunakan dalam penelitian ini, dengan presisi 1, menunjukkan performa yang sangat baik jika dibandingkan dengan model-model lainnya. Selain itu, ketika diuji menggunakan dataset validasi independen, CNN tetap menunjukkan tingkat akurasi yang tinggi, mencapai 99,5%, dan presisi 1. Hal ini menegaskan superioritas CNN atas model *Machine Learning* lainnya seperti RF, GBM, SVC, LR, dan KNN, terutama saat diuji dengan dataset validasi independent (Reshi, dkk, 2021).

Penelitian mengenai optimasi model CNN juga sudah banyak dilakukan. Hal ini bertujuan untuk meningkatkan lebih performa dari model CNN tersebut dalam melakukan tugasnya (Ghosh, dkk, 2023). Ada banyak teknik untuk

melakukan optimasi *parameter*, salah satunya adalah *Differential Evolution* (DE). DE merupakan salah satu algoritma optimasi berbasis populasi. Keunggulan DE terutama terletak pada kemampuannya menjaga keseimbangan eksplorasi dan eksploitasi solusi secara efisien, kestabilan hasil, kecepatan konvergensi, serta kesederhanaan implementasinya. Dalam studi perbandingan antara DE dan PSO, DE terbukti lebih unggul di sembilan kompetisi optimasi, dengan dominasi pada posisi empat besar dan kemenangan lebih dari dua kali lipat dibandingkan PSO. DE juga menunjukkan performa yang lebih konsisten, akurat, dan andal dalam berbagai masalah nyata, sementara PSO cenderung tidak stabil dan rentan terhadap konvergensi prematur (Piotrowski, Napiorkowski and Piotrowska, 2023). Dibandingkan GA dan IWO, DE lebih efektif dalam menghindari solusi lokal berkat mekanisme mutasi berbasis perbedaan antar individu dalam populasi, yang menghasilkan arah pencarian yang lebih terarah namun tetap fleksibel. GA dan IWO sering kali terjebak dalam solusi suboptimal, terutama pada fungsi multimodal, sedangkan DE dapat menjelajahi ruang solusi lebih luas secara adaptif dan stabil (Movassagh et al., 2023).

Selain itu, dalam konteks optimasi teknik dan rekayasa, DE menunjukkan kecepatan konvergensi yang tinggi dan konsistensi hasil yang sangat dibutuhkan dalam aplikasi dunia nyata. Keunggulan ini menjadi alasan utama DE lebih disukai dibanding WOA dan GA dalam menyelesaikan permasalahan seperti optimasi sistem tenaga Listrik (AL-Bahrani, AL-Kaabi and AL Hasheme, 2022). Dalam pengaturan *hyperparameter* model *deep learning* seperti CNN, DE juga menunjukkan performa yang lebih kompetitif dibandingkan algoritma seperti GA, FA, dan GWO. DE mampu menangani fungsi objektif yang kompleks, non-linear, dan multimodal dengan efisien, serta memberikan hasil yang lebih stabil dan cepat dalam mencapai solusi optimal. Sifat DE yang sederhana, fleksibel, dan hanya memerlukan sedikit parameter kontrol membuatnya unggul dari sisi kemudahan implementasi dan tuning (Raiaan et al., 2024). Bahkan dalam perbandingan dengan algoritma seperti CS, ABC, dan BSA, DE secara konsisten menunjukkan performa lebih baik dalam hal kecepatan konvergensi, efisiensi pencarian solusi, dan kestabilan hasil. Meskipun algoritma lain memiliki kelebihan tertentu, seperti

kemampuan eksplorasi CS atau kekuatan pencarian global ABC, DE tetap unggul karena keseimbangan optimal antara eksplorasi dan eksploitasi serta struktur algoritma yang sederhana namun efektif (Civicioglu, dkk, 2020).

Dalam penelitian yang dilakukan (Huang, dkk, 2023) mengembangkan sebuah metode desain arsitektur otomatis dengan algoritma *Differential Evolution* (DE) untuk CNN dalam melakukan deteksi instrusi dalam Sistem Kontrol Industri (ICS). DE-CNN bertujuan untuk mengoptimalkan arsitektur CNN dengan fokus pada peningkatan performa deteksi intrusi dalam hal *Precision*, *Recall*, dan *F1-Score*, serta mengurangi jumlah parameter model untuk mencapai model yang lebih ringan dan efisien. Ini ditujukan untuk mengatasi tantangan seperti keterbatasan sumber daya, waktu respons yang cepat, dan persyaratan privasi/keamanan data dalam konteks ICS. Hasil performa DE-CNN pada penelitian ini menunjukkan nilai *Precision* sebesar 0,9995, *Recall* sebesar 0,9996, dan *F1-Score* sebesar 0.9994 pada dataset SWaT. Untuk dataset WADI, DE-CNN mencapai *Precision* sebesar 0.9990, *Recall* sebesar 0,9950, dan *F1-Score* sebesar 0,9970. Performa ini menunjukkan bahwa DE-CNN berhasil mencapai tingkat deteksi intrusi yang sangat tinggi.

