# **BAB II KAJIAN PUSTAKA**

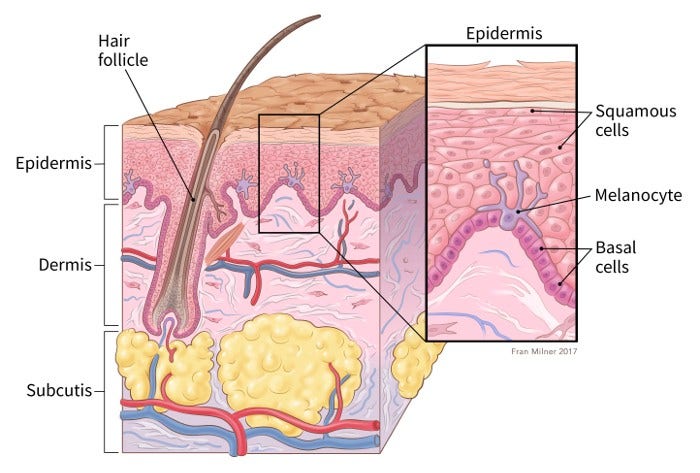
Pada bagian ini, akan diuraikan kajian pustaka yang berkaitan dengan penelitian yang dilakukan oleh penulis. Kajian pustaka terbagi menjadi tiga yaitu kajian teoritis, kajian empiris dan kerangka berpikir.

## **Kajian Teoritis**

1. **Kulit**

Kulit adalah organ terbesar dalam tubuh yang mana organ ini menyumbang sekitar 15% dari total berat badan pada orang dewasa dan sebagai lapisan terluar tubuh, kulit merupakan garis pertahanan pertama terhadap rangsangan eksternal sehingga menjadikannya bagian tubuh paling rentan terhadap cedera (Zidarič et al., 2023). Selain itu, kulit juga berperan penting sebagai penghalang awal terhadap patogen, paparan radiasi yang dipancarkan oleh sinar ultraviolet (UV), dan bahan kimia yang dapat merusak kulit tubuh (Yousef et al., 2024).

Menurut Yousef et al. (2024), sel-sel utama pada epidermis (lapisan kulit terluar) terdiri dari tiga jenis sel diantaranya sel skuamosa (*squamous cell*), sel basal (*basal cell*), dan sel melanosit (*melanocytes cell*) yang diilustrasikan pada Gambar 2.1



Gambar 2. 1 Sel-Sel Pada Lapisan Kulit Terluar

Sumber: (Kadam & Gadade, 2023)

Pada Gambar 2.1 menunjukkan kulit terdiri dari berbagai lapisan dimana lapisan terluar dari kulit manusia adalah lapisan epidermis yang berfungsi sebagai pelindung tubuh dari benda asing. Selanjutnya terdapat lapisan kulit bagian dermis sebagai letak melekatnya (vaskuler) pembuluh darah, kelenjar keringat (sudorifeous), pelindung lemak, serta akar rambut yang letaknya di bawah kulit. Lapisan terdalam dari kulit adalah lapisan hipodermis.

1. **Kanker Kulit**

Kanker kulit dapat diklasifikasikan menjadi 3 tipe utama menurut Gruber & Zito (2023) yaitu karsinoma sel basal, karsinoma sel skuamosa, dan melanoma. Kasus kanker kulit yang memiliki kasus tertinggi antara lain karsinoma sel basal diikuti karsinoma sel skuamosa, dan yang terakhir melanoma maligna. Walaupun melanoma maligna memiliki jumlah kasus kanker lebih kecil dibandingkan kedua tipe kanker kulit yang disebutkan sebelumnya, jumlah kasus kematian yang disebabkannya lebih tinggi. Melanoma maligna sendiri menyebabkan 75% dari total kematian diakibat oleh kanker kulit. Kanker kulit disebabkan oleh berbagai faktor yaitu diantaranya paparan sinar ultraviolet secara berlebihan, riwayat keluarga, paparan bahan kimia, penggunaan *tanning bed*, dan lain sebagainya. Risiko kematian akibat melanoma dapat diminimalkan dengan deteksi dini pada area kulit yang mencurigakan (Yohannes & Al Rivan, 2022).

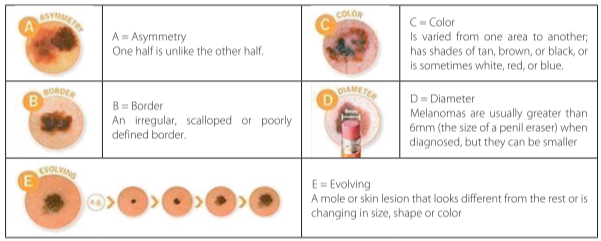
Menurut Hendaria et al. (2013), diagnosis kanker kulit yang dilakukan oleh *dermatologist* atau dokter kulit memiliki beberapa tahapan yaitu anamnesis, pemeriksaan fisik, pemeriksaan dermoskopi, dan pemeriksaan histopatologi. Berikut penjelasannya:

1. *Anamnesis*

*Anamnesis* merupakan tahapan dimana dokter kulit melakukan wawancara medis kepada pasien guna mendapatkan informasi kondisi yang sedang dialami pasien (Oktaviani, 2019).

1. Pemeriksaan Fisik

Pemeriksaan ini dilakukan dengan memeriksa kelainan pada bentuk lesi kulit dengan metode ABCDE (*Asymmetry*, *Border*, *Colour*, *Diameter*, *Evolution*) dimana metode ini dapat memeriksa pasien kanker kulit dari berbagai sudut seperti yang diilustrasikan pada Gambar 2.2.



Gambar 2. 2 Metode ABCDE Untuk Mendeteksi Kanker Kulit

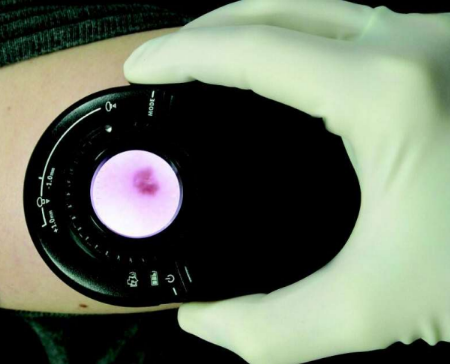
Sumber: (Miranda, 2020)

Pada jurnal yang ditulis oleh Muhammad et al. (2022) terdapat pemaparan mengenai gambaran pemeriksaan kriteria ABCDE mulai dari *Asymmetry* hingga *Evolving*. Pemeriksaan dimulai dari *Asymmetry* dengan melihat bentuk lesi apakah berbentuk simetris atau tidak. Simetris disini dapat diartikan apabila separuh lesi merupakan cerminan dari separuh lesi lainnya baik dari segi warna maupun bentuk. Selanjutnya, pemeriksaan *Border Irregularity* dengan melihat tepian lesi kulit apakah memiliki tepian beraturan dan halus atau tepian yang tidak jelas dan irregular. Irreguler disini dapat diartikan lesi tampak kabur, tidak beraturan atau bergerigi. Selanjutnya, pemeriksaan *Color Variation* dengan melihat variasi warna pada lesi. Selanjutnya, pemeriksaan *Diameter* dengan melihat ukuran lesi kulit dimana diameter lesi lebih dari 6 mm dapat dicurigai sebagai bentuk keganasan. Terakhir, pemeriksaan *Evolving* dengan melihat perkembangan pada lesi kulit itu apakah mengalami perubahan ukuran yang semakin membesar.

