BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian yang dilakukan oleh Bala dkk. (2023) dijelaskan bahwa dengan kemampuannya untuk mengadopsi pembelajaran mandiri, metode *Convolutional Neural Network (CNN)* telah menjadi salah satu pendekatan pembelajaran mendalam yang paling diminati untuk klasifikasi dalam bidang pencitraan medis. Penemuan ini menggambarkan signifikansi CNN dalam menghadirkan solusi yang efektif dan berdaya dalam mengolah data citra medis untuk tujuan klasifikasi.

Beberapa penelitian terkait pembeda penyakit kulit disajikan pada Tabel 3.1, seperti pada penelitian oleh Ahsan dkk. (2022) yang mengembangkan sistem deteksi *Monkeypox* menggunakan model *Inception V3* berbasis *DCNN (Deep Convolutional Neural Network).* Pengolahan data diawali dengan *Data Augmentation* dengan Keras *Image Processing Library.* Dalam percobaan pendahuluan yang digunakan adalah model VGG16 yang dimodifikasi. Pada model inti terdiri dari 3 elemen penting yaitu *Pre-Trained Architecture* (Untuk identifikasi fitur dimensi tinggi), *Updated Layer*, *Prediction Class*. Setelah lapisan input awal (Gambar ukuran 224x224), 2 lapisan konvolusi (filter 3x3) ditambahkan. Diikuti dengan lapisan *Max Pooling*, diikuti dua lapisan *convolutional* dan lapisan *Max Pooling* mencapai bagian lapisan yang dimodifikasi meratakan arsitektur *(Flatten),* diikuti tiga lapisan *Dense* dan 1 lapisan *Dropou*t. Penggunaan Local Interpretable Model Agnostic Explanations (LIME) dalam penelitian ini dapat membantu analisis *true prediction model* dan menawarkan kesempatan untuk memahami Blackbox di balik prediksi akhir model CNN. Temuan pada penelitian ini menunjukkan bahwa dengan menggunakan pendekatan modifikasi VGG16 yang diusulkan dapat membedakan pasien dengan gejala *Monkeypox* dari pasien lain di Studi Satu dan Dua dengan akurasi berkisar antara 78% hingga 97%. Penelitian lain dilakukan oleh Ali dkk. (2022) untuk mendeteksi lesi kulit *Monkeypox*, *Chickenpox,* dan *Measles*.

Dengan mengolah Dataset menggunakan arsitektur CNN VGG16, ResNet50, Inception V3, dan Ensemble, menunjukkan. Bahwa penelitian ini menghasilkan akurasi terbaik (82,96±4,57%) dan VGG16 menunjukkan kinerja kompetitif (81,48±6,87%).

Tabel 1.1 Daftar Penelitian Terkait

No	Penulis	Judul	Metode	Hasil
1	Ahsan dkk. (2022)	Image Data collection and implementation of deep learning-based model in detecting Monkeypox disease using modified VGG16.	Model Inception V3 berbasis DCNN (Deep Convolutional Neural Network)	Temuan pada penelitian ini menunjukkan bahwa dengan menggunakan pendekatan modifikasi VGG16 yang diusulkan dapat membedakan pasien dengan gejala Monkeypox dari pasien lain di Studi Satu dan Dua dengan akurasi berkisar antara 78% hingga 97%.
2	Ali dkk. (2022)	Monkeypox Skin Lesion Detection Using Deep Learning Models: A Feasibility Study	Arsitektur CNN VGG16, ResNet50, Inception V3, dan *Ensemble*	Dengan mengolah Dataset menggunakan arsitektur CNN VGG16, ResNet50, Inception V3, dan Ensemble, menunjukkan. Bahwa penelitian ini menghasilkan akurasi terbaik (82,96±4,57%) dan VGG16 menunjukkan kinerja kompetitif (81,48±6,87%).

Tabel 1.1 Daftar Penelitian Terkait (lanjutan)

No	Penulis	Judul	Metode	Hasil
3	Bala dkk. (2023)	MonkeyNet: A robust deep convolutional neural network for monkeypox disease detection and classification	Arsitektur CNN VGG16, ResNet50, MobileNetV1, Inception V3, Xception, DenseNet-201	Penelitian ini menghasilkan tingkat akurasi tertinggi klasifikasi multiclass dataset original 93,91% dan 98,91% dataset augmentasi dalam deteksi empat kelas klasifikasi yaitu kulit normal dan kulit dengan penyakit (campak, cacar air, dan cacar monyet) dengan model baru yang dikembangkan menggunakan base model DenseNet-201.
4	Jaradat dkk. (2023)	Automated Monkeypox Skin Lesion Detection Using Deep Learning and Transfer Learning Techniques	Arsitektur CNN ResNet50, VGG16, MobileNetV2, VGG19, dan EfficientNetB3	Hasilnya menunjukkan bahwa arsitektur VGG16 mencapai akurasi pelatihan 0,999 pada epoch 9, dan akurasi berturut-turut untuk EfficientNetB3, VGG19, MobileNetV2, dan ResNet50 pada epochs 13, 12, 14, dan 14 adalah 65,2%, 98,4%, 99,8%, dan 91,8%. Model MobileNetV2 terpilih sebagai yang terbaik dengan akurasi 99%, mengungguli model-model lainnya.

Tabel 1.1 Daftar Penelitian Terkait (lanjutan)

No	Penulis	Judul	Metode	Hasil
5	Nayak dkk. (2023)	Deep learning based detection of monkeypox virus using skin lesion images	Arsitektur CNN GoogLeNet, Places365-GoogleNet, SqueezeNet, ResNet, dan AlexNet	Hasil eksperimen menunjukkan bahwa GoogLeNet dan Places365-GoogLeNet memberikan dasar yang kokoh bagi perangkat lunak berbasis deep learning dalam mendiagnosis Monkeypox secara efisien, dengan tingkat akurasi masing-masing sekitar 97,86% dan 97,61%. Tingkat akurasi tertinggi, mencapai 99%, dicapai oleh model MobileNetV2.
6	A. Nugroho & Suhartanto (2020)		*DenseNet-BC*	*Random Search* digunakan untuk menyeleksi kandidat *hyperparameter learning rate* dan *batch size* guna mengoptimalkan model *DenseNet* dalam klasifikasi gambar menggunakan dataset *CIFAR-10* dan *CIFAR-100.* Hasil eksperimen menunjukkan bahwa *Random Search* berhasil menghasilkan model dengan rata-rata akurasi 95%.

Tabel 1.1 Daftar Penelitian Terkait (lanjutan)

No	Penulis	Judul	Metode	Hasil
7	Lestari dkk. (2025).		*DenseNet-PSO, DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-101, Inception V3, ResNet-101, MobileNet*	*PSO* diterapkan pada arsitektur *DenseNet* untuk meng optimalkan pemilihan *hyperparameter* dalam klasifikasi penyakit daun tomat. Hasil eksperimen menunjukkan bahwa *DenseNe*t yang dioptimasi dengan *PSO (DenseNet-PSO)* mencapai akurasi tertinggi sebesar 97.39%, dengan nilai *macro-precision* 97.47%, *macro-recall* 97.39%, dan *macro-F1-score* 97.38%.

Pada penelitian yang dilakukan oleh Bala dkk. (2023) bertujuan untuk mengatasi tantangan identifikasi *Monkeypox* dari gambar medis dengan salah satu teknologi Deep Learning khususnya CNN. Peneliti mengembangkan dataset baru yaitu *MSID (Monkeypox Skin Image Dataset)* yang tersedia dalam *Mendeley Data* dan Metode bernama Monkey-Net dengan *base model* DenseNet-201. Selain itu, pada penelitian ini juga membandingkan beberapa model dengan arsitektur yang berbeda diimplementasi pada studi kasus yaitu VGG16, ResNet50, MobileNetV1, Inception V3, Xception. Penelitian ini menghasilkan tingkat akurasi tertinggi klasifikasi multiclass dataset original 93,91% dan 98,91% dataset augmentasi dalam deteksi empat kelas klasifikasi yaitu kulit normal dan kulit dengan penyakit (campak, cacar air, dan cacar monyet) dengan model baru yang dikembangkan menggunakan *base model DenseNet-201*. Pada penelitian ini juga disebutkan bahwa DenseNet-201 merupakan model yang membantu mengurangi masalah hilangnya gradien sekaligus meningkatkan penggunaan kembali fitur dan mengurangi konsumsi parameter.