Penelitian yang dilakukan (Li, dkk, 2024) mengembangkan alur kerja algoritma *Differential Evolution-Convolutional Neural Network* (DE-CNN) yang mencakup inisialisasi populasi parameter, operasi *crossover* dan mutasi, pengelolaan *outlier* dan randomisasi bobot, serta evaluasi dan pemilihan struktur jaringan optimal berdasarkan kriteria terminasi yang ditentukan. Hasil dari penelitian ini menunjukkan bahwa pendekatan DE-CNN yang diperkenalkan berhasil mengoptimalkan parameter jaringan dalam tugas pengenalan emosi musik, mengungguli model-model CNN klasik dalam hal akurasi dan kecepatan konvergensi. DE-CNN membuktikan bahwa efektivitas algoritma DE dalam mengoptimasi arsitektur CNN memberikan adaptabilitas yang lebih tinggi terhadap keunikan dataset dan menghasilkan hasil klasifikasi yang lebih baik. Dengan penggunaan DE dalam optimasi arsitektur CNN menghasilkan peningkatan akurasi sebesar 4,83% pada Dataset1 dan 5,74% pada Dataset2 dibandingkan dengan CNN klasik.

DE juga dapat melakukan optimasi dalam penyetelan parameter dari model-model DL. Dalam penelitian yang dilakukan (Kaliyapillai and Krishnamurthy, 2020) mengembangkan model DL yang disetel *hyperparameter*nya menggunakan algoritma DE untuk diagnosis dan klasifikasi penyakit secara cerdas. Model yang diusulkan melibatkan proses *pre-processing*, ekstraksi fitur berbasis *Simulated Annealing* (SA), klasifikasi menggunakan Recurrent Neural Network (RNN), *Gated Recurrent Units* (GRU) dan *Long Short Term Memory* (LSTM), serta penyetelan parameter menggunakan DE. Hasil dari penelitian ini menunjukkan bahwa model DE-LSTM mencapai akurasi maksimum sebesar 97,59% pada dataset Diabetes, yang menunjukkan performa yang lebih baik dibandingkan dengan metode lain yang diuji dalam penelitian ini. Pada dataset EEG *Eye State*, model DE-LSTM juga menunjukkan hasil yang lebih baik dengan akurasi sebesar 88,52%, sedangkan model DE-LSTM mencapai akurasi sedikit lebih rendah yaitu 87,11%. Untuk dataset *Sleep Stage*, model DE-LSTM kembali menunjukkan performa superior dengan akurasi maksimum sebesar 93,18%. Selain itu, model DE-GRU juga menunjukkan hasil yang sangat baik dengan akurasi mendekati optimal sebesar 91,57% pada dataset *Sleep Stage*. Hasil ini menegaskan bahwa optimasi *hyperparameter* menggunakan algoritma DE dapat meningkatkan secara signifikan performa model DL dalam klasifikasi penyakit pada dataset medis.

Berdasarkan uraian penelitian terdahulu, dapat disimpulkan bahwa meskipun pendekatan optimasi menggunakan *Differential Evolution* (DE) telah banyak diterapkan dalam berbagai konteks, namun mayoritas penelitian tersebut hanya memanfaatkan DE untuk satu aspek saja, seperti perancangan arsitektur CNN yang dilakukan (Huang, dkk, 2023) atau penyetelan *hyperparameter* seperti yang dilakukan (Kaliyapillai dan Krishnamurthy, 2020). Bahkan dalam penelitian (Li, dkk, 2024) DE-CNN diterapkan pada domain non-visual seperti pengenalan emosi musik, bukan pada pengolahan citra buah secara spesifik. Kebaruan utama dalam penelitian ini terletak pada integrasi penuh *Differential Evolution* baik untuk merancang arsitektur CNN maupun melakukan optimasi *hyperparameter* secara otomatis, yang secara simultan diterapkan dalam sistem klasifikasi tingkat kematangan buah kelapa sawit. Selain itu, penelitian ini membandingkan secara

langsung performa DE-CNN dengan model *deep learning* terkenal lainnya seperti EfficientNetB3, ResNet50, dan DenseNet121, sehingga memberikan evaluasi yang lebih komprehensif terhadap efektivitas metode yang diusulkan. Penelitian ini juga membangun sistem klasifikasi berbasis web yang memungkinkan pengguna melakukan klasifikasi kematangan buah secara langsung melalui antarmuka aplikasi, menjadikan kontribusi penelitian ini tidak hanya teoretis, tetapi juga praktis. Dengan demikian, pendekatan yang diusulkan dalam penelitian ini menghadirkan kebaruan dalam hal metode optimasi yang komprehensif, penerapan pada domain klasifikasi citra pertanian yang spesifik, serta realisasi sistem nyata yang siap digunakan.