1. Pemeriksaan Dermoskopi

Dermoskopi menurut Miranda (2020) merupakan teknik pemeriksaan yang dapat membedakan lesi berpigmen jinak maupun ganas dimana teknik ini dapat memberikan fasilitas visualisasi yang tidak dapat dilihat secara langsung oleh mata sehingga memungkinkan diagnosis yang lebih baik. Teknik pemeriksaan ini memerlukan pelatihan dan keahlian sehingga memungkinkan terjadi penurunan diagnostik apabila dilakukan oleh tenaga medis yang kurang berpengalaman. Dermoskopi yang dilakukan oleh tenaga medis yang telah berpengalaman dapat memperoleh akurasi mencapai 90% sedangkan untuk yang kurang berpengalaman dapat menurun hingga 63%. Oleh karena itu diperlukan teknologi yang dapat membantu proses dermoskopi agar memudahkan tenaga medis dalam mendiagnosis kanker kulit seperti pemrosesan citra digital.

Cara kerja *Dermatoscope* seperti kaca pemesar yang dapat mempermudah *dermatologist* dalam mendiagnosis kanker kulit. Berikut ini contoh pengunaan alat *Dermatoscope* yang dapat dilihat pada Gambar 2.3.



Gambar 2. 3 Dermatoscope

Sumber: (Miranda, 2020)

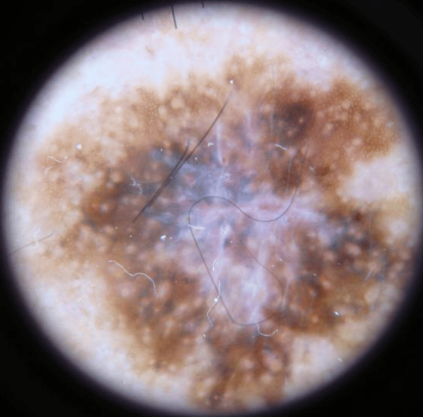
1. Pemeriksaan Biopsi

Biopsi merupakan teknik pengangkatan lesi kulit yang dicurigai sebagai sel kanker yang kemudian lesi kulit tersebut dilakukan pemeriksaan lebih lanjut di laboratorium untuk menentukan tipe jaringan kanker (Miranda, 2020).

1. ***Dermoscopy Image***

*Dermoscopy image* menurut Holmes et al. (2018) merupakan gambar yang diambil dengan alat yang berbasis mikroskopis medis yang berguna untuk meningkatkan diagnosis lesi kulit berdasarkan analisis warna dan strukturnya. Warna pada *dermoscopy* image memberikan indikasi komposisi elemen yang ada pada kulit seperti halnya keratin tampak kuning, darah tampak merah, dan kolagen tampak putih. Warna juga menggambarkan kedalaman pigmen pada kulit karena melanin tampak hitam ketika berada di stratum korneum, coklat ketika berada lebih dalam di epidermis, dan biru ketika berada di dermis. Semakin dekat pigmen kulit dengan epidermis, semakin gelap warna yang dihasilkan oleh *dermoscopy.*

Sistem klasifikasi *dermoscopy* image biasanya menggunakan fitur visual tingkat rendah seperti bentuk, tekstur, dan warna yang mana ekstraksi fitur citra ini umumnya mengukur luas atau dimensi lesi, menganalisis pola tekstur dari gambar abu-abu, dan menghitung rata-rata atau variasi warna pada area tersebut (Celebi et al., 2019). Berikut ini contoh dari gambar dermoskopi yang dapat dilihat pada Gambar 2.4.



Gambar 2. 4 Dermoscopy Image

Sumber: (Torres et al., 2010)

1. **Klasifikasi Citra**

Definisi klasifikasi menurut Nursyahfitri et al. (2021) merupakan suatu teknik yang berguna untuk mendeskripsikan serta mengelompokkan suatu pola ke dalam kategori tertentu berdasarkan kelakuan dan atribut yang telah didefinisikan pada suatu kelompok. Teknik ini memungkinkan dapat memberikan klasifikasi kelas atau kategori pada data baru berdasarkan hasil dari klasifikasi data yang digunakan untuk membuat aturan-aturan.

Klasifikasi menggunakan metode *supervised induction* dimana model dilatih dengan set data yang telah terklasifikasi (berlabel) untuk menghasilkan prediksi label dari data yang belum diketahui (Mardhotillah, 2023).

Klasifikasi dalam *data mining* dianggap sebagai proses menganalisis data dengan menggunakan data pelatihan yang berguna untuk membangun sebuah model untuk memprediksi data ke dalam suatu kelas label (Mardhotillah, 2023).

Definisi citra dalam buku yang ditulis oleh Sulistiyanti et al. (2016) adalah sekumpulan piksel yang tersusun dalam bentuk matriks dua dimensi dimana setiap elemen dalam matriks ini disebut piksel, yang memiliki nilai intensitas tertentu. Titik asal (0,0) biasanya terletak di sudut kiri atas citra, berbeda dengan koordinat grafik yang umum digunakan.

Menurut Palupi Rini et al. (2024), klasifikasi citra pada citra digital merupakan proses pengelompokkan piksel dalam suatu citra digital ke dalam beberapa kelas yang sesuai sehingga setiap kelas dapat menggambarkan suatu entitas dengan kriteria tertentu. Hasil dari klasifikasi citra adalah menghasilkan peta tematik yang mewakili suatu objek tertentu (Pangestu et al., 2020). Pada umumnya pendekatan berbasis piksel terbagi menjadi klasifikasi terbimbing (*supervised classification*) dan klasifikasi tidak terbimbing (*unsupervised classification*) (Palupi Rini et al., 2024).

1. **Machine Learning**

*Machine learning* merupakan bagian dari kecerdasan buatan yang memungkinkan komputer berperilaku layaknya manusia dimana komputer sendiri memiliki kemampuan untuk belajar dan membuat keputusan dari pengalaman tanpa harus diprogram secara berulang (Retnoningsih & Pramudita, 2020).

Menurut Bella (2022), penanganan data yang cukup besar pada *machine learning* memerlukan metode yang tepat agar mendapatkan hasil yang baik. Terdapat metode pembelajaran yang dapat dibedakan menjadi *supervised learning*, *unsupervised learning*, dan *semi supervised learning*, dan *reinforcement learning*. Supervised learning merupakan salah satu teknik dimana model dilatih menggunakan dataset yang telah diberikan label yang bertujuan agar mesin dapat dapat mengidentifikasi dan memprediksi label baru berdasarkan fitur-fitur yang dipelajari. Sedangkan, *Unsupervised learning* melibatkan pembelajaran mesin tanpa label yang telah ditentukan dimana model menarik kesimpulan dari dataset untuk mengidentifikasi struktur atau pola tersembunyi.

1. **Deep Learning**

*Deep learning* merupakan bagian dari pembelajaran mesin dimana algoritmanya terinspirasi dari struktur otak manusia yang memiliki banyak neuron yang saling terhubung, struktur ini disebut dengan *Artificial Neural Network* (ANN). Jaringan syaraf tiruan ini memiliki lebih dari tiga lapisan ANN sehingga memungkinkan model mampu memahami dan beradaptasi dataset yang besar serta menyelesaikan berbagai permasalahan kompleks yang sulit diselesaikan oleh algoritma pembelajaran mesin lainnya (Raup et al., 2022). *Deep learning* memberikan arsitektur yang sangat kuat untuk *Supervised Learning*.

