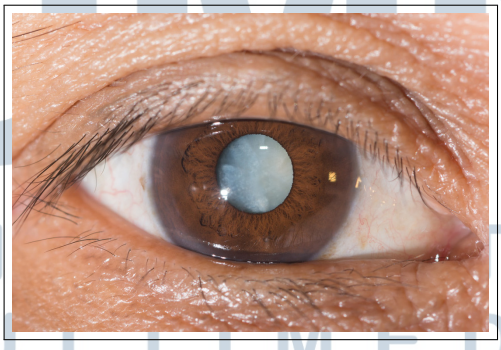
BAB 2
LANDASAN TEORI

Dalam penelitian ini, beberapa literatur dan teori pendukung digunakan untuk membangun model klasifikasi penyakit katarak menggunakan arsitektur ResNet50V2. Kajian literatur dilakukan untuk memahami teori yang relevan dengan penelitian ini. Teori-teori yang mendasari penelitian ini meliputi penyakit katarak, *deep learning*, CNN, ResNet50V2, dan *confusion matrix*.

2.1 Penyakit Katarak

Katarak merupakan kondisi degeneratif yang ditandai dengan kekeruhan pada lensa mata akibat reaksi biokimia yang memicu koagulasi protein, sehingga mengakibatkan penurunan kemampuan penglihatan hingga kebutaan [23]. Katarak dapat disebabkan oleh faktor penuaan, trauma mata, diabetes, hipertensi, genetik, kebiasaan merokok, konsumsi alkohol, dan paparan radiasi ultraviolet. Menurut penelitian [23], interaksi faktor-faktor tersebut, terutama diabetes dan hipertensi, dapat meningkatkan risiko kekeruhan lensa secara signifikan, bahkan pada usia produktif. Katarak ditandai dengan pandangan kabur seperti tertutup awan, silau terhadap cahaya, penglihatan ganda, sering mengganti kacamata atau lensa, dapat terjadi pada satu atau kedua mata tanpa menyebar, dan pada tahap lanjut bisa memicu peningkatan tekanan bola mata (glaukoma) [24].



Gambar 2.1. Mata katarak

Sumber: [25]

2.1.1 Perbedaan Mata Katarak dan Normal

Secara tampilan fisik, mata normal memiliki lensa yang jernih dan transparan tanpa ada kekeruhan apapun. Lensa mata yang normal adalah bagian mata yang transparan dan terletak di belakang pupil [10]. Sebaliknya, mata yang mengalami katarak menunjukkan perubahan fisik yang dapat diamati secara kasat mata berupa terlihatnya bercak putih abu-abu di tengah bola mata. Katarak dimulai dengan pemburaman dalam lensa, dan pada akhirnya transparansi lensa akan hilang [26]. Kekeruhan ini dapat berupa bercak-bercak kecil hingga kekeruhan menyeluruh pada lensa, memberikan tampilan keruh atau berkabut pada bagian yang seharusnya jernih dan transparan di mata normal. Perbedaan ini menandakan bahwa secara kasat mata, mata katarak menunjukkan perubahan fisik yang signifikan dibandingkan mata normal, terutama pada kejernihan dan warna lensa.

2.1.2 Pemeriksaan dan Penanganan

Pemeriksaan katarak meliputi evaluasi ketajaman penglihatan (visus), pemeriksaan *slit-lamp* untuk menilai derajat kekeruhan lensa, serta *shadow test* untuk menentukan tingkat maturitas katarak. Diagnosis katarak dapat ditegakkan melalui deteksi penurunan visus yang tidak membaik dengan koreksi refraksi, disertai temuan peningkatan opasitas pada lensa mata [27]. Penanganan katarak yang bersifat definitif hanya dapat dilakukan melalui tindakan pembedahan. Penatalaksanaan katarak melalui pembedahan hingga saat ini masih menjadi satu-satunya pendekatan kuratif [13, 28].

