BAB 2
LANDASAN TEORI

2.1 Hama Blast

Penyakit atau hama blast yang terdapat pada daun tanaman padi disebabkan oleh jamur *Pyricularia oryzae Cav* [4]. Di Indonesia, penyakit blas sudah menyebar di hampir semua sentra produksi padi. Jamur patogen P. grisea mampu menyerang tanaman padi pada berbagai stadia pertumbuhan dari benih sampai fase pertumbuhan malai (generatif). Pada tanaman stadium vegetatif biasanya patogen menginfeksi bagian daun, disebut blas daun (*leaf blast*). Perkembangan penyakit blas madalah sebagai berikut, bentuk khas dari bercak blas adalah elips dengan ujungnya agak runcing seperti belah ketupat [3]. Bercak yang telah berkembang, bagian tepi berwarna coklat dan bagian tengah berwarna putih keabu-abuan. Bentuk dan warna bercak bervariasi tergantung pada keadaan sekitarnya, kerentanan varietas, dan umur bercak. Bercak bermula kecil berwarna hijau gelap, abu-abu sedikit kebiru-biruan. Bercak ini terus membesar pada varietas yang peka, khususnya bila dalam keadaan lembab. Bercak yang telah berkembang penuh mencapai 1-1.5 cm dan lebar 0.3-0.5 cm dengan tepi berwarna coklat.

Menurut Penelitian Suganda Tarkus dkk jika tanaman padi terkena blas dengan intensitas yang tinggi dapat mengurangi hasil panen hingga 61% dan kemungkinan terparahnya dapat menyebabkan gagal panen. Oleh karena itu pendeteksian penyakit blast sangatlah penting sebagai kunci pengendalian hama blast. Sehingga petani dapat melakukan langkah selanjutnya ialah memberikan perawatan terhadap tanaman yang terkena penyakit blast. Karena itu untuk mendeteksi adanya penyakit blast pada daun padi diperlukan teknologi *Computer Vision*.

Gambar 2.1 ***Hama Blast***

2.2 Hama Blight

Penyakit hawar daun bakteri (HDB) atau dikenal dengan nama lain blight merupakan salah satu penyakit penting tanaman padi di negara-negara penghasil padi, termasuk di Indonesia. Penyakit ini disebabkan oleh bakteri Xanthomonas oryzae pv. oryzae (Xoo). Patogen ini menginfeksi daun padi pada semua fase pertumbuhan tanaman, mulai dari pesemaian sampai menjelang panen. Gejala yang timbul pada tanaman fase vegetatif disebut kresek dan pada fase generatif disebut hawar. Apabila infeksi terjadi pada fase generatif mengakibatkan proses pengisian gabah menjadi kurang sempurna. Kehilangan hasil karena penyakit HDB bervariasi antara 15-80%, bergantung pada stadia tanaman saat penyakit timbul. Perkembangan penyakit HDB dipengaruhi oleh lingkungan terutama kelembapan, suhu, cara budi daya, varietas, dan pemupukan nitrogen.

Pada tanaman dewasa umur lebih dari 4 minggu setelah tanam, penyakit HDB menimbulkan gejala hawar (blight). Gejala diawali berupa bercak kebasahan berwarna keabu-abuan pada satu atau kedua sisi daun, biasanya dimulai dari pucuk daun atau beberapa sentimeter dari pucuk daun [6]. Bercak ini kemudian berkembang meluas ke ujung dan pangkal daun dan melebar. Bagian daun yang terinfeksi berwarna hijau keabu-abuan dan agak menggulung, kemudian mongering dan berwarna abu-abu keputihan. Pada tanaman yang rentan, gejala ini terus

berkembang hingga seluruh daun menjadi kering dan kadang-kadang sampai pelepah.

Gambar 2.2 *Hama Blight*

2.3 Hama Tungro

Tungro merupakan salah satu penyakit penting pada tanaman padi di Asia tropis yang bersifat endemis. Tungro disebabkan oleh interaksi antara dua virus yang berbeda, Rice tungro bachiliform virus (RTBV) dan Rice tungro spherical virus (RTSV) yang keduanya ditularkan oleh wereng hijau [7].Tanaman padi yang mengidap penyakit tungro menunjukkan gejala kerdil, perubahan warna daun menjadi jingga kemerah- merahan, anakan berkurang, dan malai tidak sempurna.