Pada pembelajaran mesin terdapat teknik dimana teknik ini berguna untuk mengekstraksi fitur dari data *training* dan algoritma pembelajaran yang dikhususkan untuk pengklasifikasian citra. Dengan menambahkan lebih banyak lapisan, model pembelajaran dapat mewakili data citra berlabel dengan lebih baik. Meskipun metode ini memiliki beberapa kekurangan dalam hal kecepatan dan akurasi, konsep ini memungkinkan komputer untuk belajar dengan kecepatan dan skala yang lebih besar. Dalam konteks aplikasi, ini berarti kemampuan untuk mengatasi permasalahan yang lebih kompleks dan memanfaatkan data yang lebih besar (Silalahi, 2020).

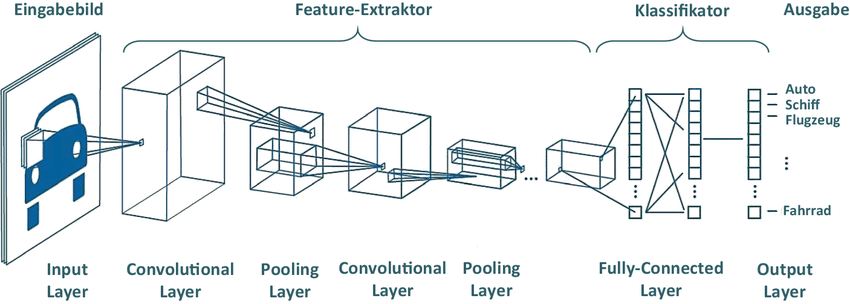
Pada penelitian ini menggunakan *deep learning* dengan ANN dimana ANN ini memiliki banyak lapisan yang memungkinkan komputer mampu belajar dengan kecepatan dan skala yang lebih besar serta mengatasi permasalahan yang kompleks. Dalam praktiknya *deep learning* telah berhasil dalam membantu memecahkan banyak masalah pada *dataset* besar seperti pengenalan wajah, deteksi objek, dan bahkan diagnosis medis.

### **Convolutional Neural Network (CNN)**

*Convolutional Neural Network* (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang dirancang khusus untuk mengolah data dua dimensi. CNN termasuk dalam kategori algoritma *Deep Neural Network* (DNN) karena memiliki kedalaman jaringan yang sangat dalam dan sering digunakan pada data citra. Klasifikasi citra dengan MLP dikatakan kurang cocok karena MLP tidak dapat mempertahankan informasi spasial yang mana MLP menganggap setiap piksel sebagai fitu yang terpisah sehingga hasilnya kurang optimal. Sebaliknya CNN terbukti lebih unggul dibandingkan metode machine learninglainnya dalam klasifikasi objek pada citra (Azhar, 2022).

*Convolutional Neural Network* (CNN) merupakan arsitektur jaringan saraf tiruan yang teruji dan telah terbukti sangat efektif dalam pengklasifikasian gambar. Dalam CNN ini sendiri terdapat operasi konvolusi yang memungkinkan ekstraksi fitur dari gambar secara otomatis, sehingga memudahkan pengenalan pola dan klasifikasi gambar. Teknik ini meningkatkan efisiensi pembelajaran mesin dalam pengolahan citra (Arsal et al., 2020).

Pada CNN, Operasi konvolusi menggabungkan beberapa lapisan pada saat pemrosesan dimana tiap neuronnya direpresentasikan dalam bentuk dua dimensi sehingga memungkinkan metode ini cocok digunakan dalam pemrosesan dengan inputan berupa data citra (Hamsy Romario et al., 2020). Arsitektur jaringan yang digunakan oleh CNN dapat dilihat pada Gambar 2.5.



Gambar 2. 5 Arsitektur Convolutional Neural Network

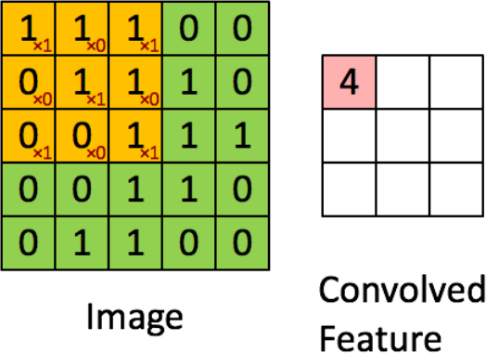
Sumber: (Zschech et al., 2021)

Pada Gambar 2.5 menunjukkan ilustrasi struktur jaringan pada CNN yang terdiri dari *input*, proses ekstraksi fitur, proses klasifikasi dan *output*. Proses ekstraksi dalam CNN terdiri dari beberapa lapisan tersembunyi atau *hidden layer* seperti lapisan konvolusi, fungsi aktivasi (ReLU), dan *pooling*. CNN bekerja secara hierarki, sehingga *output* dari lapisan konvolusi pertama menjadi *input* untuk lapisan konvolusi berikutnya. Proses klasifikasi melibatkan *fully-connected layer* dan fungsi aktivasi (*softmax*) sebagai *output* yang mana hasil akhirnya adalah klasifikasi (Dwi Setyawan et al., 2023).

1. Lapisan Konvolusi

Lapisan konvolusi merupakan proses utama yang harus dilalui dalam tahapan *feature learning* dan merupakan proses dasar dalam sebuah CNN dimana lapisan ini melakukan operasi konvolusi pada *output* dari layer sebelumnya (Nelson, 2023).

Dalam konteks citra, konvolusi adalah proses mengekstraksi fitur dari citra input dengan menerapkan fungsi kernel (kotak kuning) pada citra di semua posisi yang memungkinkan. Ilustrasi dari operasi konvolusi dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Operasi Konvolusi

Sumber: (Lakshmanna et al., 2021)

Pada tahapan ini dilakukan antara matriks *input* dan matriks *filter* dimana *filter-filter* ini akan bergeser ke seluruh area citra secara berulang yang menghasilkan matriks *feature map* sebagai *output* (Nelson, 2023). Setelah tahapan konvolusi pada data citra selesai, tahapan selanjutnya adalah menerapkan *Rectification Linear Uni*t (ReLU).

1. Fungsi Aktivasi ReLU

*Rectification Linear Uni*t (ReLU) merupakan fungsi aktivasi pada jaringan syaraf tiruan yang bertindak sebagai operasi thresholding yang bertujuan untuk memastikan bahwa hasil dari konvolusi berada dalam domain positif. Fungsi aktivasi ReLU didefinisikan sebagai berikut.

(2.1)

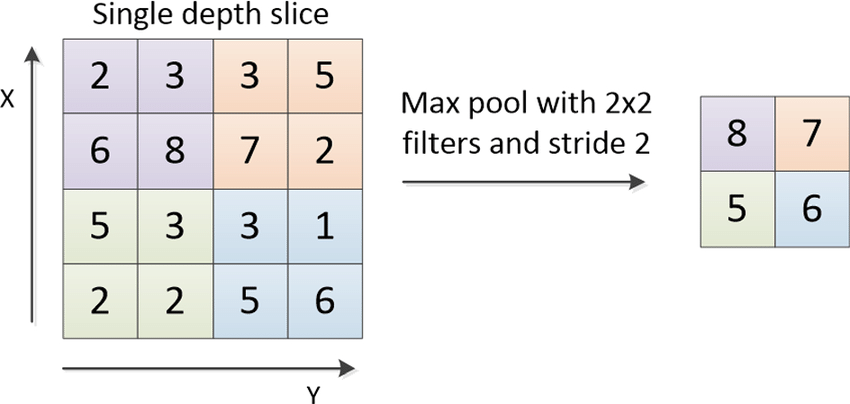
Apabila *input* yang diberikan negatif, maka hasilnya akan menjadi 0. Namun, jika inputnya positif, hasilnya akan sama dengan nilai *input* itu sendiri. Oleh karena itu, setiap hasil konvolusi yang bernilai negatif akan diproses oleh ReLU terlebih dahulu untuk diubah dari negatif menjadi 0 (Nelson, 2023).