Pada penelitian yang telah dilakukan oleh Nayak dkk. (2023), digunakan empat model deep learning, yaitu GoogLeNet, Places365-GoogleNet, SqueezeNet, ResNet, dan AlexNet, untuk mendiagnosis penyakit Monkeypox dari gambar lesi kulit. Hasil eksperimen menunjukkan bahwa GoogLeNet dan Places365-GoogLeNet memberikan dasar yang kokoh bagi perangkat lunak berbasis deep learning dalam mendiagnosis Monkeypox secara efisien, dengan tingkat akurasi masing-masing sekitar 97,86% dan 97,61%. Kedua model tersebut memiliki bobot ringan dan efisiensi sumber daya, menjadikannya pilihan ideal untuk aplikasi di berbagai fasilitas kesehatan, bahkan di daerah pedesaan dengan keterbatasan uji PCR. ResNet-18 juga terbukti efektif dalam kondisi perangkat keras terbatas, sedangkan SqueezeNet dan AlexNet dapat menjadi dasar yang solid untuk model CNN pada perangkat dengan kinerja terbatas. Tingkat akurasi tertinggi, mencapai 99%, dicapai oleh model MobileNetV2. Penelitian ini juga menggambarkan pendekatan yang cermat dalam pengembangan model, termasuk regulasi learning rate, fine-tuning, dan penyesuaian hyperparameter untuk menghindari overfitting. Secara keseluruhan, model yang diusulkan menunjukkan potensi sebagai alat diagnosis yang efektif dan andal untuk Monkeypox, dengan aplikasi potensial dalam deteksi waktu nyata *(real-time)* melalui ponsel cerdas (*smartphone)*.

Penelian lain juga telah dilakukan yaitu oleh Jaradat dkk. (2023). Penelitian ini bertujuan meningkatkan metode berbasis deep learning untuk deteksi monkeypox (mpox) dengan memanfaatkan model MobileNetV2. Studi ini melibatkan lima tahapan utama, termasuk pengumpulan data, preprocessing data, pelatihan model, dan evaluasi model. Menggunakan 117 gambar pasien, dengan 45 gambar mpox dan 74 gambar penyakit lain, penelitian membandingkan performa lima model (ResNet50, VGG16, MobileNetV2, VGG19, dan EfficientNetB3) untuk meningkatkan akurasi deteksi virus mpox. Dalam penelitian ini, model dievaluasi menggunakan beberapa metrik, termasuk optimizer, loss function, dan scoring metrics. Hasilnya menunjukkan bahwa arsitektur VGG16 mencapai akurasi pelatihan 0,999 pada epoch 9, dan akurasi berturut-turut untuk EfficientNetB3, VGG19, MobileNetV2, dan ResNet50 pada epochs 13, 12, 14, dan 14 adalah 65,2%, 98,4%, 99,8%, dan 91,8%. Model MobileNetV2 terpilih sebagai yang

terbaik dengan akurasi 99%, mengungguli model-model lainnya. Hasil penelitian menunjukkan potensi aplikasi model ini dalam deteksi real-time dan prediksi kasus mpox melalui *smartphone*.

Penelitian yang dilakukan oleh A. Nugroho & Suhartanto (2020) mengusulkan penggunaan metode *batching* dengan ukuran *batch* yang adaptif, yang disesuaikan dengan rasio *learning rate* selama pelatihan. Metode ini bertujuan untuk mempercepat waktu pelatihan tanpa mengurangi akurasi. Penelitian ini melakukan penyesuaian terhadap dua *hyperparameter* penting, yaitu *learning rate* dan *batch size,* dengan menggunakan metode *Random Search* untuk mencari kombinasi yang optimal. Hasil eksperimen menunjukkan bahwa *batch size* dengan batas bawah 64 dan *learning rate* optimal dalam kisaran 0,1 - 0,3 menghasilkan akurasi rata-rata 95%. Percobaan dilakukan pada GPU RTX 2080 Ti menggunakan dataset CIFAR-10 dan CIFAR-100.

Penelitian lain juga dilakukan oleh Lestari dkk. (2025). Penelitian tersebut mengeksplorasi penggunaan *DenseNet* yang dioptimalkan dengan algoritma *PSO* untuk deteksi penyakit pada daun tomat. *DenseNet*, yang terkenal dengan konektivitas antar lapisannya, diterapkan untuk mengidentifikasi penyakit pada daun tomat dengan lebih efektif. Penelitian ini menyarankan penggunaan *PSO* untuk menyempurnakan hyperparameter *DenseNet*, termasuk jumlah lapisan dalam *blok dense*, tingkat pertumbuhan, tingkat *dropout,* fungsi aktivasi, dan *optimizers.* Hasil eksperimen menunjukkan bahwa model *DenseNet-PSO* mampu mencapai akurasi rata-rata 97,39%, dengan *precision* makro tertinggi (97,47%), *recall* makro (97,39%), dan skor F1 makro (97,38%), serta mengungguli enam model arsitektur lainnya dalam hal parameter total, efisiensi komputasi, dan kinerja keseluruhan.

2.2 Dasar Teori

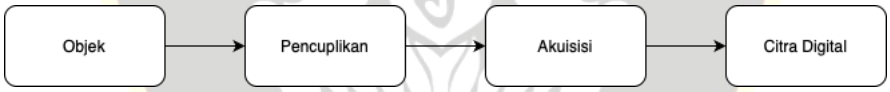
2.2.1 Pengolahan Citra Digital

Citra sendiri adalah representasi objek dunia nyata pada media dua dimensi. Pada Gambar 2.1, seperti yang diketahui bahwasannya citra pada gambar merupakan objek tiga dimensi di dunia nyata, tetapi saat ditangkap oleh kamera, objek tersebut diubah atau direpresentasikan menjadi data atau file berbentuk dua

dimensi. Citra tersusun atas sekumpulan piksel (*Picture Element*) yang memiliki koordinat (x,y).

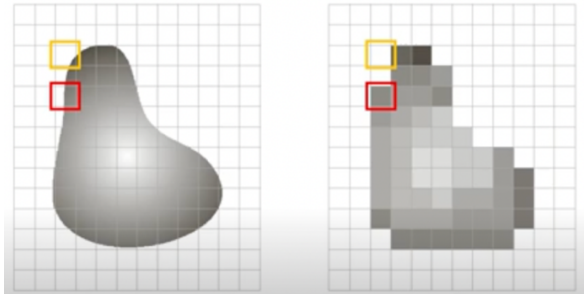
Gambar 2.1 Citra Penyakit Kulit Campak
(Sumber : Bala dkk., 2023)

Citra digital adalah citra atau gambar yang ditangkap melalui perangkat digital seperti kamera digital, pemindai, USG, CT- Scan dengan bantuan X-Ray untuk melihat organ dalam tubuh manusia, dan lain-lain. Proses yang dilakukan untuk menghasilkan citra digital disebut sebagai akuisisi citra. Proses akuisisi citra memiliki beberapa tahapan seperti pada Gambar 2.2.



Gambar 2.2 Proses Akuisisi Citra

Pada Gambar 2.3 dapat dijelaskan bahwa objek di dunia nyata ditangkap oleh kamera. Selanjutnya kamera akan melakukan proses pencuplikan dan juga kuantisasi yang akan dihasilkan Citra Digital. Jadi pencuplikan dan kuantisasi adalah objek atau proses utama dalam menghasilkan citra digital melalui perangkat digital. Pencuplikan dilakukan dengan membuat grid yang mana kotak-kotak tersebut diberi nama piksel dengan ukuran tertentu.