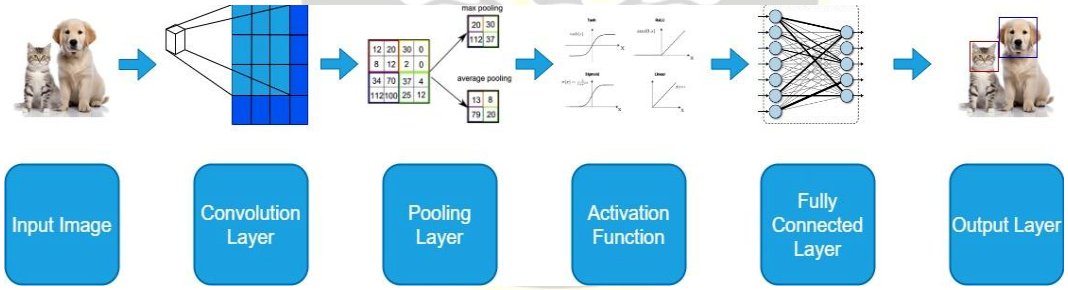
2.2 Dasar Teori

2.2.1 Deep Learning

Deep Learning (DL) merupakan subbidang dari *Machine Learning* (ML) yang menggunakan jaringan saraf tiruan (*Artificial Neural Networks*) untuk belajar mewakili data dalam beberapa tingkat abstraksi (Muchlinski, 2022). Secara lengkap, DL adalah suatu metode ML yang menggunakan jaringan saraf tiruan dengan banyak lapisan neuron yang saling terhubung seperti fungsi otak manusia. Tujuannya adalah untuk memproses dan menganalisis data yang besar dan kompleks dengan efisien. DL telah membuktikan kegunaannya dalam berbagai aplikasi, termasuk klasifikasi jenis buah, deteksi kebocoran pada sistem pendingin udara mobil, klasifikasi jenis kendaraan dalam sistem lalu lintas cerdas, pengenalan dan klasifikasi citra untuk sistem penagihan otomatis, serta analisis kematangan buah dalam pertanian. Dalam setiap aplikasi, DL menggunakan jaringan neuron yang kompleks untuk memproses data besar dan kompleks dengan tingkat akurasi yang tinggi (Kufel, dkk, 2023). Dalam melakukan klasifikasi gambar, DL mengaplikasikan berbagai metode seperti *Convolutional Neural Networks* (CNN), *Deep Belief Networks* (DBN), *Recurrent Neural Networks* (RNN) dan *Autoencoders*, *Generative Adversarial Networks* (GAN). CNN merupakan metode yang paling umum digunakan, dimana jaringan tersebut secara otomatis mempelajari fitur-fitur spasial dari gambar untuk tugas klasifikasi (Wang, dkk, 2021).

2.2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah jenis jaringan saraf tiruan yang secara khusus dirancang untuk mengenali pola visual dengan cara yang menyerupai mekanisme penglihatan manusia. CNN merupakan bagian penting dari bidang DL dan sering digunakan dalam aplikasi pengolahan gambar dan video, seperti klasifikasi gambar, deteksi objek, dan pengenalan wajah (Chen, dkk, 2021). Arsitektur CNN terinspirasi dari organisasi dan fungsi korteks visual manusia. Ini dirancang untuk menyerupai hubungan antara neuron dalam otak manusia (Bhatt, dkk, 2021). Struktur utama dari CNN terdiri dari lapisan konvolusi, lapisan penggabungan (pooling), lapisan aktivasi non-linear, dan lapisan terhubung penuh. Secara umum, gambar diproses sebelumnya dan kemudian dimasukkan ke dalam jaringan melalui lapisan masukan, diolah oleh beberapa lapisan konvolusi dan penggabungan yang disusun secara bergantian, dan kemudian diklasifikasikan oleh lapisan terhubung penuh (Chen, dkk, 2021) Ilustrasi dari struktur utama CNN terdapat pada Gambar 2.1.



Gambar 2. 1 Struktur utama CNN. Gambar dari (Bhatt, dkk, 2021)