1. *Padding*

*Padding* adalah teknik yang digunakan dalam CNN untuk menambahkan lapisan piksel yang biasanya berisi nol di sekitar gambar *input*. Hal ini memungkinkan ukuran *output* dari *convolutional layer* agar tetap sama dengan ukuran *input*, atau agar tidak berkurang secara signifikan setelah oeprasi konvolusi. Dengan *padding*, *convolutional layer* dapat menganalisis data di tepian gambar yang memungkinkan ekstraksi ciri yang lebih mendalam dan detail dari seluruh gambar (Magdalena et al., 2021).

1. Lapisan *Subsampling*

*Subsampling* adalah proses pengurangan ukuran matriks atau data citra yang berguna untuk meningkatkan invariansi posisi fitur. Sebagian besar metode *pooling* yang digunakan pada CNN adalah *max pooling*. Proses ini membagi *output* dari lapisan konvolusi menjadi beberapa *grid* kecil, yang kemudian setiap *grid* akan diambil nilai maksimumnya untuk membentuk matriks citra yang direduksi (Nelson, 2023). Berikut ini ilustrasi dari operasi *max pooling* dapat dilihat pada Gambar 2.7.



Gambar 2. 7 Operasi Max Pooling

Sumber: (Qiu et al., 2021)

Berdasarkan Gambar 2.7 terlihat bahwa *grid* dengan warna ungu, oranye, hijau, dan biru mewakili kelompok grid yang akan memilih nilai tertingginya. Hasil dari proses ini adalah sebuah matriks peta fitur yang berisi nilai-nilai maksimum yang telah dipilih, seperti yang ditunjukkan pada gambar di sebelah kanan. Proses ini memastikan bahwa fitur-fitur yang diperoleh tetap konsisten meskipun ada pergeseran pada objek dalam citra (Nelson, 2023).

1. Lapisan *Fully Connected*

Lapisan *fully connected*, yang umum diterapkan dalam MLP adalah lapisan di mana setiap lapisan sebelumnya terhubung dengan setiap neuron di setiap lapisan berikutnya. Lapisan ini berfungsi untuk mengolah dimensi data sehingga data dapat diklasifikasikan secara linear (Mbaba et al., 2022). Sebelum masuk ke lapisan fully connected, setiap neuron dari lapisan sebelumnya ditransformasi terlebih dahulu menjadi data satu dimensi. Proses ini menyebabkan hilangnya informasi spasial dari data dan tidak bisa dikembalikan seperti semula. Oleh karena itu, lapisan fully connected diimplementasikan di bagian akhir jaringan (Pangestu et al., 2020).

Feature map merupakan hasil dari lapisan sebelumnya atau lapisan *feature extraction* yang masih berupa multidimensional array sehingga perlu dilakukan transformasi dengan metode “*flatten*” atau *reshape feature map* agar menjadi sebuah vector dan dapat dimasukan kedalam lapisan *fully connected* (Peryanto & Rusydi, 2020).

Perbedaan yang dimiliki oleh lapisan *fully connected* dengan konvolusi biasa adalah terletak pada neuronnya. Pada lapisan konvolusi neuron hanya terhubung ke daerah tertentu pada masukan. Sedangkan, pada lapisan *fully connected* neuron terhubung ke seluruh daerah pada masukan. Namun, kedua lapisan tersebut memiliki fungsinya yang tidak jauh berbeda karena sama-sama masih mengoperasikan produk dot (Magdalena et al., 2021).

1. Fungsi Aktivasi *Softmax*

Fungsi aktivasi *softmax* digunakan untuk memperoleh hasil klasifikasi. Fungsi ini menghasilkan nilai yang dapat diinterpretasikan sebagai probabilitas yang belum dinormalisasi untuk setiap kelas. (Hamsy Romario et al., 2020).

1. *Overfitting*

*Overfitting* merupakan kondisi dimana model pembelaran mesin memiliki performa yang sangat baik pada data latih tetapi performa menurun secara signifikan saat diuji dengan data baru atau data uji (Guntoro et al., 2022).

1. *Dropout*

*Dropout* adalah teknik yang digunakan dalam jaringan syaraf untuk mengurangi resiko *overfitting*. *Overfitting* sendiri terjadi ketika model terlalu menyesuaikan diri dengan data latih sehingga berdampak pada performa model menurun pada data yang belum pernah dilihat sebelumnya. Selain itu, *dropout* dapat mempercepat proses pelatihan dengan cara memilih secara acak dan mengurangi neuron-neuron yang tidak tergunakan lagi (Guntoro et al., 2022).

1. *Batch Normalization*

*Batch Normalization* adalah teknik yang digunakan untuk menstandarisasi lapisan input dengan setiap *mini batch*. Teknik ini bertujuan untuk meningkatkan kecepatan dan efisiensi pelatihan dengan mengurangi pergeseran kovariat internal. Maksud dari pergeseran kovariat internal ini merujuk pada perubahan distribusi input selama proses pelatihan yang dapat mempengaruhi stabilitas jaringan. Dengan teknik ini, mampu meningkatkan performa, kecepatan, dan stabilitas jaringan syaraf dalam (Fandisyah et al., 2021).

1. *Epoch*

Dalam konsep pembelajaran mesin, *epoch* adalah satu siklus lengkap dimana algoritma telah melatih menggunakan seluruh sat data latih. *Epoch* digunakan untuk menentukan seberapa lama jaringan CNN dilatih (Wasil, 2022). Sebagai contoh, peneliti memiliki 1000 gambar dan menjalankan pelatihan dengan 10 epoch, maka algoritma akan melalui seluruh 1000 gambar tersebut sebanyak 10 kali.

1. **Arsitektur VGG-16**

VGG merupakan arsitektur jaringan syaraf konvolusional (CNN) yang diusulkan oleh Karen Simonyan dan Andrew Zisserman dari *Visual Geometry Group* (VGG) yang ada di Universitas Oxford. VGG sendiri memiliki beberapa macam tergantung pada jumlah lapisan konvolusionalnya. Arsitektur VGG yang paling sering ditemui yaitu VGG-16 dan VGG-19. Model VGG-16 mencapai akurasi uji top-5 sebesar 92.7% pada *dataset* ImageNet yang berisi 14 juta gambar dari 1000 kelas yang berbeda. Simonyan dan Zisserman menggantikan *filter kernel* berukuran besar dengan beberapa *filter kernel* berukuran 3 × 3, yang menghasilkan kinerja yang lebih baik dibandingkan dengan model sebelumnya, yaitu AlexNet (Naqvi et al., 2023).

1. **Python**

Python adalah bahasa pemrograman serbaguna yang cukup terkenal dikalangan para pengembang untuk berbagai jenis aplikasi baik itu aplikasi *desktop*, *mobile*, situs *web*, *game*, robotik, *data mining*, hingga kecerdasan buatan (Kosudiwandi, 2020). Hal tersebut dikarenakan sintaksnya yang sederhana dan menyerupai bahasa manusia sehingga lebih mudah dipahami dan digunakan. Selain itu, Python memungkinkan aplikasi berjalan dengan lebih cepat dibandingkan dengan bahasa pemrograman lainnya (Gat et al., 2023).

Python merupakan bahasa pemrograman yang memiliki keunggulan dan manfaat di berbagai platform. Kelebihannya termasuk tidak memerlukan kompilator dan *interpreter* sehingga dapat menghemat waktu dalam pengembangan program. *Interpreter* Python juga dapat digunakan secara interaktif, memudahkan pengembang aplikasi dalam mengevaluasi fungsionalitas, bereksperimen dengan fitur bahasa, dan merancang program sementara (Rahman, 2023).