2.2 Deep Learning

Deep learning adalah cabang dari *machine learning* yang menekankan penggunaan jaringan saraf tiruan berlapis-lapis untuk memodelkan representasi data secara hierarkis. Salah satu kelebihan *Deep Learning* terletak pada kemampuannya menangani permasalahan yang kompleks, seperti pengenalan gambar dan suara, serta meniru proses kerja otak manusia melalui penggunaan jaringan saraf tiruan dalam algoritmanya [29]. Salah satu kekuatan utama *deep learning* adalah kemampuannya dalam mengekstrak pola secara otomatis tanpa perlu perancangan fitur secara eksplisit oleh manusia, yang menjadikannya sangat efisien dalam menangani data besar dan kompleks [30].

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu arsitektur *deep learning* yang dirancang khusus untuk memproses data *grid-like*, seperti gambar, dengan cara meniru cara kerja visual manusia dalam mengenali pola spasial dan hubungan lokal dalam data. CNN terdiri dari lapisan konvolusi, *pooling*, dan *fully connected* yang memungkinkan sistem belajar fitur penting dari data gambar secara otomatis, tanpa membutuhkan proses ekstraksi fitur secara manual [25].

Salah satu keunggulan CNN adalah kemampuannya dalam mengurangi kompleksitas data secara bertahap dengan mempertahankan informasi penting melalui operasi konvolusi dan *pooling*, sehingga sangat efektif untuk tugas seperti klasifikasi citra dan deteksi objek [31]. Struktur dasar CNN mencakup tiga komponen utama yaitu, *convolutional layer*, *pooling layer*, dan *fully connected layer* [32].

2.3.1 Convolutional Layer

Convolutional layer merupakan komponen inti dalam struktur dasar CNN yang berfungsi untuk mengekstraksi fitur spasial dari data input, seperti gambar atau sinyal. Cara kerjanya melibatkan proses konvolusi, di mana kernel (filter) bergerak melintasi input untuk menghasilkan peta fitur (*feature map*) melalui operasi *dot product* antara nilai-nilai piksel input dan bobot *kernel*. Filter ini belajar mendeteksi pola-pola lokal seperti tepi, sudut, atau tekstur.

Setiap filter dalam satu lapisan dirancang untuk mengekstraksi fitur berbeda, dan hasil konvolusi biasanya dilanjutkan dengan fungsi aktivasi seperti ReLU untuk memperkenalkan *non-linearitas*. Struktur berlapis konvolusi dalam CNN memungkinkan model secara bertahap mengenali fitur kompleks dari data visual, dimulai dari fitur dasar hingga representasi yang lebih abstrak, yang krusial untuk tugas-tugas seperti klasifikasi objek dan segmentasi gambar [33].

2.3.2 Pooling layer

Pooling layer dalam CNN berfungsi sebagai mekanisme reduksi dimensi spasial dari data fitur hasil konvolusi, sekaligus mempertahankan informasi penting yang telah dipelajari. Cara kerjanya dilakukan dengan menerapkan fungsi seperti *max pooling* atau *average pooling* pada area lokal dari *feature map*, biasanya

dengan jendela berukuran 2×2 atau 3×3 untuk memilih nilai maksimum atau rata-rata dari jendela tersebut. Operasi ini membantu mengurangi ukuran data, jumlah parameter, dan risiko *overfitting*, serta meningkatkan efisiensi komputasi. *Pooling layer* bekerja sebagai mekanisme *downsampling* yang efektif dalam membangun representasi hierarkis fitur dalam jaringan CNN, memungkinkan model untuk mengenali pola dengan lebih efisien dan tahan terhadap variasi spasial kecil dalam input [34].