Gejala utama penyakit tungro tampak pada perubahan warna pada daun muda menjadi kuning oranye dimulai dari ujung daun, daun muda menggulung, jumlah anakan berkurang, tanaman kerdil dan pertumbuhannya ter-hambat [8]. Bila serangan berat terdapat bintik-bintik hitam pada daun. Gejala penyakit tersebar mengelompok, hamparan tanaman padi terlihat seperti bergelombang karena adanya perbedaan tinggi tanaman antara tanaman sehat dan yang terinfeksi. Intensitas serangan bergantung pada tingkat ketahanan varietas padi dan umur tanaman pada saat terinfeksi. Tanaman muda lebih peka ter-hadap infeksi dibanding tanaman tua. Gejala pertama pada umumnya timbul 6-15 hari setelah terinfeksi.

Gambar 2.3 *Hama Tungro*

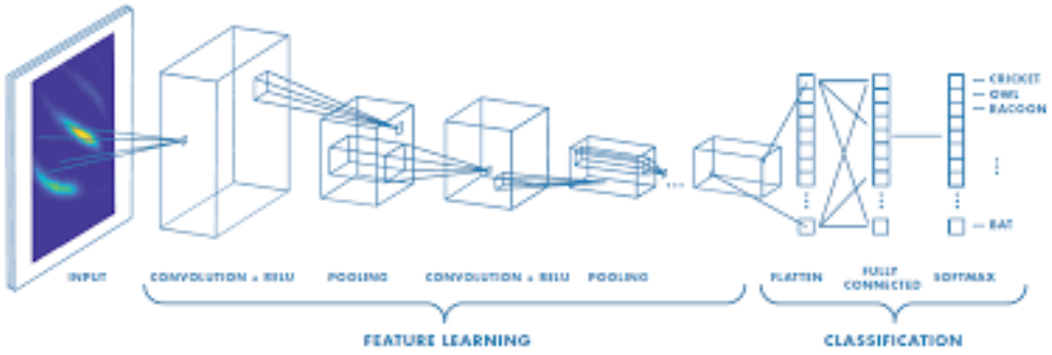
2.4 Computer Vision

Computer Vision merupakan teknologi yang membuat komputer dapat melihat, mendeteksi dan memproses gambar layaknya penglihatan manusia, kemudian komputer akan menampilkan hasil yang sesuai dengan input yang di berikan [2]. Computer Vision membahas bagaimana langkah suatu sistem dibangun untuk mendapatkan pemaham tingkat tinggi dari suatu inputan berupa gambar maupun video. Computer vision berhubungan dengan ekstraksi otomatis, analisa, dan pemahaman informasi yang berguna dari satu gambar atau urutan gambar [9]. Computer Vision dapat di implementasikan pada bagian pengawasan, mendeteksi objek, menghitung objek, hingga pengenalan suatu objek. Computer vision menggabungkan kamera, komputasi awan, perangkat lunak, serta AI (kecerdasan buatan) sehingga sistem dapat melakukan penghilatan dan mengidentifikasi suatu objek. Computer vision menggunakan deep learning (pembelajaran mendalam) berfungsi membuat jaringan neural atau jaringan saraf yang mengarahkan sistem dalam pemrosesan serta analisis. Computer vision yang telah selesai dilatih atau terlatih dapat mengenali objek, melihat pergerakan, bahkan mengenali objek.

Dalam computer vision algoritma CNN sangat terkenal dan banyak digemari karena bagus dan memiliki tingkat akurasi yang tinggi.

2.5 Algoritma CNN

Convolutional Neural Network atau kerap disingkat CNN merupakan algoritma neural network yang sangat terkenal dan banyak digemari. Convolutional Neural Network merupakan peningkatan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dengan ukuran dua dimensi. Proses kerja CNN (Convolutional Neural Network) hampir sama dengan neural network (jaringan saraf) pada umumnya, namun terdapat perbedaan yang paling utama yaitu pada algoritma CNN (Convolutional Neural Network) menggunakan kernel 2 dimensi atau dimensi tinggi pada tiap unit dalam lapisan CNN yang akan dilakukan konvolusi [10]. Pada klasifikasi citra, MLP kurang efektif karena tidak menyimpan informasi spasial dari data citra dengan menganggap setiap piksel adalah fitur yang berdiri sendiri sehingga menghasilkan hasil yang kurang maksimal. Kernel dalam CNN digunakan untuk menggabungkan fitur spasial dengan bentuk spasial yang menyerupai media input.