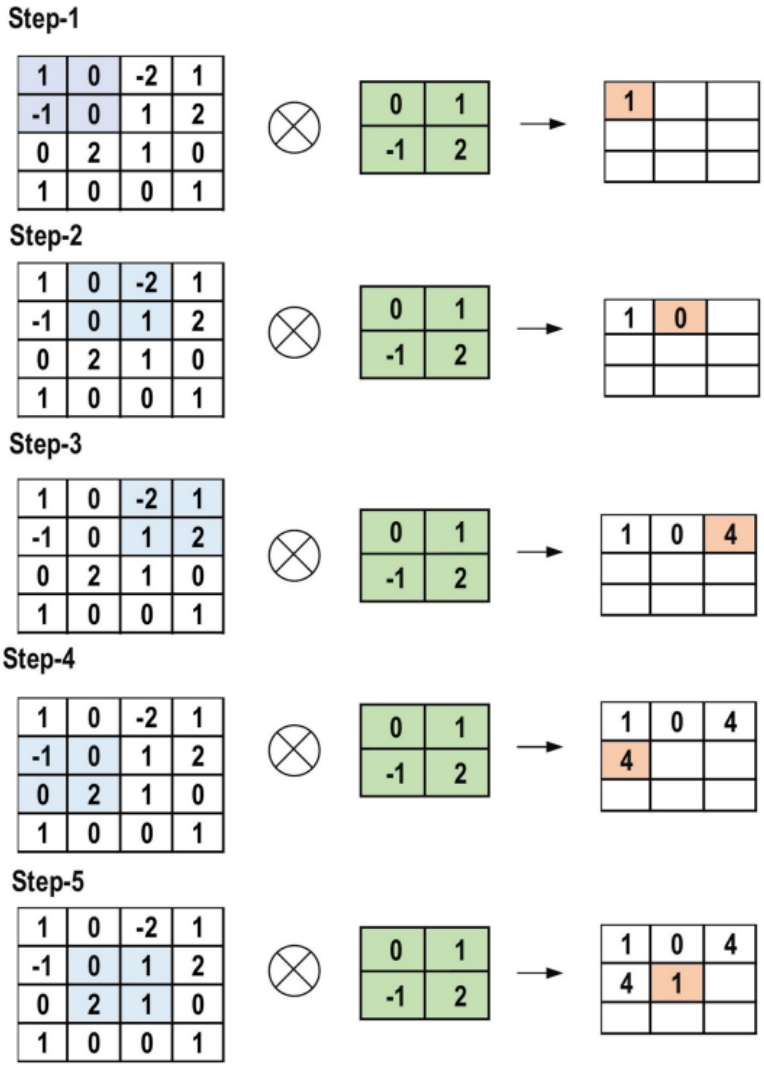
2.2.2.1 Input Image

Piksel merupakan blok bangunan gambar komputer, merepresentasikan data visual dalam bentuk biner. Dalam gambar digital, piksel disusun dalam matriks dengan nilai 0–255, menentukan kecerahan dan warna. Saat manusia melihat gambar, otak memproses informasi dengan cepat. Setiap *neuron* otak memiliki lapangan reseptif dan terhubung dengan *neuron* lain, menangkap informasi visual. CNN juga menggunakan konsep lapangan reseptif, menganalisis data secara

bertahap dari pola sederhana ke kompleks, memberikan kemampuan penglihatan kepada komputer.

2.2.2.2 Convolution Layer

Dalam arsitektur CNN, komponen yang paling signifikan adalah *Convolution Layer* atau layer konvolusi. Ini terdiri dari kumpulan filter konvolusi (yang disebut kernel). Gambar *input*, yang dinyatakan sebagai metrik N-dimensi, dikonvolusi dengan filter ini untuk menghasilkan peta fitur *output* (Alzubaidi, dkk, 2021). Lapisan konvolusi adalah lapisan yang sangat penting dalam arsitektur CNN. Ini mengambil gambar sebagai input dan menggunakan filter yang pada umumnya 2 × 2 atau 3 × 3. Layer konvolusi dalam arsitektur CNN adalah komponen kunci yang terdiri dari kumpulan filter konvolusi atau kernel. Kernel ini memiliki bobot yang diinisialisasi secara acak pada awal pelatihan CNN dan kemudian disesuaikan selama proses pelatihan untuk mengekstrak fitur yang signifikan dari gambar input. Operasi konvolusi melibatkan pergeseran kernel di atas gambar *input*, di mana nilai piksel yang sesuai dikalikan dan dijumlahkan untuk menghasilkan nilai skalar tunggal yang mewakili peta fitur *output*. Penambahan padding dan pengaturan langkah juga dapat mempengaruhi dimensi peta fitur *output*. Manfaat inti dari layer konvolusi termasuk konektivitas sparse, di mana hanya sedikit bobot yang diperlukan antara dua layer yang berdekatan, dan berbagi bobot, di mana satu grup bobot beroperasi pada seluruh input, mengurangi waktu pelatihan dan biaya. Untuk ilustrasi operasi konvolusi dapat dilihat pada Gambar 2.2.



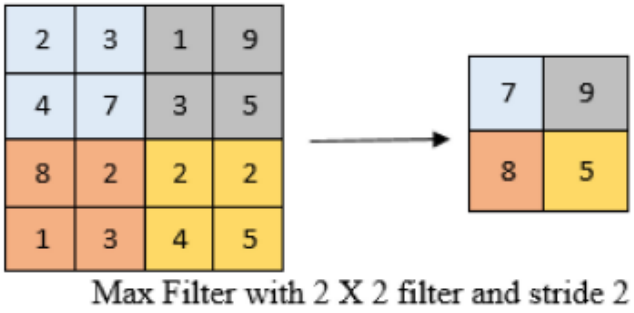
Gambar 2. 2 Perhitungan pada langkah *Convolution Layer*. Gambar dari (Alzubaidi, dkk, 2021)

2.2.2.3 Pooling Layer

Setelah mendapatkan peta fitur, diperlukan penambahan lapisan *pooling* (sub-sampling) dalam CNN, sebelah lapisan konvolusi. Tugas dari lapisan *pooling* adalah untuk mengecilkan ukuran spasial fitur yang dikonvolusi. Akibat dari pengurangan dimensionalitas ini, daya komputasi yang dibutuhkan untuk memproses data menjadi lebih rendah. Hal ini juga membantu dalam ekstraksi karakteristik utama yang invarian terhadap posisi dan rotasi, yang mempertahankan pelatihan model yang praktis. *Pooling* mengurangi waktu pelatihan sambil juga

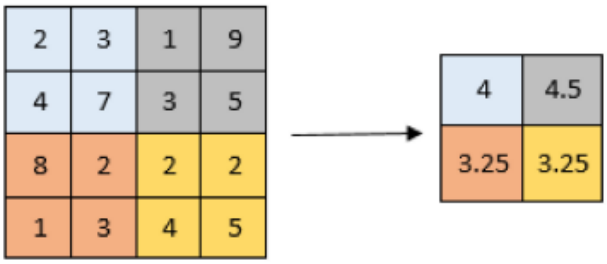
mencegah *overfitting*. Ada dua bentuk *pooling* yaitu *pooling* maksimum dan *pooling* rata-rata.

Dalam *pooling* maksimum tensor menjadi input untuk lapisan *pooling*. Kernel berukuran n × n (2 × 2 dalam contoh tersebut) digerakkan melintasi matriks dalam kasus *pooling* maksimum, seperti yang diilustrasikan dalam Gambar 2.3. dan nilai maksimum dipilih dan ditempatkan di lokasi yang sesuai dalam matriks *output*.