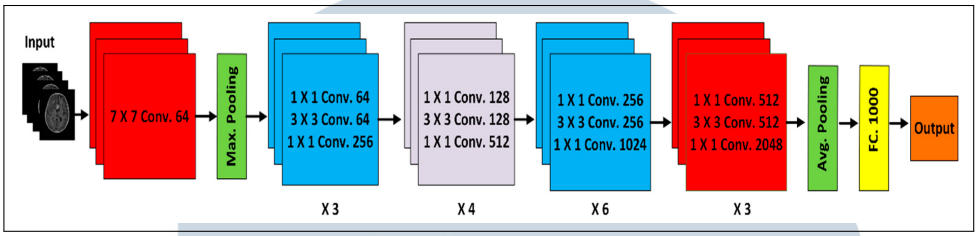
2.3.3 Fully Connected Layer

Fully Connected Layer (FCL) merupakan tahap akhir dalam arsitektur CNN yang berfungsi mengintegrasikan dan menerjemahkan fitur-fitur spasial yang telah diekstraksi sebelumnya menjadi keputusan klasifikasi. FCL bertindak sebagai pemetaan akhir dari fitur non-linear tinggi ke dalam ruang keputusan, dan secara operasional bekerja seperti *multilayer perceptron* dalam jaringan saraf klasik [35]. Dalam cara kerjanya, hasil dari layer konvolusi dan *pooling* diratakan (*flattened*) menjadi vektor satu dimensi, kemudian setiap neuron di FCL terhubung dengan semua neuron di layer sebelumnya. Setiap koneksi ini memiliki bobot dan bias yang dipelajari selama pelatihan menggunakan algoritma *backpropagation*. Aktivasi dari tiap neuron dihitung sebagai kombinasi linier dari input, yang dilanjutkan dengan fungsi aktivasi seperti ReLU atau Softmax, tergantung pada tujuan klasifikasi.

2.4 ResNet50V2

ResNet50V2 (Residual Network) adalah arsitektur Convolutional Neural Network (CNN) yang diperkenalkan dalam penelitian oleh Kaiming He, Xiangyu Zhang, Shaoqing Ren, dan Jian Sun pada tahun 2016 [36]. ResNet50V2 memiliki 190 lapisan dan 25.613.800 parameter dalam arsitektur jaringannya [37]. ResNet50V2 merupakan pengembangan dari versi sebelumnya, ResNet50, dengan angka ”50” menunjukkan jumlah *weighted layers* atau kedalaman jaringan. Perbedaan utama antara ResNet50 dan ResNet50V2 terletak pada desain residual block-nya. ResNet50 menggunakan pendekatan *post-activation*, di mana aktivasi seperti ReLU dilakukan setelah penjumlahan *shortcut* dan *output* konvolusi. Sebaliknya, ResNet50V2 mengadopsi *pre-activation*, yaitu aktivasi dan normalisasi dilakukan sebelum konvolusi. Perubahan ini membuat aliran gradien lebih stabil dan mempermudah pelatihan jaringan yang sangat dalam, karena setiap blok

memiliki struktur yang lebih konsisten dan memfasilitasi koneksi yang lebih baik antar lapisan.



Gambar 2.2. Arsitektur ResNet50V2

Sumber: [38]

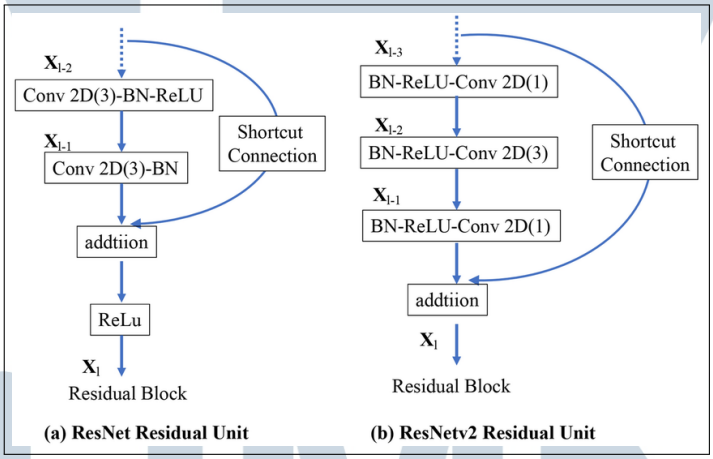
Gambar 2.2 menunjukkan arsitektur umum dari ResNet50V2. Jaringan ini diawali dengan lapisan konvolusi awal berukuran 7×7 dengan 64 filter, dilanjutkan oleh proses *max pooling* untuk mereduksi dimensi fitur. Setelah itu, data melewati empat tahap blok *residual* yang masing-masing terdiri dari beberapa lapisan konvolusi berurutan (1×1, 3×3, dan 1×1) dalam konfigurasi bottleneck. Keempat blok tersebut masing-masing diulang sebanyak tiga, empat, enam, dan tiga kali, menghasilkan total 16 residual blocks. Karena setiap blok terdiri dari tiga lapisan konvolusi berparameter (*weighted layers*), maka jumlah total lapisan konvolusi utama adalah $3 \times (3+4+6+3) = 48$, ditambah 1 lapisan konvolusi awal dan satu lapisan *fully connected* (FC) di bagian akhir, sehingga totalnya menjadi 50 *weighted layers*. Inilah yang menjadi dasar penamaan ResNet50V2.