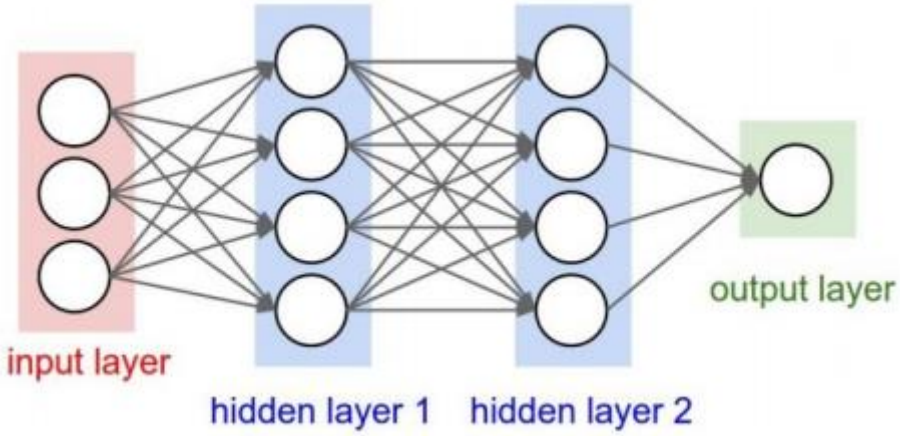


Gambar 2.4 ***Alur proses CNN dalam mengolah citra***

2.5.1 Konsep Kerja CNN

Pada dasarnya konsep Kerja algoritma CNN (Convolutional Neural Network) memiliki kemiripan dengan MLP (Multilayer Perceptron), tetapi perbedaan mendasarnya yaitu dalam algoritma CNN memiliki neuron yang dimana

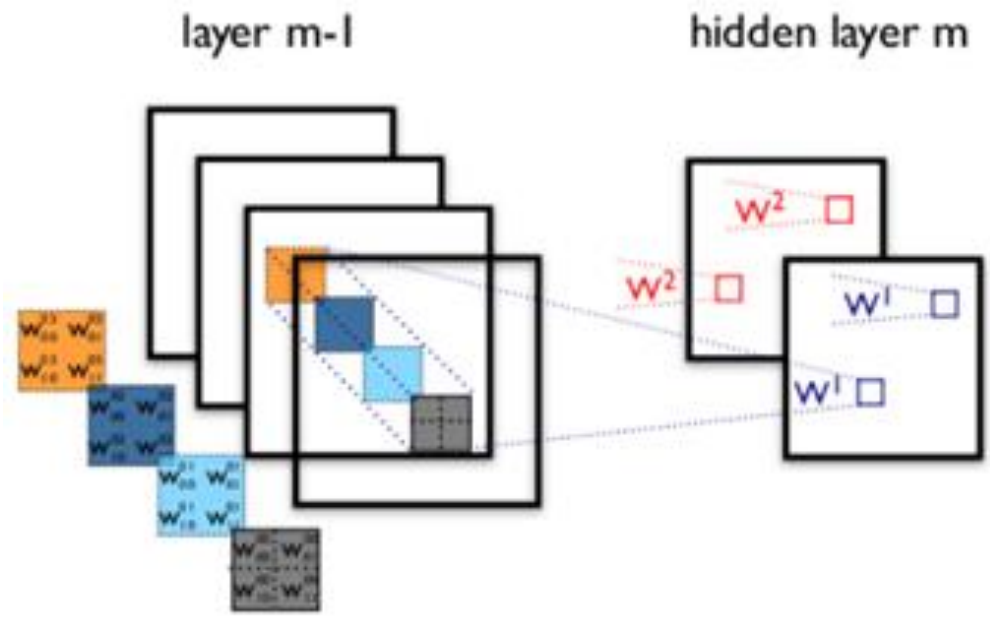
setiap neuron dipresentasikan dalam bentuk dua dimensi [11]. Berikut gambaran arsitektur MLP:



Gambar 2.5 ***Arstiketur MLP Sederhana***

Pada Gambar diatas merupakan gambaran sebuah MLP sederhana dimana memiliki i layer yang direpresentasikan dengan sebuah kotak berwarna merah dan biru dengan setiap layer berisi ji neuron yang direpresentasikan dengan lingkaran putih. Dalam MLP data inputan yang diterima berupa data satu dimensi dan dipropagasikan pada jaringan hingga menghasilkan output. Tiap-tiap hubungan antar neuron pada dua layer yang bersampingan memiliki parameter bobot satu dimensi yang menetapkan kualitas model. Pada tiap data inputan pada layer dilangsungkan sebuah operasi linear dengan nilai bobot yang telah ada, kemudian hasil perthitungan akan ditransformasi memanfaatkan operasi non linear yang disebut sebagai fungsi aktivasi.

Dalam CNN, proses propagasi terhadap data pada jaringan merupakan data dua dimensi, sehingga berimbas pada operasi linear dan parameter bobot pada algortima CNN berbeda. Didalam algoritma CNN proses operasi linear memanfaatkan operasi konvolusi, sehingga bobot tidak lagi berbentuk satu dimensi, akan tetapi berbentuk empat dimensi yang tidak lain gabungan kernel konvolusi [12]. Berikut gambar dimensi bobot pada CNN.



Gambar 2.6 ***Proses Konvolusi pada CNN***

Dikarenakan proses operasi linear pada algoritma CNN bersifat konvolusi, maka algoritma CNN hanya bisa digunakan pada data yang memiliki bentuk dua dimensi layaknya gambar dan video.

2.5.2 Arsitektur CNN

Pada dasarnya CNN memiliki lima elemen layer utama yakni sebagai berikut:

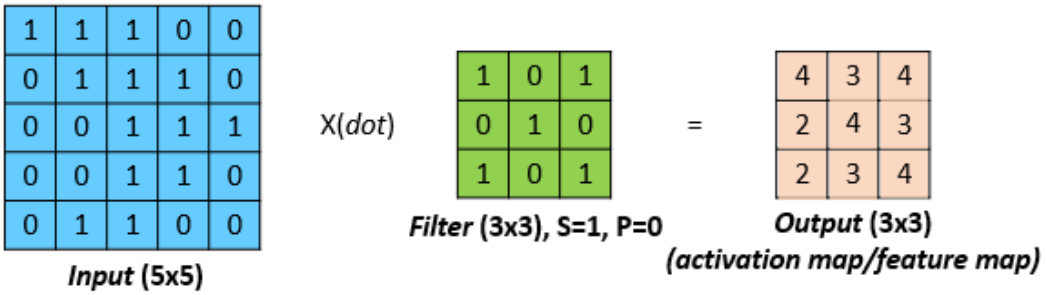
a. Lapisan Masukkan (Input Layer)

Lapisan ini merupakan semua data yang diperoleh dari gambar pada data masukkan. *Input Layer* yakni semua node piksel pada gambar. Untuk citra dengan ukuran 300 x 300 piksel dengan 3 channel warna RGB (Red, Green, dan Blue) sehingga terdapat 90.000 nodes terdapat pada input layer. Tetapi jika melihat keterangan pada arstitektur tersebut terdapat 270.000.0000 yang diperoleh dari 90.000 node dari piksel asli yang selanjutnya dikalikan dengan 3 channel warna sebelumnya.

b. Lapisan Konvolusi (Convolution Layer)

Lapisan Konvolusi merupakan lapisan pusat dari CNN [13]. Pada lapisan Convolution Layer memiliki hasil citra baru yang menunjukkan fitur dari citra

masukan. Pada proses tersebut, Convolution Layer mengandalkan filter pada tiap-tiap citra yang menjadi inputan. Convolution Layer merupakan kumpulan neuoron yang memiliki susunan sedemikian rupa sehingga terbentuk sebuah fikter dengan memiliki panjang dan tinggi dalam satuan piksel. Pada Algoritma CNN mempunyai beberapa filter yang di gambarkan sebagai kedalaman atau jumlah dari Convolutional Layer. Pada saat operasi berlangsung, filter akan bergeser kesemua bagian dari gambar. Pada saat terjadi pergeseran, tiap pergesaran akan berlangsung operasi "dot" antara input (masukkan) dan value dari filter tersebut hingga membuahkan sebuah output (keluaran), biasa disebut juga sebagai feature map atau activation map. Proses konvolusi ini dengan memanfaatkan filter pada layer akan membuahkan feature map yang nantinya akan dimanfaatkan pada lapisan aktivasi (activation layer). Berikut gambaran Alur pada proses convolution layer pada gambar 2.7 dibawah ini