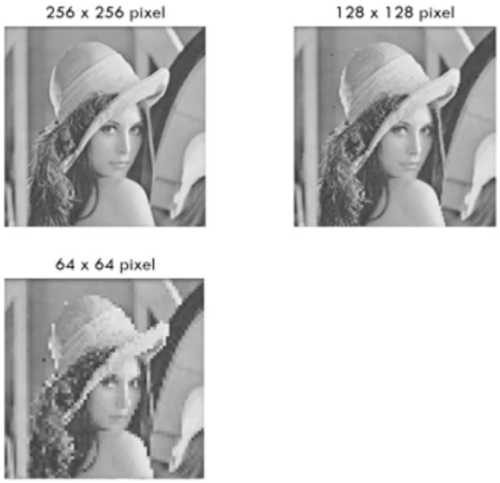


Gambar 2.3 Pencuplikan
(Sumber : Nugroho dkk., 2019)

Pada Gambar 2.3 sebagai contoh memiliki ukuran 14 baris dan 12 kolom. Kemudian pencuplikan akan dilakukan kuantisasi menjadi gambar baru yang sebelah kanan (Citra Digital). Pada gambar kiri, objek dunia nyata terlihat mulus (citra kontinu) tidak ada patahan seperti gambar sebelah kanan. Kuantisasi dilakukan dengan melihat piksel sebagai kotak-kotak. Jika kotak tersebut mengandung bagian dominan dari suatu objek maka dijadikan satu warna baru dalam representasi digital. Seperti pada kotak merah menandakan area piksel tersebut mengandung bagian yang cukup dominan sehingga pada representasi dua dimensinya diubah menjadi satu warna sendiri sedangkan pada kotak kuning menandakan area tersebut hanya mengandung sedikit informasi dari bagian objek nyata yang ditangkap maka dari itu, bagian piksel tersebut kosong atau tidak menjadi bagian dari citra digital yang dihasilkan. Jadi gambar sebelah kanan merupakan citra digital yang dihasilkan lewat proses pencuplikan dan kuantisasi.

Dari hasil akuisisi citra yang dijelaskan sudah mendapatkan sebuah citra digital. Semakin padat piksel tersebut maka semakin tajam gambar yang dilihat. Pada citra digital, objek dua dimensi mempunyai ukuran panjang dan lebar yang disebut sebagai baris dan kolom.

Jika hasil akuisisi citra melakukan pencuplikan dengan ukuran 256 x 256 piksel maka artinya terdapat 256 baris dan 256 kolom seperti Gambar 2.4.



Gambar 2.4 Hasil Akuisisi Citra
(Sumber : Pazos & Bhaya, 2014)

Pada Gambar 2.4, jika dilakukan pencuplikan dengan ukuran pencuplikan menjadi lebih kecil (gambar berukuran 128 x 128 pixel dan 64 x 64 pixel) maka gambar yang dihasilkan sedikit tidak setajam gambar pertama (gambar 256 x 256 pixel) dimana terdapat patahan setiap sudut. Komputer melihat struktur cita digital sebagai sebuah matriks yang memiliki baris dan kolom. Nilai pada setiap baris dan kolom mewakilkan warna yang dihasilkan oleh gambar tersebut. Jika nilai semakin kecil, maka menandakan warna dalam citra semakin besar dan sebaliknya jika nilai terebut semakin besar, maka warna dalam citra semakin terang karena angka 0 merupakan representasi dari piksel yang memiliki warna hitam. Jadi dapat disimpulkan bahwa gambar digital adalah data matrik yang memiliki elemen berisi piksel yang memiliki angka atau nilai dan memberi warna.

Representasi matematis sebuah citra ditunjukkan pada Persamaan 2.1.

$$f(x,y) = \begin{bmatrix} f(0,0) & \dots & f(0,B-1) \\ \dots & \dots & \dots \\ f(A-1,0) & \dots & f(A-1,B-1) \end{bmatrix} \quad (2.1)$$

Sebuah citra dapat direpresentasikan sebagai suatu fungsi kontinu 2 dimensi $f(x,y)$ dimana x dan y adalah koordinat bidang dan amplitude dari fungsi f. Secara matematis persamaan citra ditulis sebagai berikut.

$$0 \leq x \leq A-1 \quad (2.2)$$

$$0 \leq y \leq B-1 \quad (2.3)$$

$$0 \leq f(x, y) \leq H - 1 \quad (2.4)$$

Sebuah citra didefinisikan sebagai sebuah fungsi yang memuat kombinasi intensitas warna dengan koordinat tertentu. Fungsi diatas, A dan B menunjukan posisi diskrit sedangkan H adalah skala keabuan. Proses diskritisasi terjadi akibat pengaruh alat pengambilan suatu citra seperti kamera atau sebagainya. Besar nilai H biasanya adalah bilangan bulat pangkat dua seperti 2, 4, 8, 16, 32, dan seterusnya. Representasi citra dibagi menjadi tiga yaitu :

1. Citra Biner, merupakan citra yang terdiri dari dua nilai yaitu 0 (warna hitam) dan 1 (warna putih).
2. Citra Skala Keabuan, merupakan citra memiliki nilai dengan rentang antara 0 sampai 255.
3. Citra RGB, merupakan citra yang memiliki tiga komponen warna keabuan yaitu merah, hijau, dan biru dengan masing-masing komponen memiliki nilai dengan rentang antara 0 sampai 255.

Tujuan utama pengolahan citra digital adalah membantu manusia untuk meningkatkan kualitas suatu citra atau mendeskripsikan karakteristik suatu citra. Komputer atau sensor pada dasarnya tidak dapat melakukan pendefinisian bila tanpa implementasi algoritma kecerdasan. Pengolahan citra digital telah banyak diaplikasikan di berbagai bidang seperti bidang kesehatan untuk membangkitkan citra tubuh manusia sehingga pakar dapat memberikan keputusan dan pandangan klinis berdasarkan observasinya. Pada bidang transportasi sebagai pengenalan plat nomor kendaraan, pendeteksian rambu lalu lintas. Pada bidang pertanian sebagai inspeksi tanaman mendeteksi hama daun hingga kualitas atau tingkat kematangan buah-buahan.

2.2.2 Klasifikasi Dalam Data Mining

Menurut Witten dan Frank, *Data Mining* adalah proses ekstraksi suatu data atau pola (sebelumnya tidak diketahui, bersifat implisit, dianggap tidak berguna) menjadi informasi atau pengetahuan dari data yang jumlahnya besar (Azuaje, 2006). Ilustrasi sederhana dari *Data Mining* seperti semisal ada data yang dianggap sampah karena tidak berpola, tidak terstruktur, dan tidak berguna diolah (difilter) sehingga membentuk informasi atau pengetahuan yang berguna. Dalam Data

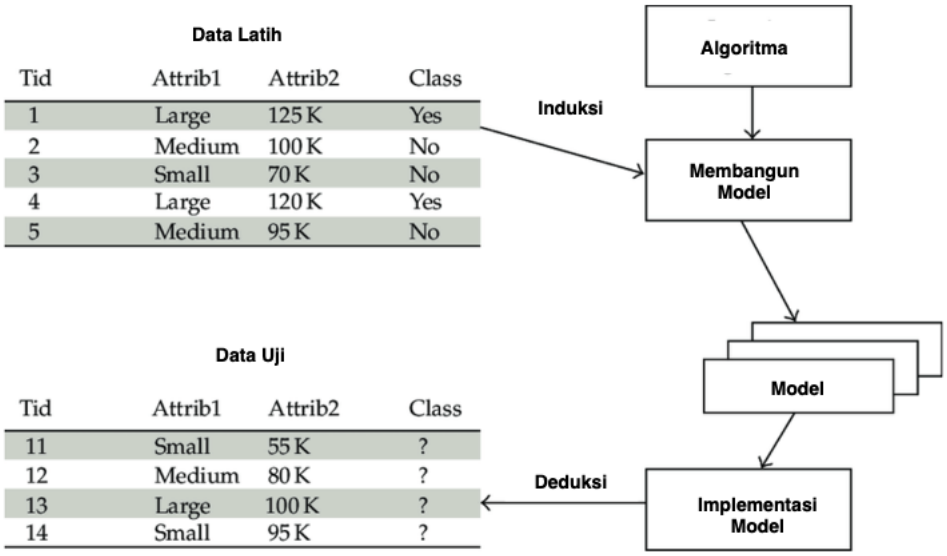
mining secara umum terdapat 5 peranan yaitu Estimasi, Prediksi, Klasifikasi, Clustering, dan Asosiasi (Durugkar dkk., 2022).

Klasifikasi adalah proses untuk menyatakan suatu objek data sebagai salah satu kategori atau kelas atau mempelajari sekumpulan data sehingga dihasilkan aturan yang bisa mengklasifikasikan atau mengenali data-data lain yang belum pernah dipelajari. Dalam model klasifikasi, data input untuk klasifikasi adalah koleksi dari record. Setiap *record* dikenal sebagai *instance* atau contoh, yang ditentukan oleh sebuah *tuple* (x, y), dimana x adalah himpunan atribut dan y adalah sebuah atribut tertentu, yang dinyatakan sebagai label kelas (juga dikenal sebagai kategori atau atribut target) (Vujović, 2021).