Meskipun dinamakan ResNet50V2, jumlah total lapisan dalam arsitektur ini bukan hanya 50. Jika dihitung secara keseluruhan, termasuk lapisan-lapisan *non-trainable* seperti *Batch Normalization*, *ReLU*, dan *Add*, maka jumlah total lapisan jaringan mencapai 190 lapisan [37]. Perbedaan ini muncul karena perhitungan jumlah lapisan total mencakup semua komponen fungsional dalam jaringan, tidak hanya lapisan yang memiliki parameter. Setelah blok *residual*, terdapat lapisan *average pooling* dan *FC layer* yang menghasilkan *output* klasifikasi akhir.

2.4.1 Perbedaan ResNet50V1 dan ResNet50V2

Gambar 2.3 menunjukkan perbandingan antara *ResNet Residual Unit* (ResNet V1) dan *ResNetV2 Residual Unit* (ResNet V2). Perbedaan utama antara kedua arsitektur ini terletak pada urutan operasi normalisasi, aktivasi, dan konvolusi di dalam blok *residual*.

- **ResNet V1** (Gambar 2.3(a)) menggunakan pendekatan *post-activation*, yaitu aktivasi ReLU diterapkan setelah operasi penjumlahan (*addition*) antara input (X) dengan hasil transformasi $F(X)$. Dalam blok ini, urutan operasinya adalah *Conv2D* $\rightarrow$ *Batch Normalization* $\rightarrow$ *ReLU*, kemudian *Conv2D* $\rightarrow$ *Batch Normalization*, dan setelah penjumlahan barulah dilakukan ReLU terakhir.
- **ResNet V2** (Gambar 2.3(b)) mengadopsi pendekatan *pre-activation*, di mana *Batch Normalization* dan *ReLU* dilakukan sebelum operasi konvolusi. Blok residual ini terdiri dari tiga lapisan konvolusi (1×1, 3×3, dan 1×1), masing-masing diawali oleh BN dan ReLU. Setelah seluruh transformasi selesai, hasilnya dijumlahkan dengan *shortcut* tanpa aktivasi tambahan setelahnya.



Gambar 2.3. Perbedaan ResNetV1 dan V2
Sumber: [39]

Pendekatan *pre-activation* pada ResNetV2 meningkatkan stabilitas pelatihan, khususnya pada jaringan yang sangat dalam, karena memfasilitasi propagasi gradien yang lebih baik melalui jalur *shortcut*.

2.5 Fungsi Aktivasi

Fungsi aktivasi merupakan elemen penting dalam jaringan saraf tiruan karena memungkinkan jaringan untuk menangani persoalan non-linear secara efektif. Tanpa fungsi aktivasi, jaringan saraf hanya akan mampu melakukan transformasi linier, yang membatasi kemampuannya dalam menyelesaikan tugas-tugas kompleks seperti klasifikasi citra, pemrosesan bahasa alami, dan prediksi

deret waktu. Fungsi aktivasi bekerja dengan mengubah hasil penjumlahan bobot dan bias menjadi nilai *output* yang dapat diteruskan ke neuron berikutnya. Bentuk umum dari fungsi ini mencakup Sigmoid, Tanh, ReLU, dan varian lainnya. Fungsi aktivasi berfungsi sebagai pemetaan dari input menjadi *output* yang sesuai dalam konteks pembelajaran berlapis pada jaringan saraf, memungkinkan model membentuk representasi fitur yang lebih mendalam dan akurat terhadap data [40].