Gambar 2. 3 *Max Pooling*. Gambar dari (Bhatt, dkk, 2021)

Dalam *pooling* rata-rata, kernel berukuran n × n digerakkan melintasi matriks dalam pooling rata-rata, dan rata-rata dari semua nilai diperoleh untuk setiap titik dan ditempatkan di posisi yang sesuai dalam matriks output. Ini diulang untuk setiap saluran tensor input. Akibatnya, kita memiliki tensor *output*. Penting untuk diingat bahwa, sementara *pooling* mengurangi tinggi dan lebar gambar, jumlah saluran (kedalaman) tetap sama. Lapisan *pooling* menghitung statistik ringkasan dari output sekitarnya untuk menggantikan output jaringan pada titik-titik tertentu. Akibatnya, ini membantu dalam mengurangi dimensi spasial representasi, yang mengurangi jumlah komputasi dan bobot yang dibutuhkan. Prosedur *pooling* dilakukan secara independen pada setiap potongan representasi. Rata-rata dari tetangga persegi panjang, norma L2 dari tetangga persegi panjang, dan rata-rata tertimbang tergantung pada jarak dari piksel pusat adalah semua fungsi *pooling* seperti yang ditunjukkan dalam Gambar 2.4. Metode yang paling umum, bagaimanapun, adalah *pooling* maksimum, yang melaporkan *output* terbesar dari tetangga.



Gambar 2. 4 *Average Pooling*. Gambar dari (Bhatt, dkk, 2021)

2.2.2.4 Activation Function

Fungsi aktivasi adalah inti dari jaringan saraf yang memetakan *input* ke *output*. Mereka membuat keputusan apakah *neuron* harus aktif atau tidak berdasarkan *inputnya*. Lapisan aktivasi non-linear digunakan setelah lapisan-lapisan dengan bobot dalam CNN, memungkinkan jaringan untuk belajar pola yang kompleks. Fungsi aktivasi harus dapat diferensiasi, penting untuk melatih jaringan. Beberapa jenis fungsi aktivasi umum termasuk sigmoid, tanh, dan ReLU.

Sigmoid adalah fungsi aktivasi yang menerima bilangan real sebagai *input* dan menghasilkan *output* terbatas antara nol dan satu. Kurva sigmoid memiliki bentuk seperti huruf "S" dan dapat diwakili secara matematis pada persamaan 2.1. Tanh, mirip dengan sigmoid, juga menerima bilangan real sebagai *input*, namun *outputnya* terbatas antara -1 dan 1. Representasi matematisnya dapat dilihat pada persamaan 2.2. ReLU, singkatan dari *Rectified Linear Unit*, adalah fungsi aktivasi yang paling umum digunakan dalam konteks CNN. Ini mengubah seluruh nilai *input* menjadi bilangan positif. Salah satu keunggulan utama ReLU adalah beban komputasi yang lebih rendah dibandingkan dengan fungsi aktivasi lainnya. Representasi matematisnya dapat dilihat pada Persamaan 2.3.

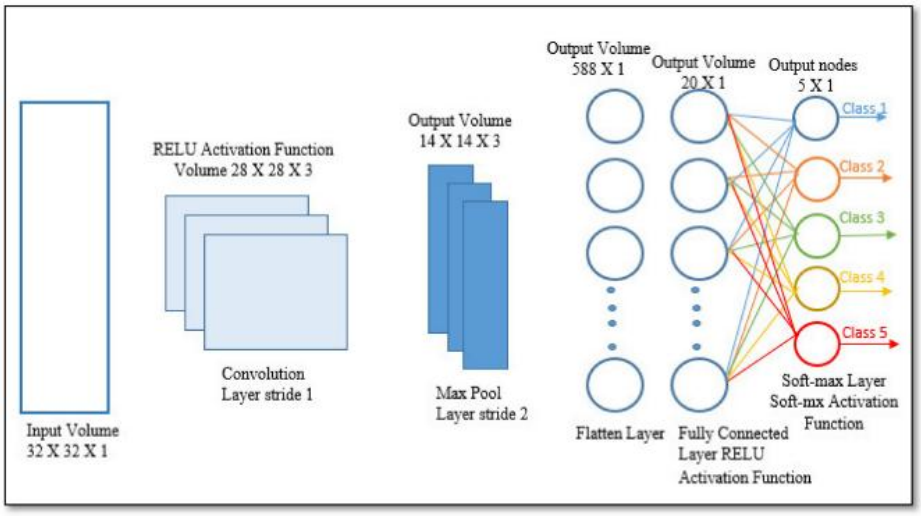
$$f(x)sigm = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$f(x)tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