Klasifikasi sebagai pemodelan prediktif dapat berguna untuk menentukan kelas yang sesuai berdasarkan fitur-fitur apa yang mendefinisikan. Klasifikasi banyak digunakan dalam berbagai aplikasi diantaranya (Tharwat, 2018):

a. Klasifikasi profil pelanggan, sebagai contoh dapat memperkirakan apakah suatu pengajuan hipotek oleh nasabah merupakan suatu kredit baik atau buruk.
b. Deteksi kecurangan, sebagai contoh dapat menentukan apakah transaksi curang atau bukan.
c. Diagnosis medis, sebagai contoh dapat mendiagnosis penyakit seorang pasien termasuk kategori penyakit apa.

Algoritma klasifikasi sangat banyak seperti pada Pohon Keputusan (Algoritma ID3, C4.5, CART, CHAID, Rainforest), Naïve Bayesian, Neural Network, Algoritma kNN, dan lain-lain. Jadi model yang dibangun berdasarkan algoritma harus sesuai dengan data input dan dapat memprediksi dengan benar label kelas dari record yang belum pernah terlihat sebelumnya. Pendekatan umum model klasifikasi dapat terlihat pada Gambar 2.5.



Gambar 2.5 Pendekatan Umum Model Klasifikasi
(Sumber : Gokceoglu dkk., 2010)

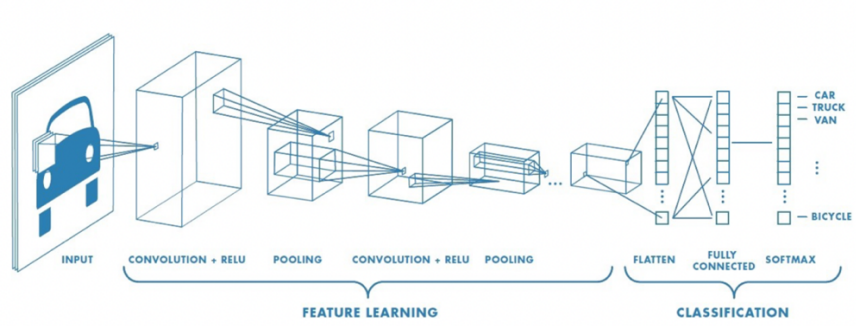
Pada Gambar 2.5, Data Latih *(Training Set)* yang berisi record dengan label kelas diketahui harus tersedia untuk membangun model klasifikasi. Setelah memperoleh model klasifikasi, Model tersebut diaplikasikan ke Data Uji (*Test Set)* yang berisi record dengan label kelas yang tidak diketahui.

2.2.3 Convolutional Neural Network

Convolutional Neural Network merupakan salah satu jenis algoritma klasifikasi *deep learning* yang dapat menerima input berupa gambar dan menentuan aspek atau objek apa saja dalam sebuah gambar yang bisa digunakan mesin untuk belajar mengenali gambar dan membedakan antara satu gambar dengan yang lain (Chauhan dkk., 2018). Arsitektur dari CNN dibagi menjadi 3 bagian besar yaitu *input*, *feature learning*, dan *classification* seperti pada Gambar 2.6.

Data semula akan dibagi berdasarkan banyak *batch size* yaitu jumlah sebaran data yang digunakan pada *neural network*. Data pada tiap *mini-batch size* akan diproses pada tahap *feature learning* (*convolution layer, pooling layer*, ReLU, dan *batch normalization*). Hasil dari *feature learning* berupa vektor fitur dimana akan dijadikan sebagai data input pada tahap klasifikasi. Tahap klasifikasi pada

CNN yaitu *fully connected layer* dan *softmax* untuk mencari nilai probabilitas pada tiap kelas klasifikasi.



Gambar 2.6 Arsitektur CNN
(Sumber : Dhande dkk., 2023)

1. Tahap *feature learning*, tersusun atas beberapa jenis layer sebagai berikut (Prijono, 2018) :
   a. Convolution Layer

      Pada layer ini, citra input akan ditranslasikan menjadi fitur-fitur berdasarkan ciri citra. Perkalian matriks input beserta *padding*-nya dengan matriks filter akan menghasilkan keluaran matriks H. Matriks filter pada proses ini berguna untuk membantu mengeskstraksi ciri fitur yang tepat dan relevan dari tiap citra masukan. Matriks filter merupakan suatu matriks yang berisi bobot bernilai kecil dengan rentang nilai antara -1 sampai 1. Ukuran matriks filter tidak selalu sama pada tiap convolution yang bertujuan untuk mendapatkan pola yang lebih detail, terperinci, dan bervariasi, Misalkan pada ukuran 7x7 kemudian selanjutnya berukuran 3x3. Ukuran matriks H ditentukan dari nilai *stride* dan *padding* yang ditentukan. *Stride* adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah 1, maka *convolutional filter* akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Semakin kecil *stride* maka akan semakin detail informasi yang diperoleh dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. *Padding* atau *Zero Padding* adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input bertujuan untuk meningkatkan performa dari model karena *convolutional filter* akan fokus

pada informasi yang sebenarnya yaitu yang berada diantara *zero padding* tersebut. Untuk menghitung ukuran matriks H, digunakan persamaan 2.5

$$u_h = \frac{w-f+2P}{S} + 1 \quad (2.5)$$

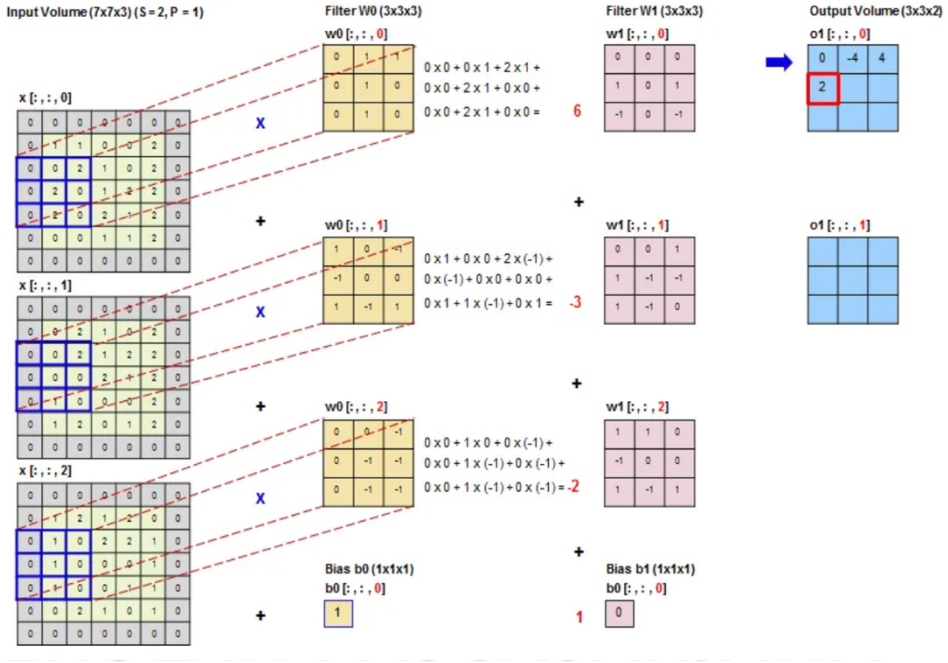
dengan u_h mengacu pada ukuran matriks H, w mengacu pada ukuran matriks input, f mewakili ukuran matriks filter, P menunjukkan *padding* yang diterapkan, dan S adalah nilai *Stride*.