2.5.1 Sigmoid

Fungsi aktivasi Sigmoid merupakan salah satu fungsi non-linear yang paling klasik dan banyak digunakan pada tahap awal pengembangan jaringan saraf. Fungsi ini mengubah input numerik menjadi nilai antara 0 dan 1, sehingga sangat cocok digunakan dalam kasus klasifikasi biner. Dalam konteks *backpropagation* dan pelatihan jaringan, Sigmoid membantu dalam menyebarkan gradien melalui jaringan, meskipun juga diketahui rawan terhadap masalah *vanishing gradient* pada jaringan yang dalam. Sigmoid sering digunakan untuk menyederhanakan representasi data dalam rentang probabilistik, sehingga berguna dalam menginterpretasi hasil *output* sebagai kemungkinan prediksi [41]. Selain itu, Sigmoid memiliki bentuk kurva halus dan kontinu yang membuatnya cocok untuk diferensiasi, yang esensial dalam proses pelatihan menggunakan algoritma turun gradien.

2.5.2 Rectified Linear Unit

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang paling umum digunakan dalam jaringan saraf dalam (deep neural networks) karena keefektifannya dalam mengatasi permasalahan *vanishing gradient* yang sering terjadi pada fungsi aktivasi tradisional seperti Sigmoid dan Tanh [42]. ReLU bekerja dengan prinsip yang sederhana namun efisien, yaitu mengubah semua nilai masukan yang bernilai nol atau negatif menjadi nol, sementara nilai masukan yang positif dipertahankan sebagaimana adanya [43]. Mekanisme ini tidak hanya mempercepat proses komputasi karena bersifat linear terhadap nilai positif, tetapi juga membantu mempertahankan sinyal penting dalam proses pelatihan model. Dalam konteks pengolahan citra menggunakan jaringan konvolusional, ReLU berperan penting dalam menentukan nilai akhir dari hasil konvolusi, di mana hanya fitur-fitur yang memiliki kontribusi positif yang diteruskan ke lapisan berikutnya, sementara nilai

negatif yang dianggap tidak relevan dieliminasi.

2.6 Optimizer

Optimizer adalah algoritma yang digunakan dalam pelatihan jaringan saraf untuk menyesuaikan bobot model berdasarkan perhitungan gradien dari fungsi *loss*, dengan tujuan meminimalkan kesalahan prediksi. Proses ini bekerja secara iteratif, di mana setiap parameter model diperbarui melalui teknik optimisasi berbasis gradien agar model konvergen ke minimum global atau lokal dari fungsi *loss*. *Optimizers* memainkan peran fundamental dalam mempercepat konvergensi dan meningkatkan performa model, dengan mekanisme perhitungan terarah yang mempertimbangkan perubahan parameter secara efisien di ruang pencarian [44].

2.6.1 Root Mean Square Propagation

RMSProp (Root Mean Square Propagation) adalah algoritma optimisasi yang mengatur *learning rate* untuk setiap parameter dengan membagi gradien dengan rata-rata bergerak dari kuadrat gradien sebelumnya. Teknik ini menjaga ukuran *update* tetap stabil dan menghindari osilasi pada gradien besar. RMSProp sangat cocok untuk jaringan rekursif atau arsitektur yang menghadapi gradien eksplosif, karena algoritma ini menstabilkan pembelajaran melalui normalisasi gradien lokal [45].

2.6.2 Adaptive Moment Estimation

Adam (Adaptive Moment Estimation) merupakan algoritma optimisasi yang menggabungkan keunggulan dari dua teknik sebelumnya, momentum dan RMSProp. Adam menyimpan estimasi rata-rata pertama (*mean*) dan kedua (*variance*) dari gradien, sehingga mampu melakukan penyesuaian bobot yang lebih adaptif terhadap dinamika *loss function*. Adam telah terbukti sangat efisien dalam pelatihan jaringan dalam berbagai domain karena kemampuannya menyesuaikan laju belajar (learning rate) setiap parameter berdasarkan sejarah gradien, menjadikannya pilihan *default* dalam banyak aplikasi *deep learning* [46].