$$f(x)ReLU = max\ (0, x) \tag{2.3}$$

2.2.2.5 Fully Connected Layer

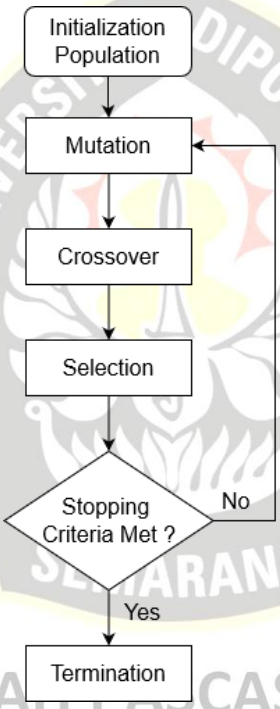
Fully Connected Layer adalah bagian dari arsitektur jaringan saraf yang menerima *input* dari *layer output pooling* atau konvolusional terakhir. Layer ini ditempatkan di bagian bawah jaringan dan bertindak seperti jaringan saraf maju (*feed-forward neural network*). Sebelum disampaikan sebagai *input*, *output* dari layer sebelumnya diluruskan menjadi sebuah vektor. Proses pelurutan ini, seperti yang terlihat pada bagian "*Flatten Layer*" di Gambar 2.5, mengubah matriks 3D hasil operasi pooling atau konvolusi menjadi vektor satu dimensi. Setelah proses pelurutan, semua nilai yang diperoleh dari output sebelumnya digabungkan menjadi satu vektor untuk dimasukkan ke dalam *Fully Connected Layer*. Peran dari *Fully Connected Layer* adalah mempelajari kombinasi nonlinier dari fitur-fitur tingkat tinggi yang direpresentasikan oleh *output* layer konvolusional. Ini berarti layer ini bertanggung jawab untuk memahami pola-pola yang lebih kompleks dari fitur-fitur yang telah diekstraksi. *Fully Connected Layer* menghubungkan semua neuron pada layer sebelumnya dengan neuron di layer ini. Proses ini dilanjutkan ke lapisan aktivasi (Softmax Layer), yang berfungsi untuk mengklasifikasikan *output* ke dalam berbagai kelas yang sudah ditentukan.



Gambar 2. 5 *Fully Connected Layer*. Gambar dari (Bhatt, dkk, 2021)

2.2.3 Differential Evolution

Differential Evolution (DE) adalah algoritma evolusioner yang digunakan untuk optimasi dalam ruang kontinu (Dhar, dkk, 2023). DE bekerja dengan cara menghasilkan solusi kandidat baru dengan memanfaatkan strategi perbandingan antar-individu dalam populasi (Noman and Iba, 2020). Algoritma DE melibatkan empat operasi utama: inisialisasi populasi, mutasi, *crossover* dan seleksi yang dapat dilihat pada Gambar 2.6. Algoritma ini dirancang untuk menemukan parameter berbobot optimal dalam berbagai aplikasi, termasuk dalam konteks pembelajaran mesin dan optimasi arsitektur jaringan saraf konvolusional CNN (Gonwirat and Surinta, 2021).



Gambar 2. 6 Alur Kerja *Differential Evolution*. Gambar dari (Bilal, dkk, 2020)

Algoritma dimulai dengan inisialisasi populasi, di mana serangkaian individu yang mewakili arsitektur CNN diperkenalkan (Ghosh, dkk, 2023). Setiap solusi ini mewakili titik awal dari pencarian solusi optimal dalam ruang solusi. Dengan memulai dari berbagai titik, *Differential Evolution* (DE) berupaya mengeksplorasi ruang solusi secara efisien untuk menemukan solusi yang optimal atau mendekati optimal (Li, dkk, 2024).

Mutasi merupakan variasi atau gangguan dengan elemen acak dalam konteks paradigma komputasi evolusioner. Operasi mutasi digunakan untuk menghasilkan solusi kandidat baru. DE menggunakan pendekatan diferensial, di mana solusi kandidat baru dihasilkan dengan menggabungkan perbedaan antara beberapa individu dalam populasi. Persamaan untuk operasi mutasi dapat dilihat pada Persamaan 2.4.

$$\mathrm{v}_i^g = \mathrm{x}_{best}^g + F \times (\mathrm{x}_{r1}^g - \mathrm{v}_{r2}^g) \tag{2.4}$$

Dalam persamaan 2.4, v_i^g menunjukkan vektor donor ke-i dari vektor target x_i pada generasi ke-g. x_{best}^g mendefinisikan individu terbaik dalam populasi berdasarkan nilai kebugaran pada generasi ke-g. Ini menunjukkan bahwa dalam operasi mutasi, menggunakan informasi dari individu terbaik dalam populasi untuk mempengaruhi solusi kandidat baru. x_{r1}^g dan x_{r2}^g adalah dua individu yang dipilih secara acak dan saling eksklusif dari populasi pada generasi ke-g. Artinya, kedua individu ini dipilih secara independen satu sama lain. Dan F adalah faktor skala yang mengatur seberapa besar perubahan yang akan diterapkan dalam operasi mutasi. Nilai F biasanya berada di dalam rentang (0,1) dan mempengaruhi seberapa besar dampak perubahan yang dihasilkan oleh operasi mutasi terhadap solusi kandidat baru (Ghosh, dkk, 2023).