Gambar 2.7 ***Proses pada Convolution Layer***

Pada Convolutional Layer ini terdapat tiga hyperparameter yang difungsikan untuk pengaturan ukuran volume output neuron yakni kedalaman (depth), langkah (stride), dan zero-padding. Tujuannya dilakukan konvolusi pada data gambar yaitu untuk mengekstraksi fitur dari gambar input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada lapisan tersebut mengspesifikasi kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN. Kemudian dimensi output dari feature maps dapat dihitung menggunakan rumus berikut:

$$\text{Output f.maps} = \frac{W-N+2P}{S} + 1$$

Keterangan:

W = ukuran masukkan

N = ukuran filter

P = zero padding

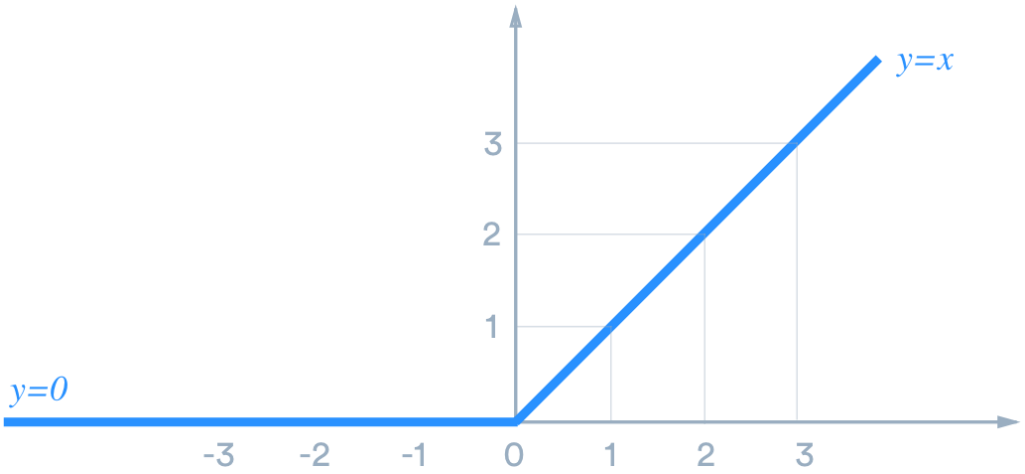
S = stride

C. Lapisan Aktivasi (Activation Layer)

Pada tahap selanjutnya adalah tahap activation layer, dimana dilakukan perhitungan nilai dengan Activation Function yang digunakan untuk mencari nilai non-linier pada nilai hasil konvolusi. Secara umum, Activation function dapat dibagi menjadi 2 tipe yaitu fungsi aktivasi linier dan fungsi aktivasi non linier. Ada beberapa activation function yang akan kita temukan dalam prakteknya, diantaranya Sigmoid, Tanh, ReLU, dan softmax. Untuk penggunaan activation lebih banyak pada ReLU karena sangat mempercepat proses konvergensi yang dilakukan dengan stochastic gradient descent jika dibandingkan dengan sigmoid / tanh. Pada penelitian kali hanya akan menggunakan Aktivasi ReLu dan Softmax.