Hasil matriks H menggunakan persamaan sebagai berikut

$$H_{i,j} = c_{(i,j)} * f_{(i,j)} \quad (2.6)$$

dengan C adalah matriks input dan $f_{(i,j)}$ adalah convolutional filter.

$$H_{i,j} = \sum_m \sum_n c_{(m,n)} * f_{(i+m,j+n)} \quad (2.7)$$



Gambar 2.7 Perhitungan *Convolutional Layer*
(Sumber : Raycad, 2017)

Perhitungan *feature map* dilakukan dengan operasi konvolusi antara matriks input dan matriks filter menghasilkan suatu nilai *feature map*, kemudian dilanjutkan dengan pergeseran sebesar *stride* pada matriks input dan dikalikan juga dengan matriks filter hingga memenuhi seluruh *feature*

map. Pada Gambar 2.7, perhitungan *result* atau *feature map* dilakukan dengan operasi konvolusi antara matriks input dengan penambahan *padding* dan matriks filter berukuran 3x3 untuk menghasilkan suatu nilai *feature map*, kemudian dilanjutkan dengan pergeseran sebesar *stride* pada matriks input dan dikalikan juga dengan matriks filter hingga memenuhi seluruh *feature map.*

Dalam matematika (khususnya, analisis fungsional) operasi konvolusi adalah operasi matematika pada dua fungsi (f dan g) yang menghasilkan fungsi ketiga (f*g) yang menyatakan bagaimana bentuk yang satu dimodifikasi oleh yang lain. Nilai matriks filter pada permasalahan nyata algoritma CNN diinisialisasi secara acak dan dipelajari serta dioptimalkan. Beberapa jenis matriks filter pada *convolution layer* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh Jenis Matriks Filter

Jenis matriks filter	Matriks filter
Ridge Detection	$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$
Sharpen	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$
Box Blur	$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
Gaussian Blur 3x3	$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$
Gaussian Blur 5x5	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$

b. Batch normalization

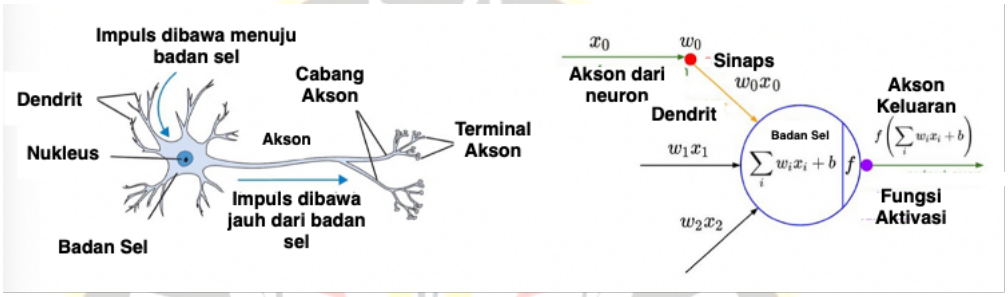
Pada layer ini dilakukan normalisasi untuk mengurangi *overfitting* pada tiap *mini batch* dan meningkatkan pembelajaran *neural network*. Dengan menggunakan persamaan 2.8,

$$\mathrm{BN} = \gamma \frac{I-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta \qquad (2.8)$$

γ dan β mewakili parameter pembelajaran, I adalah matriks input I, μ mewakili nilai rata-rata pada matriks input I, σ^2 adalah nilai varians pada matriks input I, dan ϵ merupakan nilai konstan untuk stabilitas numerik.

c. Fungsi Aktivasi ReLU

Algoritma CNN terdiri dari neuron layaknya bagaimana *neuron* dalam otak manusia bekerja, memiliki *weight*, bias, dan fungsi aktivasi. Ilustrasi algoritma CNN terdapat pada Gambar 2.8.

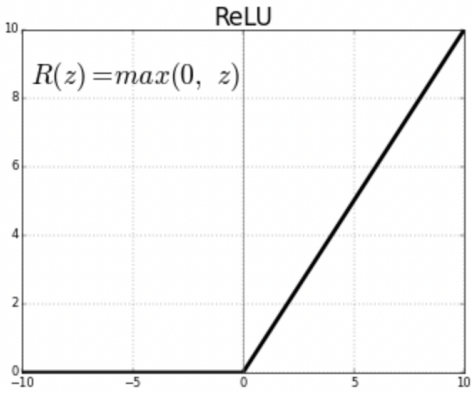


Gambar 2.8 Ilustrasi Neuron dan Model Matematisnya
(Sumber : Kadar & Botnari, 2023)

Tiap neuron menerima input dan melakukan operasi dot dengan sebuah *weight*, menjumlahkannya *(weighted sum)* dan menambahkan bias. Hasil dari operasi tersebut akan dijadikan parameter dari *activation function* yang akan dijadikan output dari *neuron* tersebut. Fungsi aktivasi digunakan untuk menentukan apakah *neuron* tersebut harus "aktif" atau tidak berdasarkan weighted sum dari input. Secara umum ada 2 jenis *activation function* yaitu *Linear* dan *Non-Linear Activation Function*. Contoh *Linear Function* adalah f(x) = x. Sedangkan contoh *non-linear function* adalah sigmoid and *tanh function* dan *ReLU*. Layer *ReLU* merupakan layer yang berfungsi sebagai fungsi aktivasi sehingga nilai pada *feature map* bernilai positif dengan persamaan 2.9.

$$f(x) = \begin{cases} 0, & jika\ x < 0 \\ x, & jika\ x \geq 0 \end{cases} \quad (2.9)$$

ReLU pada Gambar 2.9 melakukan *"treshold"* dari 0 hingga tak hingga. Perubahan *feature map* yang bernilai negatif diganti dengan nilai 0.



Gambar 2.9 *Non-Linear Function*; *ReLU Function*
(Sumber : Joy & Kounte, 2022)

d. Pooling layer

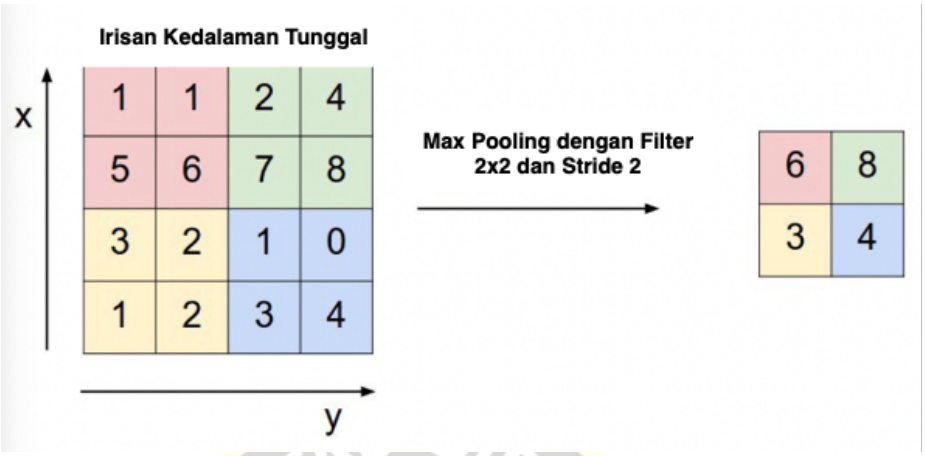
Layer ini biasanya berada setelah *convolutional layer* (Wang, 2018). *Pooling layer* digunakan untuk mengurangi ukuran matriks input dengan cara reduksi secara spasial. Pada prinsipnya, hasil matriks keluaran pada layer ini terdiri dari ukuran filter dan *stride* tertentu yang akan bergeser seperti pada persamaan 2.10

$$u_K = \frac{w-f}{s} + 1 \qquad (2.10)$$

dengan u_K mewakili ukuran matriks K, w mewakili ukuran matriks input, f mewakili ukuran matriks filter, dan S merupakan *stride*.

Pooling yang biasa digunakan adalah *max pooling* dan *average pooling*.

Sebagai contoh pada Gambar 2.10 jika menggunakan *Max Pooling* 2x2 dengan *stride* 2, maka pada setiap pergeseran filter, nilai maksimum pada area 2x2 piksel tersebut yang akan dipilih, sedangkan *Average Pooling* akan memilih nilai rata-rata. Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map (downsampling),* sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi *overfitting* (kondisi ketika model terlalu baik dalam memprediksi data pelatihan, tetapi buruk dalam memprediksi data validasi atau data yang belum pernah ditemui saat proses training).