2.6.3 Stochastic Gradient Descent

SGD adalah algoritma optimisasi paling dasar yang bekerja dengan memperbarui bobot model berdasarkan *subset* acak dari data (*mini-batch*). Pendekatan ini membuat proses pelatihan lebih cepat dan memungkinkan keluar dari *local minima*. Meskipun sederhana, SGD memiliki kelemahan dalam hal konvergensi yang lambat dan sensitif terhadap pengaturan *learning rate*. SGD tetap relevan terutama saat digunakan dengan teknik tambahan seperti momentum atau *decay*, yang mampu meningkatkan stabilitas dan kecepatan pelatihan [47].

2.7 Confusion Matrix

Confusion matrix adalah alat evaluasi yang menampilkan perbandingan antara hasil prediksi model dengan data aktual dalam bentuk matriks. Matriks ini mengkategorikan hasil ke dalam empat kelas: *true positive*, *true negative*, *false positive*, dan *false negative*. Dengan informasi ini, dapat dilihat secara jelas seberapa banyak prediksi yang benar dan salah, serta menghitung metrik evaluasi tambahan untuk menilai performa model secara lebih mendalam. Hasil dari *confusion matrix* dapat digunakan untuk menghitung beberapa metrik, seperti *accuracy*, *precision*, *recall*, dan *f1-score*.

- True Positive (TP): Kondisi ketika model melakukan prediksi positif (kelas yang ditargetkan) dan prediksinya sesuai dengan fakta. Misalnya, dalam deteksi penyakit melalui gambar medis, TP terjadi jika model mengidentifikasi adanya penyakit dan hasil diagnosis tersebut akurat.
- True Negative (TN): Kondisi ketika model memprediksi negatif (kelas non-target) dan prediksinya benar. Contohnya, jika model menyimpulkan tidak ada penyakit pada gambar medis dan kenyataannya pasien sehat, maka itu termasuk TN.
- False Positive (FP): Kesalahan prediksi di mana model mengklasifikasikan suatu kasus sebagai positif, padahal sebenarnya negatif. Dalam konteks medis, FP terjadi jika model mendeteksi penyakit, tetapi pasien sebenarnya tidak mengidap penyakit tersebut. Kesalahan ini sering disebut sebagai kesalahan tipe I.
- False Negative (FN): Kesalahan prediksi di mana model mengabaikan atau tidak mendeteksi kasus positif. Misalnya, model menyatakan pasien sehat,

padahal sebenarnya menderita penyakit. Kasus ini dikenal sebagai kesalahan tipe II.

2.7.1 Accuracy

Accuracy atau akurasi adalah ukuran yang menunjukkan proporsi prediksi yang tepat, yaitu jumlah *True Positive* dan *True Negative*, dibandingkan dengan keseluruhan prediksi yang dilakukan, termasuk *False Positive* dan *False Negative*. Cara perhitungan *accuracy* ditunjukkan pada Persamaan 2.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2.7.2 Precision

Precision atau presisi menggambarkan seberapa tepat model dalam mengklasifikasikan data positif. Nilai *precision* yang tinggi menunjukkan bahwa sedikit data negatif yang salah diklasifikasikan sebagai positif (*false positive* rendah). Perhitungan *precision* dapat dilihat pada Persamaan 2.2.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

2.7.3 Recall

Recall, juga dikenal sebagai sensitivitas, menilai sejauh mana model mampu mengidentifikasi data positif secara benar. Nilai *recall* yang tinggi berarti model menghasilkan sedikit *false negative*. Persamaan 2.3 menyajikan rumus untuk menghitung *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

2.7.4 F1-score

F1-score merupakan metrik gabungan antara *precision* dan *recall*, yang dihitung sebagai rata-rata harmonik dari keduanya. Metrik ini berguna ketika diperlukan keseimbangan antara presisi dan sensitivitas. Nilai *F1-score* berada dalam rentang 0 hingga 1, dengan nilai mendekati 1 menunjukkan performa model yang semakin baik. Persamaan 2.4 menunjukkan cara perhitungannya.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.4}$$