1. **Kotlin**

Kotlin merupakan bahasa pemrograman yang terkenal dalam pengembangan aplikasi *smartphone*, terutama untuk *platform Android*. Selain *Java*, bahasa pemrograman ini mendapat dukungan resmi dari *Google* dan kompatibel dengan Android Studio. Bahasa pemrograman ini menawarkan pendekatan modern dan statis. Penggunaan *Kotlin* memungkinkan eksekusi di *platform* JVM atau *Java Virtual Machine*. Selain itu, *Kotlin* menggunakan *compiler* LLVM yang memungkinkan dikompilasi menjadi *JavaScript*.

Penggunaan bahasa pemrograman Kotlin terbukti sangat efisien karena selain menawarkan fitur-fitur yang mudah dipahami, Kotlin bersifat *open-source* yang memungkinkan penggunaan Bahasa pemrograman lebih sederhana dan kompatibel dengan *Java* serta terintegrasi dengan *Android Studio*. Keunggulan yang dimiliki antara lain kecepatan *upgrade* yang tinggi dengan pengamanan otomatis yang menjadikannya pilihan yang cocok untuk beragam *platform* (Gita et al., 2023).

1. **Tensorflow**

Tensorflow merupakan kerangka kerja *open-source* yang didesain untuk pengembangan, pelatihan, dan penerapan model pada deteksi objek. *Framework* Tensorflow ini telah banyak digunakan oleh tim Google Brain dalam divisi AI Google dimana *framework* ini diterapkan pada berbagai produk Google antara lain pada fitur pencarian gambar, fitur pengenalan wajah, dan deteksi plat nomor kendaraan pada *google street view*, fitur *google assistant*, dan masih banyak lagi (Syarif, 2021).

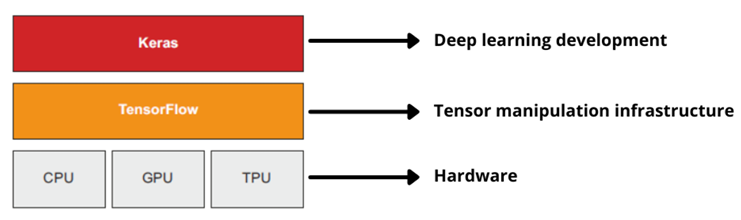
Menurut Syarif (2021), Tensorflow memiliki beberapa keunggulan utama. Salah satunya adalah penggunaan XLA, sebuah kompilator aljabar linear canggih yang memungkinkan kode Tensorflow berjalan dengan sangat cepat pada berbagai jenis prosesor, termasuk CPU, GPU, TPU, dan perangkat keras lainnya. Selain itu, Tensorflow menawarkan API tingkat tinggi yang memudahkan pengembangan dan pelatihan model, serta menyediakan control tingkat rendah untuk fleksibilitas dan kinerja yang optimal.

Tensorflow memiliki suatu pustaka yang memfasilitasi proses pembelajaran mesin yang memungkinkan para pengembang mengimplementasikan model *machine learning* mereka pada perangkat *mobile*, mikrokontroler dan perangkat lainnya yaitu *Tensorflow Lite*.

1. **Keras**

Keras merupakan sebuah pengembangan *framework* yang berguna untuk membantu dalam pembelajaran komputer. Keras juga merupakan salah satu perpustakaan dalam jaringan syaraf tiruan tingkat tinggi yang diimplementasikan dengan bahasa pemrograman Python. Perpustakaan ini menawarkan fitur-fitur yang berguna untuk memudahkan pengembangan lebih dalam terhadap model *Deep Learning* (W. B. Pratiwi et al., 2023).

Keras dibangun di atas Tensorflow sehingga dapat berjalan di atas berbagai jenis perangkat keras GPU, TPU, atau CPU biasa seperti yang diilustrasikan pada Gambar 2.8 sebagai berikut.



Gambar 2. 8 Struktur Framework Keras

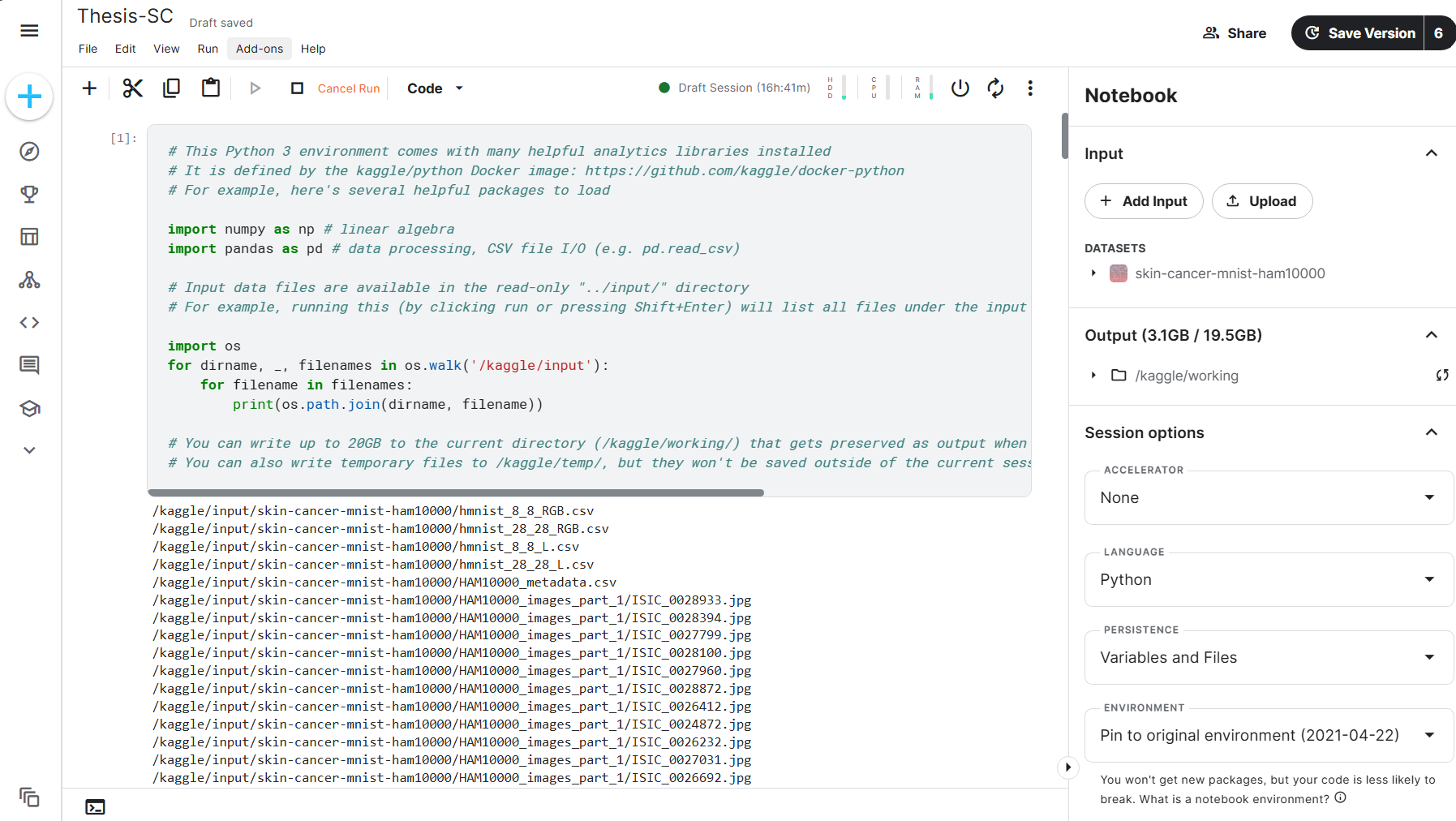
Sumber: (Sabri, 2022)