Gambar 2.10 Contoh *Max Pooling*
(Sumber : Sandhya dkk., 2020)

2. Tahap klasifikasi, tersusun dari *fully connected* yang mentransformasikan hasil dari layer sebelumnya ke bentuk vektor dan mengklasifikasikan citra menggunakan suatu fungsi aktivasi *softmax* yang tersusun atas beberapa jenis layer sebagai berikut (Bhatt dkk., 2021) :

   a. *Fully connected layer*

      Dihasilkan dari *feature map* yang masih berbentuk *multidimensional array* sehingga harus melakukan *flatten* atau *reshape feature map* menjadi 1 dimensi atau sebuah vektor agar dapat digunakan sebagai input pada layer ini. Vektor tersebut akan terhubung dengan jaringan syaraf tiruan dengan Persamaan 2.11

      $$f_{c_j} = b_j + \sum_i x_i W_{i,j} \ , \ j = 1,2,3, \dots, t \qquad (2.11)$$

      dengan b merupakan nilai vektor bias, W mewakili bobot, t mewakili jumlah kelas, dan x merupakan vektor input.

   b. *Softmax*

      Softmax merupakan suatu fungsi aktivasi yang digunakan untuk mengklasifikasikan secara linear dengan menghasilkan nilai yang diinterpretasi sebagai probabilitas terhadap suatu kelas. Hasil nilai keluaran tiap kelas memiliki rentang nilai 0 sampai 1 dan jumlah keluaran semua kelas sama dengan 1. Nilai kelas dihitung dengan fungsi kelas softmax seperti Persamaan (2.12)

$$softmax_j = \frac{e^{fc_j}}{\sum_t e^{fc_j}},\ j = 1,2,3,\dots,t \quad (2.12)$$

$dengan\ f_{c_j}$ sebagai nilai dari *fully connected layer.*

c. *Cross entropy*

Metode yang sering digunakan pada neural network untuk menghitung urutan solusi memusat pada arah yang optimal. *Cross entropy* digunakan sebagai *loss function* untuk memaksimalkan kinerja pembelajaran dimana akan dihitung nilai error dari hasil model tersebut. *Cross entropy* menggunakan Persamaan 2.13

$$C\ E_{t,s} = -\sum_{j=1}^{n} t_j \log(s_j) \quad (2.13)$$

dengan s mewakili nilai keluaran dari *softmax* dan t mewakili target.

2.2.4 Deep Convolutional Neural Network

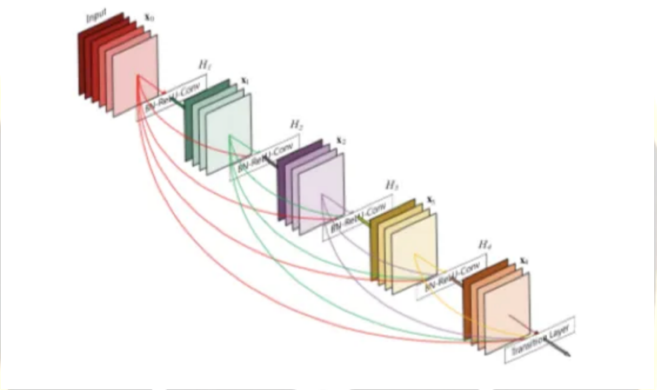
Deep Convolutional Neural Network (Deep CNN) merupakan perkembangan dari arsitektur *CNN*. *CNN* adalah jenis jaringan saraf tiruan yang dirancang khusus untuk bekerja dengan data dalam bentuk *grid*, seperti gambar. *CNN* terdiri dari beberapa lapisan, termasuk lapisan konvolusional, lapisan *pooling*, dan lapisan *fully connected*, yang bekerja secara bersamaan untuk mengekstraksi dan menganalisis fitur-fitur gambar. Seiring dengan perkembangan kebutuhan akan pengolahan citra yang lebih kompleks, *Deep CNN* diperkenalkan dengan jumlah lapisan konvolusional yang lebih dalam (Too dkk., 2019). Sama seperti *CNN*, *Deep CNN* juga dilengkapi dengan lapisan *pooling* untuk mengurangi dimensi data dan lapisan aktivasi (seperti *ReLU*) untuk meningkatkan efisiensi jaringan (Sabyasachi dkk., 2024). Namun, dengan banyaknya lapisan konvolusional, proses ini lebih difokuskan pada ekstraksi fitur tingkat tinggi.

Salah satu perbedaan mendasar antara *CNN* dan *Deep CNN* terletak pada kedalaman lapisan konvolusional yang dimilikinya. *CNN* tradisional memiliki lapisan konvolusional yang lebih dangkal dan terbatas dalam mengenali fitur-fitur dasar (Li dkk., 2020). Sebaliknya, *Deep CNN* memiliki lebih banyak lapisan, yang memungkinkan jaringan ini untuk mengekstraksi dan mengenali fitur-fitur yang lebih abstrak dan kompleks. Keunggulan ini membuat *Deep CNN* lebih efektif

dalam menangani tugas-tugas pengolahan citra yang membutuhkan analisis yang mendalam, meskipun dengan tuntutan komputasi yang lebih tinggi (Li dkk., 2020).

2.2.5 Arsitektur *Dense-Net*

Dense Convolutional Network (DenseNet) merupakan suatu arsitektur jaringan saraf konvolusional yang menghadirkan koneksi yang padat, menghubungkan setiap lapisan atau blok secara langsung dengan setiap lapisan atau blok lainnya melalui umpan maju. Keunggulan utama dari *DenseNet* melibatkan mitigasi masalah gradien, penguatan penyebaran fitur, mendorong penggunaan kembali fitur, dan signifikan mengurangi jumlah parameter (Zhang dkk., 2019).



Gambar 2.11 Arsitektur DenseNet
(Sumber : Tripathi, 2021)

Arsitektur *DenseNet*, seperti yang terlihat pada Gambar 2.11, menunjukkan bahwa setiap komposisi lapisan menggunakan *batch normalization*, *ReLU activation*, dan konvolusi dengan filter 3x3. Dalam setiap blok, input berupa matriks sesuai dengan pixel citra kemudian melalui proses *batch normalization* untuk mengurangi *overfitting* selama proses pelatihan. *ReLU activation* digunakan untuk mengubah nilai menjadi 0 jika bernilai negatif, sementara nilainya tetap dipertahankan jika tidak kurang dari 0 (Wakili dkk., 2022). Proses *convolution* dengan filter 3x3 kemudian dilakukan setelah operasi *ReLU activation*, menghasilkan nilai matriks yang telah diproses sebelumnya.

Dalam konteks *bottleneck*, *convolution* dengan filter 1x1 dijelaskan sebagai lapisan *bottleneck* sebelum setiap *convolution* dengan filter 3x3. Hal ini bertujuan

untuk mengurangi jumlah peta fitur masukan, meningkatkan efisiensi komputasi, dan merujuk pada desain jaringan dengan lapisan *bottleneck*.

Gambar 2.12 menampilkan arsitektur *DenseNet* yang mencakup varian seperti *DenseNet-121*, *DenseNet-169*, *DenseNet-201*, dan *DenseNet-264*. Tiap arsitektur terdiri dari 4 *dense block* dan 3 *transition layer*. Setiap *dense block* terdiri dari *batch normalization*, *ReLU activation*, dan *convolution* dengan filter 3x3. *Transition layer*, yang berada di antara dua blok yang berdekatan, mengubah ukuran fitur melalui operasi *convolution* dan *average pooling* (Zhang dkk., 2019). Tahap klasifikasi menggunakan *global average pooling*, diikuti oleh tahap *softmax activation.*

Layers	Output Size	Dense Net- 121	Dense Net- 169	Dense Net- 201	Dense Net- 264
Convolution	112x112	7x7 conv, stride 2			
Pooling	56x56	3x3 max pool, stride 2			
Dense Block (1)	56x56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56x56	1x1 conv			
	28x28	2x2 average pool, stride 2			
Dense Block (2)	28x28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28x28	1x1 conv			
	14x14	2x2 average pool, stride 2			
Dense Block (3)	14x14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14x14	1x1 conv			
	7x7	2x2 average pool, stride 2			
Dense Block (4)	7x7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1x1	7x7 global average pool			
		1000D fully-connected, softmax			

Gambar 2.12 Layer DenseNet
(Sumber : Anand dkk., 2023)

2.2.6 *Random Search*

Random search adalah pendekatan dalam penyetelan hiperparameter yang memilih sampel secara acak dari ruang pencarian dan menghasilkan penghematan biaya komputasi yang signifikan. Dalam *random search*, hanya sebagian kecil sampel yang diambil untuk dievaluasi sehingga metode ini lebih efisien dalam penggunaan sumber daya komputasi (Villalobos-Arias dkk., 2020).

Dalam *random search*, sampel diambil dari ruang pencarian dan dievaluasi berdasarkan distribusi probabilitas tertentu. Teknik ini menggunakan kombinasi acak dari hiperparameter untuk mencari solusi terbaik untuk model yang sedang dipertimbangkan. Sebagai contoh, dari total kemungkinan kombinasi dalam ruang pencarian, hanya sejumlah kecil sampel yang dipilih secara acak untuk dievaluasi, yang berarti hanya sebagian kecil dari ruang pencarian yang dieksplorasi. Jumlah evaluasi dalam *Random Search* harus ditentukan sebelumnya, sebelum proses optimasi hiperparameter dimulai, sehingga kompleksitas dari *Random Search* dalam menjalankan n evaluasi adalah O(n) (Elgeldawi dkk., 2021). Keunggulan utama dari *random search* adalah kemampuannya untuk mencapai hasil yang memuaskan dengan biaya komputasi yang lebih rendah.