Crossover merupakan salah satu operasi yang digunakan untuk menghasilkan solusi baru dalam proses evolusi populasi. Operasi ini mirip dengan proses perkawinan atau pertukaran informasi antara individu dalam populasi untuk menciptakan keturunan baru. *Crossover* dilakukan dengan membandingkan setidaknya satu solusi individu (vektor) dengan solusi individu lainnya dalam populasi. Kemudian, untuk setiap dimensi (atau elemen) dari vektor, sebuah bilangan acak dibangkitkan. Jika bilangan acak tersebut lebih kecil dari tingkat persilangan yang ditentukan sebelumnya, maka nilai dimensi tersebut diambil dari solusi individu pertama; jika tidak, maka nilai tersebut diambil dari solusi individu kedua. Proses ini menghasilkan solusi baru yang memiliki sifat-sifat dari kedua

solusi individu yang dipertukarkan. *Crossover* yang paling sering digunakan adalah persilangan binomial dimana y_i dihitung sebagai berikut :

$$y_{i,j} = \begin{cases} y_{i,j} \; if \; rand_{i,j} \leq CR \; or \; j = j_{rand} \\ x_{i,j} \qquad \text{otherwise} \end{cases} \text{for } j = 1, \dots, D \tag{2.5}$$

Dalam Persamaan 2.5, $rand_{i,j}$ merupakan bilangan acak riil dalam rentang [0, 1], yang digunakan untuk menentukan apakah akan menggunakan elemen dari vektor turunan atau vektor target dalam operasi *crossover*. Nilai $rand_{i,j}$ digunakan untuk membandingkannya dengan probabilitas *crossover*. j_{rand} Merupakan bilangan bulat acak dalam rentang {1, ..., D}, yang digunakan untuk menentukan indeks elemen yang akan diambil dari vektor target jika kondisi *crossover* tidak terpenuhi. Artinya, jika nilai randi,j lebih besar dari CR atau tidak memenuhi kondisi *crossover*, maka elemen ke-j dari vektor target akan digunakan dalam vektor turunan. CR Merupakan probabilitas *crossover* yang ditentukan sebelumnya, dengan nilai berada dalam rentang [0, 1]. Probabilitas ini menentukan seberapa sering operasi *crossover* akan diterapkan dalam pembentukan solusi turunan. Semakin tinggi nilai CR, semakin besar kemungkinan *crossover* terjadi. Setelah operasi *crossover* dilakukan, operator seleksi membandingkan setiap vektor turunan $y_i$dengan vektor target yang sesuai x_i. Dari kedua vektor tersebut, vektor yang memiliki kinerja atau nilai kebugaran yang lebih baik dipilih untuk menjadi bagian dari populasi pada generasi berikutnya (Baioletti, dkk, 2020).

Setelah seleksi, algoritma memeriksa stopping criteria atau kriteria penghentian yang telah ditetapkan di awal proses. Kriteria ini bisa berupa jumlah generasi maksimum, nilai fungsi objektif yang diinginkan, waktu komputasi, atau tingkat konvergensi populasi. Jika salah satu dari kriteria penghentian terpenuhi, algoritma akan berhenti; jika tidak, siklus mutasi, *crossover*, dan seleksi diulang untuk generasi berikutnya. Kriteria penghentian ini penting untuk mengontrol dan membatasi proses optimasi, memastikan bahwa algoritma tidak berjalan tanpa batas dan menggunakan sumber daya komputasi secara efisien. Setelah itu, pada tahap terminasi, algoritma berhenti, dan solusi terbaik yang ditemukan selama iterasi diambil sebagai hasil akhir. Pada tahap ini, hasil optimasi dilaporkan, dan solusi

yang ditemukan dapat disimpan untuk penggunaan lebih lanjut atau diterapkan dalam konteks yang membutuhkan solusi optimal. Terminasi memastikan bahwa proses optimasi telah mencapai tujuannya dan memberikan hasil yang dapat digunakan dalam aplikasi praktis (Bilal, dkk, 2020).

2.2.4 Evaluasi Model

Untuk menentukan kualitas dan keandalan sebuah model klasifikasi, diperlukan proses evaluasi yang sistematis dan terukur. Evaluasi ini bertujuan untuk mengukur seberapa baik performa model dalam membuat prediksi yang benar pada data yang belum pernah dilihat sebelumnya. Alat fundamental yang digunakan dalam evaluasi klasifikasi adalah *Confusion Matrix* (Pham, 2021). *Confusion matrix* adalah sebuah tabel yang memvisualisasikan performa sebuah algoritma klasifikasi. Dalam konteks klasifikasi multi-kelas, matriks ini berbentuk N×N, di mana N adalah jumlah kelas. Baris pada matriks merepresentasikan kelas aktual (kebenaran dasar atau *ground truth*), sedangkan kolom merepresentasikan kelas yang diprediksi oleh model (Heydarian dan Doyle, 2022). Ilustrasi rancangan *confusion matrix* untuk penelitian ini, yang memiliki tiga kelas (Mentah, Matang, Busuk), dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Rancangan *Confussion Matriks*

Matriks		Prediksi Kelas		
		Mentah	Matang	Busuk
Kelas Sebenarnya	Mentah			
	Matang			
	Busuk			

Dari *confusion matrix*, kita dapat menurunkan berbagai metrik performa. Namun, pertama-tama kita perlu memahami konsep dasar *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN) dalam skenario multi-kelas. Konsep-konsep ini dihitung untuk setiap kelas secara individual dengan pendekatan *one-vs-rest* (Heydarian dan Doyle, 2022). *True Positive* (TP) adalah jumlah prediksi yang benar untuk kelas yang sedang dievaluasi. *False Positive* (FP) adalah jumlah prediksi

yang salah, di mana sampel dari kelas lain diprediksi sebagai kelas yang sedang dievaluasi. *False Negative* (FN) adalah jumlah prediksi yang salah, di mana sampel dari kelas yang sedang dievaluasi diprediksi sebagai kelas lain. *True Negative* (TN) jumlah prediksi yang benar untuk semua kelas lain (bukan kelas yang sedang dievaluasi). Berdasarkan komponen-komponen tersebut, metrik performa untuk setiap kelas dapat dihitung dan untuk formulasi dari akurasi dapat dilihat pada Persamaan 2.6.