Awalnya, Keras dibangun di atas Theano dan dikembangkan oleh *Montréal Institute for Learning Algorithms* (MILA) di Université de Montréal. Setelah rilisnya Tensorflow, Keras berkembang menjadi arsitektur multi-backend yang memungkinkan penggunaannya dengan Theano atau Tensorflow. Selain itu, Keras juga menambahkan dukungan untuk dua backend tambahan yaitu Theano dan CNTK. Namun, pengembangan Theano dan CNTK dihentikan, dan MXNet jarang digunakan oleh perusahaan di luar Amazon. Akibatnya, satu-satunya opsi backend yang tersisa adalah Tensorflow, menjadi Keras populer sebagai cara yang mudah untuk mengembangkan aplikasi menggunakan Tensorflow. Akhirnya, Keras dipilih secara resmi oleh pimpinan Tensorflow sebagai API tingkat tinggi dari Tensorflow (Sabri, 2022). Framework Keras pada penelitian ini berguna untuk membantu dalam pengembangan model klasifikasi *skincancer disease* pada citra dermoskopi lesi kanker kulit.

1. **Kaggle Notebooks**

*Kaggle notebooks* merupakan lingkungan komputasi yang memiliki versi berbeda-beda, berbasis pada kontainer *Docker* yang dijalankan di mesin *cloud*, yang memungkinkan untuk menulis dan menjalankan *script coding* dan *notebook* dalam bahasa R dan Python. Kelebihan yang dimiliki *Kaggle notebooks* antara lain terintegrasi dengan lingkungan Kaggle (dapat menyimpan hasil kerja di Kaggle), dilengkapi dengan segala kebutuhan untuk keperluan *data science,* diperbolehkan melakukan sedikit kostumisasi (seperti mengunduh file dan memasang paket tambahan).

*Notebook Kaggle* dasar hanya berbasis CPU, tetapi ada versi yang ditingkatkan dengan NVIDIA Tesla P100 atau TPU v3-8. TPU adalah perangkat keras yang dipercepat khusus untuk tugas *deep learning*. Selain total waktu berjalan, *notebook* CPU dan GPU dapat berjalan maksimal 12 jam per sesi sebelum berhenti (*notebook* TPU hanya 9 jam) (Banachewicz & Massaron, 2022). Contoh ilustrasi dari *Kaggle notebooks* dapat dilihat pada Gambar 2.9.



Gambar 2. 9 Tampilan Kaggle Notebooks

1. **Android Studio**

Dalam buku yang berjudul “Android Apprentice”, dijelaskan bahwa Android Studio adalah *Integrated Development Environtment* (IDE) resmi untuk membangun aplikasi *android*. IDE ini didasarkan pada IntelliJ IDEA dimana sebuah lingkungan terintegrasi (IDE) yang dikembangkanoleh JetBrains. Android Studio menyediakan serangkaian alat yang kuat untuk pengembangan aplikasi Android. IDE ini dapat dijalankan pada berbagai sistem operasi (Bayliss et al., 2019).

Android Studio memiliki banyak fitur unggulan yang dapat dimanfaatkan untuk mengembangkan aplikasi *Android*. Yudhanto & Wijayanto (2018) menyebutkan bahwa beberapa fitur tersebut termasuk Instan Run, yang mempercepat siklus pengeditan, pembuatan, dan menjalankan aplikasi, sehingga mempermudah pekerjaan pengguna. Selain itu, Android Studio menyediakan lingkungan pengembangan terpadu untuk semua perangkat android. Dengan dukungan Gradle, pengguna dapat mengotomatisasi pembuatan aplikasi berkinerja tinggi, mengelola dependensi yang kuat, dan menyesuaikan konfigurasi versi sesuai kebutuhan.

Berdasarkan penjelasan di atas, peneliti memilih Android Studio sebagai perangkat lunak untuk membantu mengambangkan aplikasi android yang direncanakan. Selain berbagai keunggulan fitur yang ditawarkan, peneliti juga merasa lebih mudah memahami bahasa pemrograman yang digunakan di Android Studio.

1. **Flowchart**

Dalam buku yang ditulis oleh Setiawan (2016), menjelaskan bahwa *Flowchart* adalah cara untuk mempermudah dalam membaca bahasa pemrograman yang digambar dengan menggunakan symbol tertentu. *Flowchart* berguna untuk menuliskan alur program dalam bentuk gambar atau *symbol*.

*Flowchart* dibagi menjadi dua bagian, yaitu *flowchart* yang menggambarkan alur suatu sistem dan program. *Flowchart* terdapat simbol-simbol tertentu untuk dapat membaca alur sistem dalam bentuk gambar agar mudah dipahami sebagai berikut:

1. Simbol *Flowchart* Sistem

Simbol pada *flowchart* sistem dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Simbol Sistem Flowchart

| No | Gambar | Keterangan |
| --- | --- | --- |
| 1. | Manual Input | Manual *input* (*keyboard*) atau sebagai *entry* data |
| 2. | Card | Sumber data berasal dari kartu data |
| 3. | Punched Tape | Menunjukan mesin pembolong pita |
| 4. | Stored Data | Media penyimpanan data dapat berupa *harddisk* |
| 5. | Sequential Access Storage | Media penyimpanan data berupa pita yang dibaca berurut |
| 6. | Magnetic Disk | Media penyimpanan data seperti *Floppy disk* |
| 7. | Direct Access Storage | Media penyimpanan data yang dapat dibaca/disimpan secara acak |
| 8. | Display | Media untuk melihat hasil proses dilayar (monitor) |
| 9. | Document | Menunjukkan data akan dicetak ke kertas melalui mesin printer |

Sumber: (Setiawan, 2016)

1. Simbol *Flowchart* Program

Simbol pada *flowchart* program dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Simbol Flowchart Program

|  |  |  |
| --- | --- | --- |
| No | Gambar | Keterangan |
| 1. | Terminator | menunjukan awal dan akhir dari suatu alur program *flowchart* |
| 2. | Process | menunjukan proses seperti perhitungan aritmatik, penulisan suatu formula atau dapat bersisi pemberian nilai terhadap variabel |
| 3. | Read/Write | menunjukan sumber data yang akan diproses atau dapat juga menunjukan data yang akan dicetak/ditulis. |
| 4. | Decision | menunjukan suatu proses evaluasi atau pemeriksaan terhadap nilai data dengan operator relasi. |
| 5. | Preparation | menunjukan deklarasi atau pemesanan variabel atau konstanta |
| 6. | Sub Program | menunjukan sub program yang akan diproses dapat berupa *procedure* dan *function* |
| 7. | Connector | menunjukan tanda sambungan dari suatu *flowchart* pada satu halaman kertas |
| 8. | Off page connector | menunjukan tanda sambungan dari suatu flowchart untuk beda halaman kertas |
| 9. | Arrow | menunjukan arah dari suatu proses dapat ke atas, bawah kanan dan kiri. |

Sumber: (Setiawan, 2016)

1. ***Unified Modelling Language* (UML)**

Dalam buku yang berjudul “Diagram UML Dalam Membuat Aplikasi Android Firebase Studi Kasus Aplikasi Bank Sampah”, *Unified Modelling Language* (UML) dapat dijelaskan sebagai bahasa pemodelan yang berdasarkan gambar atau grafik yang berguna untuk memvisualisasikan, menspesifikasi, mengkonstruksi, dan mendokumentasikan *artifacts* berupa model, deskripsi, ataupun perangkat lunak dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non-perangkat lunak lainnya. UML sendiri merupakan bahasa pemodelan yang berorientasikan objek dan dapat digunakan dalam semua bidang yang membutuhkan pemodelan tidak hanya pada perangkat lunak (Destriana et al., 2021).