Secara umum, proses dalam *Random Search* dapat dijelaskan melalui langkah-langkah berikut:

1. Menentukan *hyperparameter* yang akan dioptimalkan dalam suatu model pembelajaran mesin atau *deep learning*.
2. Menentukan ruang pencarian untuk setiap *hyperparameter*, biasanya dalam bentuk interval nilai numerik atau sekumpulan pilihan diskrit.
3. Menghasilkan kombinasi *hyperparameter* secara acak dalam ruang pencarian yang telah ditentukan.
4. Melatih model menggunakan kombinasi *hyperparameter* yang dipilih, dengan mengukur kinerjanya pada data validasi.
5. Mengevaluasi performa model berdasarkan metrik evaluasi yang relevan, seperti akurasi, *precision, recall*, dan *F1-score*.
6. Memilih kombinasi *hyperparameter* terbaik berdasarkan hasil evaluasi, yang kemudian dapat diterapkan pada model akhir atau digunakan dalam metode optimasi lanjutan.

2.2.7 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) adalah sebuah algoritma metaheuristik yang terinspirasi oleh perilaku sosial sederhana dalam populasi hewan seperti kelompok burung atau ikan. Dalam *PSO*, satu solusi disebut sebagai partikel,

sementara kumpulan semua solusi disebut sebagai kawanan. Ide utama di balik *PSO* adalah mengatur populasi partikel yang bergerak dalam ruang pencarian solusi dengan mencari posisi yang paling optimal. Setiap partikel dalam *PSO* mewakili sebuah solusi potensial dalam ruang pencarian, dengan masing-masing memiliki posisi dan kecepatan saat ini. Selain itu, setiap partikel juga memiliki dua nilai yang penting: posisi terbaik yang pernah dicapai secara individu *(pbest)* dan posisi terbaik yang pernah dicapai oleh seluruh populasi *(gbest)* (Aguerchi dkk., 2024). Dengan menggunakan konsep ini, *PSO* dapat menemukan solusi optimal dengan menggerakkan partikel dalam ruang pencarian, memungkinkannya untuk menyesuaikan dan mengoptimalkan posisi berdasarkan prestasi individu dan kolektif. Langkah-langkah utama algoritma *PSO* adalah (Aguerchi dkk., 2024):

1. Menginisialisasi populasi partikel, di mana setiap partikel merepresentasikan suatu kombinasi hyperparameter dalam ruang pencarian.
2. Menetapkan batas pencarian *(upper bound* dan *lower bound)* untuk setiap *hyperparameter,* yang akan membatasi ruang eksplorasi.
3. Menentukan nilai awal *hyperparameter* berdasarkan strategi pencarian tertentu, seperti hasil optimasi awal dari metode lain.
4. Melatih model menggunakan kombinasi *hyperparameter* dalam rentang pencarian yang telah ditentukan, guna mengevaluasi performa model dengan berbagai konfigurasi.
5. Mengevaluasi performa model berdasarkan metrik tertentu, seperti akurasi, *precision*, *recall*, dan *F1-score*.
6. Menghentikan iterasi ketika kriteria konvergensi tercapai, misalnya ketika tidak ada perbaikan signifikan dalam beberapa iterasi terakhir atau jumlah iterasi maksimum telah dilewati.

2.2.8 Evaluasi *Confusion Matrix*

Evaluasi klasifikasi didasarkan pada banyaknya *test record* yang diprediksi secara benar dan tidak benar oleh model. Dimana banyaknya test record tersebut dapat ditabulasikan dengan *Confusion Matrix*. *Confusion Matrix* merupakan suatu

matrix yang berbentuk tabel untuk mencatat kinerja klasifikasi (Hasnain dkk., 2020).

Tabel 2.2 *Multiclass Confusion Matrix*

		Actual Classification				
	Classes	C_1	C_2	C_3	…	C_n
Predicted Classification	C_1	T_1	F_{12}	F_{13}	…	F_{1n}
	C_2	F_{21}	T_2	F_{23}	…	F_{2n}
	C_3	F_{31}	F_{32}	T_{33}		F_{3n}
	…	…	…	…	…	…
	C_n	F_{n1}	F_{n2}	F_{n3}	…	T_n

Terdapat empat istilah yang merupakan representasi dari *Confusion Matrix* sebagai berikut (Markoulidakis dkk., 2021) :

a. *True Positive* (TP) : Jumlah data yang bernilai positif dan diprediksi benar sebagai positif, dengan formulasi (2.20).

$$TP_{c_n} = |\{x_i|\hat{y}_i = y_i = c_n\}| \quad (2.20)$$

b. *True Negative* (TN): Jumlah data yang bernilai negatif dan diprediksi benar sebagai negatif, dengan formulasi (2.21).

$$TN_{c_n} = |\{x_i|\hat{y}_i \neq c_n \, dan \, y_i \neq c_n\}| \quad (2.21)$$

c. *False Positive* (FP): Jumlah data yang bernilai negatif tetapi diprediksi sebagai positif, dengan formulasi (2.22).

$$FP_{c_n} = |\{x_i|\hat{y}_i = c_n \, dan \, y_i \neq c_n\}| \quad (2.22)$$

d. *False Negative* (FN): Jumlah data yang bernilai positif tetapi diprediksi sebagai negatif, dengan formulasi (2.23).

$$FN_{c_n} = |\{x_i|\hat{y}_i \neq c_n \, dan \, y_i = c_n\}| \quad (2.23)$$

Pada Tabel 2.2 menampilkan kelas yang diprediksi dan kelas actual yang digunakan untuk memvisualisasikan kinerja masing-masing kelas. Berikut ini metrik kinerja yang digunakan dalam *Multiclass Confusion Matrix :*

a. *Averaged Accuracy* menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar (Salauddin Khan dkk., 2023). Dengan kata lain, *Averaged accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Nilai *Averaged Accuracy* dapat diperoleh dengan

$$Averaged\ Accuracy = \frac{\Sigma_{i=1}^{k} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i}}{k} \quad (2.24)$$

b. *Averaged Precision* (pM) menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model (Salauddin Khan dkk., 2023).

$$Averaged\ Precision\ (pM) = \frac{\Sigma_{i=1}^{k} \frac{tp_i}{tp_i + fp_i}}{k} \quad (2.25)$$

c. *Averaged Recall* (rM) menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi (Salauddin Khan dkk., 2023).

$$Averaged\ Recall\ (rM) = \frac{\Sigma_{i=1}^{k} \frac{tp_i}{tp_i + fn_i}}{k} \quad (2.26)$$

d. *F-1 Score (Averaged F-Measure)* secara definisi adalah *harmonic mean* dari *Averaged Precision* dan *Averaged Recall* yang menggambarkan perbandingan rata-rata *Averaged Precision* dan *Averaged Recall* yang dibobotkan (Riyono dkk., 2022).

$$F\text{-}1\ Score = \frac{(2 * pM * rM)}{(pM + rM)} \quad (2.27)$$

2.2.9 Jenis Ruam Kulit Manusia

Ruam kulit adalah gejala yang sering muncul sebagai respons tubuh terhadap berbagai kondisi kesehatan, termasuk infeksi virus, reaksi alergi, dan masalah kulit lainnya. Selain infeksi virus, seperti cacar air, ruam juga bisa disebabkan oleh infeksi bakteri, reaksi alergi terhadap makanan atau obat-obatan, dan kondisi kulit kronis seperti psoriasis atau eksim Lüthy & Kantor, 2020). Setiap jenis ruam memiliki karakteristik tersendiri, mulai dari tampilan fisik hingga durasi,

yang dapat membantu dalam menentukan penyebab yang mendasarinya dan langkah penanganan yang tepat (Rivianto dkk., 2023). Memahami jenis dan penyebab ruam sangat penting dalam menentukan diagnosis yang tepat dan memberikan penanganan yang efektif.

Cacar air *(chickenpox)* adalah penyakit menular yang disebabkan oleh virus Herpes Varicella atau disebut *Varicella-zooster Virus (VZV)* (Somasekar, 2020). Penyakit ini dapat menyerang segala umur, dengan risiko lebih tinggi terhadap seseorang dengan kekebalan tubuh rendah. Virus ini sangat mudah menular terutama melalui percikan ludah, bersin, batuk dan kontak langsung dengan penderita (Somasekar, 2020).