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

Dimana TP sebagai *True Postive*, mewakili hasil prediksi yang benar positif, dimana keduanya kelas actual dan kelas yang diprediksi adalah positif. TN sebagai *True Negative*, mewakili hasil prediksi yang benar negatif, di mana model memprediksi nilai yang sesuai dengan kelas aktual yang negative. FP sebagai *False Positive,* mewakili hasil prediksi yang salah positif, di mana model memprediksi nilai positif tetapi kelas aktualnya negative. FN sebagai *False Negative*, mewakili hasil prediksi yang salah negatif, di mana model memprediksi nilai negatif tetapi kelas aktualnya positif.

Presisi bertujuan untuk mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan total prediksi positif yang dilakukan oleh model. Ini memberikan indikasi seberapa ”bersih” atau seberapa akurat model dalam mengklasifikasikan sampel positif. Formulasi dari presisi dapat dilihat pada Persamaan 2.7.

$$Precision = \frac{TP}{TP+FP} \tag{2.7}$$

Recall betujuan untuk mengukur seberapa banyak dari seluruh sampel positif yang berhasil diklasifikasikan dengan benar oleh model. Ini mengukur ”kepekaan” dari model terhadap sampel-sampel yang sebenarnya positif. Formulasi dari recall dapat dilihat pada Pesamaan 2.8.

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

Dan *F1-Score* adalah rata-rata harmonik dari presisi dan *recall*. Ini memberikan keseimbangan antara kedua metrik tersebut. *F1-Score* berguna ketika keseimbangan antara presisi dan *recall* diinginkan. Formulasi dari *F1-Score* dapat dilihat pada Persamaan 2.9.

$$F1\text{-}Score = 2.\frac{precision\,.\,recall}{precision + recall} \quad (2.9)$$

Setelah metrik evaluasi seperti presisi, recall, dan F1-score dihitung untuk setiap kelas, langkah selanjutnya adalah mengagregasi hasil tersebut untuk mendapatkan gambaran performa model secara komprehensif. Untuk tujuan ini, akurasi keseluruhan (*overall accuracy*) menjadi tolok ukur utama yang merepresentasikan seberapa baik model bekerja secara total. Metrik ini sangat penting karena menyajikan satu skor tunggal yang memudahkan perbandingan antar model. Formulasi untuk menghitung akurasi keseluruhan dapat dilihat pada Persamaan 2.10

$$Accuracy_{Keseluruhan} = \frac{Jumlah\ Semua\ Prediksi\ Benar}{Total\ Seluruh\ Sampel} \quad (2.10)$$

Selain akurasi keseluruhan, performa model klasifikasi secara umum juga dapat diukur menggunakan metrik agregat untuk *precision*, *recall*, dan *F1-score*. Dalam konteks klasifikasi multikelas, terdapat dua pendekatan umum yang digunakan, yaitu *macro average* dan *weighted average*. *Macro average* menghitung rata-rata dari nilai *precision*, *recall*, dan *F1-score* untuk masing-masing kelas tanpa mempertimbangkan jumlah sampel di setiap kelas. Dengan demikian, pendekatan ini menilai performa model secara seimbang terhadap setiap kelas, meskipun distribusi data antar kelas tidak merata. Formulasinya dapat dilihat pada persamaan 2.11 sampai 2.13:

$$Macro\ Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP+FP} \quad (2.11)$$

$$Macro\ Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP+FN} \quad (2.12)$$

$$Macro\ F1-Score = \frac{1}{n}\sum_{i=1}^{n}2.\frac{precision\ .\ recall}{precision + recall} \quad (2.13)$$

Sementara itu, *weighted average* menghitung rata-rata dari setiap metrik dengan mempertimbangkan proporsi jumlah data (*support*) di tiap kelas. Pendekatan ini memberikan gambaran kinerja model yang lebih realistis terhadap distribusi aktual data. Formulasinya dapat dilihat pada persamaan 2.14 sampai 2.16:

$$Weighted\ Precision = \frac{\sum_{i=1}^{n} Precision.Support}{\sum_{i=1}^{n} Support} \quad (2.14)$$

$$Weighted\ Recall = \frac{\sum_{i=1}^{n} Recall.Support}{\sum_{i=1}^{n} Support} \quad (2.15)$$

$$Weighted\ F1-Score = \sum_{i=1}^{n}2.\frac{precision\ .\ recall}{precision + recall} \quad (2.16)$$

Dengan menggunakan *macro* dan *weighted average*, dapat memperoleh gambaran performa model secara keseluruhan yang lebih komprehensif, baik dari sisi keadilan antar kelas (*macro*) maupun proporsionalitas terhadap distribusi data (*weighted*). Dalam penelitian ini, nilai *precision*, *recall*, dan *F1-score* keseluruhan dihitung menggunakan metode *weighted average* karena mempertimbangkan jumlah sampel dari masing-masing kelas, sehingga hasil evaluasi menjadi lebih representatif terhadap dataset yang digunakan.