Diagram-diagram pada UML yang digunakan dalam beberapa tahapan pengembangan aplikasi baik dari sisi *analyst*, *programmer* maupun infrastruktur aplikasi maupun jaringan. Berikut adalah diagram-diagram UML pada pengembangan aplikasi:

1. *Use Case Diagram*

*Use case diagram* merupakan pemodelan sistem yang menggambarkan peran *user* dan bagaimana peran tersebut ketika menggunakan sistem. *Use case* *diagram* ini merepresentasikan interaksi antara *user* dengan sistem secara berurutan (Destriana et al., 2021). Simbol pada *use case* dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Notasi Use Case Diagram

|  |  |
| --- | --- |
| Simbol | Deskripsi |
| Actor | Menggambarkan aktor atau pengguna yang berperan berinteraksi dengan sistem |
| Use Case | Menggambarkan interaksi antara pengguna dengan sistem untuk mencapai tujuan tertentu |
| Association | Menggambarkan penghubung antara objek satu dengan objek yang lain |
| <<include>> | Menggambarkan bagaimana satu *use case* termasuk bagian aktivitas dari *use case* lain untuk menjalankan fungsinya |
| <<extend>> | Menggambarkan bagaimana dan kapan *use case* tambahan dapat dimasukkan ke dalam *use case* utama dalam kondisi tertentu |
| Generalization | Menggambarkan bagaimana hubungan objek anak (*descendant*) mewarisi semua karakteristik dari objek induk (*ancestor*) |

Sumber: (Destriana et al., 2021)

1. *Activity Diagram*

*Activity diagram* adalah pemodelan sistem yang menggambarkan bagaimana suatu sistem bekerja*. Activity diagram* merepresentasikan langkah-langkah atau aktivitas yang terjadi dalam suatu proses, dari awal hingga akhir. Dengan adanya *Activity diagram* dapat membantu memahami dan merencanakan bagaimana suatu sistem atau proses bekerja secara dinamis dan terstruktur (Destriana et al., 2021).

Berikut ini simbol-simbol yang terdapat pada *activity diagram* dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Notasi Activity Diagram

|  |  |
| --- | --- |
| Simbol | Deskripsi |
| Initial Node | Menggambarkan titik awal atau bagaimana suatu aktivitas dimulai |
| Final Node | Menggambarkan titik akhir atau bagaimana suatu aktivitas berakhir |
| Activity | Menggambarkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| Action | Menggambarkan suatu aktivitas yang dilakukan oleh sistem |
| Decision | Menggambarkan bagaimana sebuah pilihan atau keputusan dibuat untuk menentukan aliran aktivitas selanjutnya |
| Join Node | Menggambarkan bagaimana beberapa aliran aktivitas bergabung menjadi satu aliran |
| Fork Node | Menggambarkan bagaimana satu aliran aktivitas bercabang menjadi beberapa aliran yang dapat berjalan bersamaan atau paralel |
| Swim Line | Mengelompokkan aktivitas-aktivitas berdasarkan entitas atau aktor yang bertanggung jawab atas aktivitas di bagian tertentu |

Sumber: (Destriana et al., 2021)

1. *Sequence Diagram*

*Sequence diagram* adalah pemodelan sistem yang menggambarkan bagaimana urutan tindakan serta komunikasi antar objek atau komponen dalam suatu sistem, memperlihatkan siapa yang berkomunikasi dengan siapa, dan dalam urutan apa tindakan atau komunikasi itu terjadi (Destriana et al., 2021).

Berikut ini simbol-simbol yang terdapat pada *sequence diagram* dapat dilihat pada Tabel 2.5

Tabel 2. 5 Notasi Sequence Diagram

|  |  |
| --- | --- |
| Simbol | Deskripsi |
| Lifeline | Menggambarkan garis vertikal yang mewakili objek atau partisipan dalam interaksi |
| Object | Menggambarkan suatu objek yang ikut serta dalam interaksi |
| Time Active | Menggambarkan informasi waktu ketika objek melakukan proses atau menunggu proses dari objek lain |
| <<create>>    Pesan tipe create | Menggambarkan suatu *lifeline* membuat objek baru dalam sistem |
| nama\_metode()    Pesan tipe call | Menggambarkan bagaimana satu *lifeline* memanggil atau meminta suatu operasi atau metode dari *lifeline* lain |
| masukan    Pesan tipe send | Menggambarkan bagaimana satu *lifeline* mengirim pesan ke *lifeline* lain dalam urutan waktu yang ditentukan |
| keluaran    Pesan tipe return | Menggambarkan bagaimana suatu *lifeline* memberikan hasil kembali kepada *lifeline* yang mengirim pesan sebelumnya |
| Pesan tipe destroy | Menggambarkan bagaimana suatu *lifeline* menghapus atau mengakhiri objek dalam sistem |

Sumber: (A. O. C. Pratiwi, 2023)

1. *Class Diagram*

*Class diagram* adalah pemodelan sistem yang menggambarkan entitas-entitas atau objek-objek yang memiliki peran dalam sistem dimana setiap entitasnya memiliki atribut-atribut yang mendefinisikan karakteristik yang dimilikinya. *Class diagram* bersifat statis yang artinya diagram tidak menunjukkan urutan atau aliran proses dalam sistem melainkan fokus pada struktur dan hubungan antar entitas dalam sistem (Destriana et al., 2021).

Berikut adalah simbol-simbol yang terdapat pada *class diagram* yang dapat dilihat pada Tabel 2.6.

Tabel 2. 6 Notasi Class Diagram

|  |  |
| --- | --- |
| Simbol | Deskripsi |
| Class | Menggambarkan setiap class sebagai sebuah kotak yang dibagi menjadi 3 bagian untuk menunjukkan nama class, atribut-atributnya, dan *method-method*-nya |
| Association | Menggambarkan garis yang menghubungkan antar dua *class* dan dapat menunjukkan berbagai tipe dan aturan multiplisitas seperti *one-to-one, one-to-many*, atau *many-to-many* |
| Generalization | Menggambarkan relasi *inheritance* (pewarisan) antara dua *class* dimana satu *class* (*subclass*) mewarisi atribut dan metode dari *class* lain (*superclass*) |
| Dependency | Menggambarkan operasi pada suatu *class* yang menggunakan *class* yang lain |
| Aggregation | Menggambarkan relasi dimana satu *class* terdiri dari beberapa *class* yang berbeda |

Sumber: (Destriana et al., 2021)

1. **Confusion Matrix**

Co*nfusion matrix* adalah matriks yang digunakan untuk melakukan evaluasi proses model klasifikasi berupa jumlah data uji yang benar dan salah. Dengan adanya matriks ini dapat mengetahui kualitas kinerja model klasifikasi (Salsabila, 2022). *Confusion matrix* merupakan pengukuran yang sangat populer digunakan saat menyelesaikan masalah klasifikasi. Ini dapat diterapkan pada klasifikasi biner maupun untuk masalah klasifikasi multikelas. Pengukuran ini menunjukkan jumlah dari nilai prediksi dan nilai aktual (Kulkarni et al., 2020). Ilustrasi *confusion matrix* yang memuat 4 nilai dapat dilihat pada Tabel 2. 7.