Gejala yang ditimbulkan, pada permulaanya penderita merasakan demam selama 1-2 hari, Setelah itu akan muncul ruam khas atau timbul kemerahan pada kulit yang berukuran kecil, selama 7-10 hari ruam bentol berisi cairan akan melebar biasanya pertama kali ditemukan disekitar dada dan perut atau punggung lalu menyebar di anggota gerak dan wajah, Sedangkan fase ruam cacar air ada tiga fase (Rivianto dkk., 2023). Fase pertama, akan muncul papule pink atau kemerahan selama beberapa hari. Setelah itu, pada fase kedua akan muncul lepuhan berisi cairan *(vesicle)* yang akan bertahan selama beberapa hari sebelum pecah. Pada fase ketiga, akan muncul krusta yang memerlukan beberapa hari sebelum hilang. Penderita cacar air dapat saja mengalami ketiga fase tersebut dengan bersamaan sampai krusta di kulit benar-benar kering dan hilang (sekitar 5-6 hari sejak muncul papule).

Virus dapat menular mulai dari 2 hari sebelum hingga 5 hari setelah lenting muncul dan sampai semua lenting mengering (Somasekar, 2020). Pengobatan dilakukan kompres air dingin, minum obat berkandungan difenhidramin, loratadine, atau cetirizine untuk atasi gatal, jika bentol di sekitar mulut atas dengan cairan dingin, minum banyak putih, dan istirahat yang cukup (Noronha dkk., 2017). Sedangkan pencegahan yang bisa dilakukan adalah dengan mengisolasi penderita penyakit tersebut dan melalukan vaksinasi, vaksinasi dianjurkan bagi orang-orang berumur diatas 12 tahun yang tidak punya kekebalan ataupun yang belum pernah terkena penyakit ini.

Penyakit campak adalah penyakit infeksi saluran pernapasan akut yang sangat menular disebabkan oleh virus campak *(measles)* dari kelompok paramyxovirus (Lüthy & Kantor, 2020). Penularan umumnya terjadi melalui percikan air liur yang dikeluarkan penderita saat batuk dan juga bersin. Virus campak juga bisa bertahan selama beberapa jam dan mudah menempel pada benda-benda yang disentuh penderita campak.

Penderita demam campak biasanya mengalami gejala gangguan respiratori dan disertai ruam pada kulit. Umumnya, gejala muncul sekitar satu hingga dua minggu setelah tubuh terkena virus campak tersebut. Gejala pada hari pertama hingga hari ke-12 penderita mengalami demam dengan suhu badan yang tinggi, hari ke-2 dan 3 kemunculan bintik-bintik merah, pada hari ke 3 penderita juga mengalami batuk hingga sensitif pada cahaya, pada hari ke-3 hingga hari ke-5 bintik- bintik merah tersebut menyebar ke wajah dan seluruh badan (Maulana, 2021). Sampel citra penyakit campak dilampirkan pada Gambar 2.13.

Gambar 2.13 Penyakit Campak
(Sumber : Bala dkk., 2023)

Demam campak bisa menyebabkan komplikasi seperti sakit pada telinga, penyakit paru- paru (pneumonia), penyakit yang menyerang otak (encephalitis), dan diare (Lüthy & Kantor, 2020). Penyakit ini umumnya menyerang anak-anak berusia lima tahun dan yang belum pernah terkana penyakit campak atau belum mendapatkan vaksinasi. Cara pencegahan mulai dari suntikan imunisasi MMR *(Measles, Mumps, and Rubella)* pada usia 12 bulan dan berumur 7 tahun (Donadel dkk., 2021).

Cacar monyet (*Monkeypox*) merupakan penyakit infeksi virus yang dibebkan oleh virus genus *Orthopoxvirus* dengan famili *Poxviridae* (Farahat dkk., 2022). Penularan dapat terjadi melalui kontak dengan darah, cairan tubuh, atau lesi pada kulit ataumukosa dari binatang yang tertular virus. Sumber penularan penyakit ini bisa terjadi dari monyet,tupai, tikus, dan manusia. Gejala yang ditimbulkan ada demam, sakit kepala hebat, limfadenopati (pembesaran kelenjar getah bening), nyeri punggung, nyeri otot dan lemas, serta ruam yang menyebar ke bagian tubuh lainnya (Kaler dkk., 2022). Ruam ini berkembang mulai dari *macula* (ruam kemerahan yang mendatar), *papula* (ruam kemerahan yang terasa menonjol), *pustula* (ruam kemerahan menonjol berisi cairan) kemudian mengeras (Kaler dkk., 2022).

Masa penularan dan infeksi penyakit cacar monyet terjadi dalam tiga masa (Kaler dkk., 2022). Pertama, masa inkubasi yaitu masa sejak tertular hingga muncul gejala dimana terjadi 5-13 hari atau 5-21 hari. Kedua, masa invasi yaitu masa munculnya gejala seperti demam tinggi, sakit kepala, dan pembengkakan kelenjar dimana terjadi 0-5 hari. Dan ketiga, masa erupsi yaitu masa muncul ruam pada wajah, telapak kaki, telapak tangan, dan lain-lain dimana terjadi 1-3 hari pasca demam.

Cowpox atau penyakit cacar sapi pada manusia adalah infeksi kulit *zoonosis* yang relatif langka, terutama ditemukan di negara-negara Eropa. *Cowpox Virus (CPXV)* termasuk dalam genus *Orthopoxvirus* dari keluarga (Haj Hasan dkk., 2023). Sebagian besar kasus penyakit cacar sapi pada manusia ditularkan melalui kontak dengan kucing domestik yang terinfeksi dan tikus (Krankowska dkk., 2021). Reservoir alami dari penyakit cacar sapi adalah hewan pengerat.

Infeksi penyakit cacar sapi ditandai dengan munculnya lesi pada kulit yang mirip dengan cacar, namun biasanya bersifat lebih ringan. Gejala dapat mencakup demam, nyeri otot, dan pembengkakan kelenjar getah bening, tetapi tidak semua individu mengalami gejala yang sama. Penularan virus ini umumnya terjadi melalui kontak langsung dengan lesi pada kulit hewan yang terinfeksi atau melalui lingkungan yang terkontaminasi (Krankowska dkk., 2021).

Pencegahan infeksi penyakit cacar sapi dapat dilakukan dengan menghindari kontak dengan hewan yang menunjukkan gejala infeksi dan dengan menerapkan praktik kebersihan yang baik. Meskipun penyakit cacar sapi jarang menyebabkan komplikasi serius, penting untuk meningkatkan kesadaran tentang risiko zoonosis ini, terutama di daerah di mana infeksi lebih umum terjadi.

Penyakit kaki, tangan, dan mulut *(HFMD)* adalah infeksi virus yang umumnya menyerang bayi dan anak-anak di bawah usia 5 tahun (Ji dkk., 2024). Penyebab utama *HFMD* adalah enterovirus, khususnya Enterovirus A71 (EVA71) dan Coxsackievirus, yang dikenal luas sebagai penyebab infeksi ini(Dai dkk., 2024).

HFMD biasanya ditandai dengan gejala awal seperti demam, ruam vesikular pada tangan dan kaki, serta luka di mukosa mulut yang dapat menyebabkan ketidaknyamanan saat makan atau minum (Fahim dkk., 2024). Ruam ini sering kali muncul dalam bentuk gelembung yang berisi cairan, yang kemudian dapat pecah dan menjadi luka. Gejala lain yang mungkin menyertai adalah nyeri tenggorokan, kehilangan nafsu makan, dan malaise umum (Ji dkk., 2024).

Pencegahan *HFMD* dapat dilakukan dengan menjaga kebersihan, seperti mencuci tangan secara rutin dan menghindari kontak dengan individu yang terinfeksi (Dai dkk., 2024). Selain itu, meningkatkan kesadaran tentang risiko penularan virus ini di lingkungan, seperti di sekolah dan tempat penitipan anak, juga merupakan langkah penting untuk meminimalkan penyebaran penyakit.

Selain dari penyebab jenis ruam kulit di atas, pada penelitian ini juga akan dilakukan klasifikasi terhadap kulit normal, yaitu kulit yang tampak bersih dan tidak menunjukkan gejala atau gangguan. Pendekatan ini bertujuan untuk membedakan antara kulit yang sehat dan berbagai jenis ruam atau kondisi kulit lainnya. Klasifikasi ini penting untuk memastikan identifikasi yang akurat dan membedakan kondisi kulit yang normal dari yang memerlukan perhatian medis.