Tabel 2. 7 Confusion Matrix

|  |  |  |
| --- | --- | --- |
| Nilai Prediksi | Nilai Aktual | |
| *Positive* (P) | *Negative* (N) |
| *Positive* | TP | FN |
| *Negative* | FP | TN |

Sumber:(Salsabila, 2022)

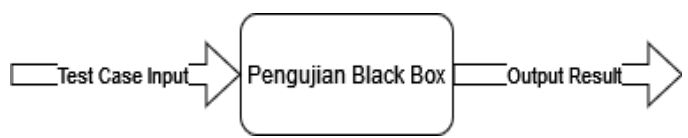
Menurut Zulfallah (2022), nilai prediksi adalah *positive* dan *negative*, sedangkan nilai aktual adalah *true* dan *false*. Berikut merupakan penjelasan *confusion matrix* dari Tabel 2.7:

1. TP (*True Positive*), menunjukkan prediksi positif dan benar.
2. TN (*True Negative*), menunjukkan prediksi negatif dan benar.
3. FP (*False Positive*), menunjukkan prediksi positif dan salah.
4. FN (*False Negative*), menunjukkan prediksi negatif dan salah.
5. **Blackbox Testing**

Menurut Soetam (2021), blackbox *testing* adalah metode pengujian perangkat lunak dimana fungsionalitas suatu aplikasi diuji tanpa melihat struktur internal kode program. Metode ini hanya memeriksa apakah aplikasi berfungsi sesuai dengan yang diharapkan ketika diberikan *input* tertentu dan apakah *output*-nya akurat tanpa mengetahui bagaimana aplikasi dapat memproses *input* tersebut di dalam kotak hitamnya seperti yang diilustrasikan pada Gambar 2.10. Metode pengujian ini juga dapat menemukan beberapa fungsi yang tidak benar atau tidak ada, kesalahan *interface*, kesalahan pada struktur data, kesalahan performance, kesalahan inisialisasi, dan terminasi Achmad & Yulfitri (2020).

*Blackbox testing* seperti yang dijelaskan oleh Febriyanti et al. (2021) merupakan sebagai salah satu cara sederhana untuk menguji sebuah program komputer dari segi spesifikasi fungsional untuk mengetahui apakah fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang diharapkan. Metode pengujian ini berguna untuk menemukan celah kesalahan yang ada pada sistem sehingga dapat diperbaiki sebelum digunakan oleh pengguna.

Dari beberapa pengertian *blackbox testing* di atas, dapat disimpulkan bahwa *black box testing* adalah metode pengujian perangkat lunak yang memfokuskan pada pengujian fungsionalitas aplikasi tanpa memperhatikan struktur internal atau kode programnya. Metode ini berguna untuk menemukan berbagai jenis kesalahan, termasuk kesalahan fungsi, *interface*, struktur data, performa, inisialisasi, dan terminasi, sehingga aplikasi dapat diperbaiki sebelum digunakan oleh pengguna.



Gambar 2. 10 Konsep Black Box Testing

Sumber: (Rachmatika, 2024)

## **Kajian Empiris**

Adapun beberapa penelitian terdahulu yang dapat dijadikan bahan perbandingan dan referensi dalam penerapan metode *Convolutional Neural Network* (CNN) dan model VGG-16 pada aplikasi *mobile* *android* untuk mendeteksi kanker kulit berbasis *android* antara lain:

Penelitian terkait dengan sistem klasifikasi citra menggunakan metode *Convolutional Neural Network* (CNN) diantaranya dilakukan oleh Umam Wiranda (2024) untuk mendeteksi mata katarak sejak dini. Dimana penelitian ini memperoleh hasil akurasi pengujian terbaik mencapai 93%. Hal ini menunjukkan bahwa hasil klasifikasi citra mata katarak dengan CNN dikatakan cukup baik. Hasil tersebut dapat menjadi indikasi pengembangan sistem deteksi penyakit mata katarak berbasis *mobile* sebagai solusi untuk memudahkan masyarakat dalam mencegah dan melakukan penanganan dini katarak. Perbedaan penelitian yang dilakukan oleh Umam Wiranda dengan penelitian ini yaitu terdapat pada pokok pembahasan penelitian yang akan dilakukan yaitu tentang deteksi kanker kulit.

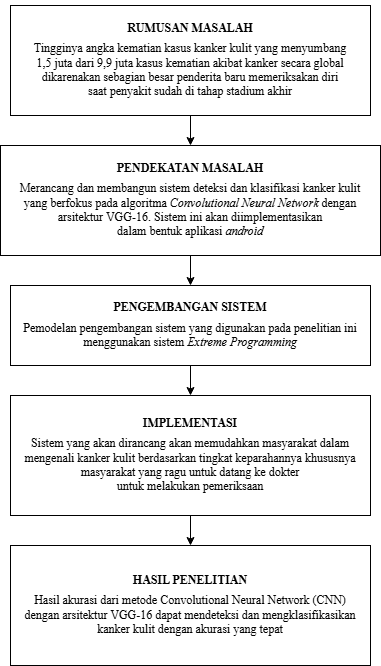
Penelitian yang dilakukan oleh Sa’idah et al. (2022) yaitu mengklasifikasikan kanker kulit menggunakan metode *Convolutional Neural Network* (CNN). Hasil yang didapatkan yaitu akurasi yang mencapai 97,73% dan *loss* 1,7063. Penelitian ini berfokus pada 2 kelas saja yaitu *benign* dan *malignant*. Penelitian Sa’idah et al. hanya sampai pada hasil akurasi dari klasifikasi sehingga disarankan untuk penelitian selanjutnya melakukan implementasi sistem ke dalam sebuah aplikasi dan menggunakan metode *deep learning* lainnya. Perbedaan penelitian Sa’idah et al. terdapat pada arsitektur CNN yaitu arsitektur GoogLeNet sedangkan pada penelitian ini menggunakan arsitektur VGGNet. Selain itu, penelitian ini memiliki 5 kelas untuk diklasifikasikan.

Adapun penelitian yang menggunakan metode serupa yaitu penelitian yang dilakukan oleh Yohannes & Al Rivan (2022) yaitu mengklasifikasikan citra jenis kanker kulit hanya memperoleh akurasi sebesar 65,33%. Perbedaan penelitian Yohannes & Al Rivan dengan penelitian ini yaitu penelitian Yohannes & Al Rivan menggunakan arsitektur VGG19 sedangkan penelitian ini menggunakan VGG16.

Penelitian lain yang dilakukan oleh Palupi Rini et al. (2024) menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur VGG16 untuk klasifikasi kanker payudara. Perbedaan antara penelitian Palupi Rini et al. dengan penelitian ini yaitu terdapat pada topik pembahasan. Dimana topik pembahasan ini membahas tentang kanker payudara hingga hasil akurasi klasifikasi. Sedangkan pada penelitian ini akan melakukan implementasi dalam bentuk aplikasi.

Berdasarkan uraian penjelasan diatas maka penelitian ini akan menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur VGG16 untuk melakukan klasifikasi gambar kanker kulit dengan menguji 5 kelas kanker kulit yaitu *actinic keratoses*, *basal cell carcinoma*, *benign keratosis*, *melanocytic nevi*, dan *melanoma*. Dengan dataset yang berjumlah 1857 gambar sehingga memungkinkan untuk dapat melakukan diagnosa penyakit kanker kulit lebih tepat dan akurat yang kemudian akan dilakukan implementasi sistem ke dalam aplikasi *mobile* *android*.

## **Kerangka Berpikir**



Gambar 2. 11 Kerangka Berpikir