- Fungsi Aktivasi ReLU

Rectified linear unit atau yang dikenal juga sebagai *ReLU*, adalah pendeketan paling umum dan dasar untuk mengenalkan non-linearitas ke dalam jaringan saraf atau *neural network*. Fungsi ReLU terdiri hanya max (0, x). Pada proses fungsi ReLU (Rectified Linear Unit) melakukan “treshold” dari 0 hingga angka tak terbatas. Fungsi ReLU ini merupakan salah satu fungsi yang banyak digunakan dan digemari pada saat ini. Berikut ini grafik fungsi aktivasi ReLU:



Gambar 2.8 ***Aktivasi ReLu***

Pada fungsi *ReLU* nilai inputan dari neuron-neuron berbentuk bilangan negatif, melalui prosesnya fungsi ini akan melakukan penerjemahan nilai tersebut kedalam nilai 0, dan jika pada prosesnya inputan bernilai positif maka output dari neuron adalah nilai aktivasi itu sendiri [14]. Fungsi aktivasi ini memiliki kelebihan yaitu dapat mempercepat proses konfigurasi yang dilakukan dengan Stochastic Gradient Descent (SGD) jika dibandingkan dengan fungsi sigmoid dan tanh. Namun aktivasi ini juga memiliki kelemahan yaitu aktivasi ini bisa menjadi rapuh pada proses training dan bisa membuat unit tersebut mati.

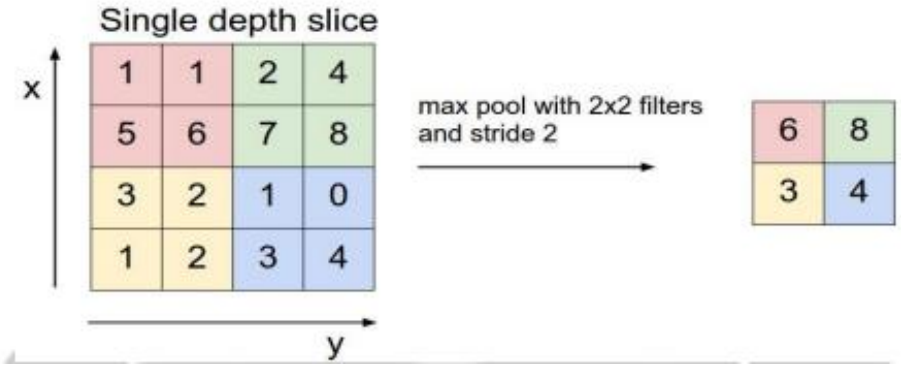
- Fungsi Aktivasi Softmax

Fungsi Aktivasi Softmax atau dikenal dengan Softmax Classifier merupakan sebuah fung aktivasi yang dimanfaatkan guna melakukan klasifikasi, biasanya pada algoritma Convolutional Neural Network (CNN) aktivasi ini di implementasikan pada output layer. Softmax classifier adalah generalisasi untuk beberapa kelas. *Softmax classifier* meanwarkan output yang sedikit lebih intuitif (probabilitas kelas yang dinormalisasi) dan juga memiliki interpretasi probabilitas yang akan langsung dijabarkan. Pada intinya fungsi ini merupakan probabilitas eksponential yang dinormalisasi dari pengamatan kelas yang di sebut sebagai aktivasi neuron. Tujuan dari lapisan ini yaitu untuk memprediksi output klasifikasi dalam bentuk nilai probabilitas, dimana nilai probabilitas kelas terbesar merupakan output prediksi kelas yang diperoleh. Softmax juga memberikan hasil yang lebih

intuitif dan memiliki hasil interpretasi probabilistik yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. Softmax memungkinkan peneliti untuk menghitung nilai probabilitas untuk semua label. Hasil dari label yang ada, akan diambil sebuah vektor nilai yang mempunyai nilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu. Jika semua hasil dijumlah maka akan bernilai sama dengan satu.

D. Pooling Layer

Pooling Layer adalah lapisan yang menggunakan fungsi dengan Feature Map sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. Pada model CNN, lapisan *Pooling* biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan Pooling yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume output pada Feature Map, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, dan untuk mengendalikan Overfitting. Hal terpenting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan Pooling dan hal ini bisa menguntungkan kinerja model. Lapisan Pooling bekerja di setiap tumpukan Feature Map dan mengurangi ukurannya. Bentuk lapisan Pooling yang paling umum adalah dengan menggunakan filter berukuran 2x2 yang diaplikasikan dengan langkah sebanyak 2 dan kemudian beroperasi pada setiap irisan dari input. Bentuk seperti ini akan mengurangi Feature Map hingga 75% dari ukuran aslinya.

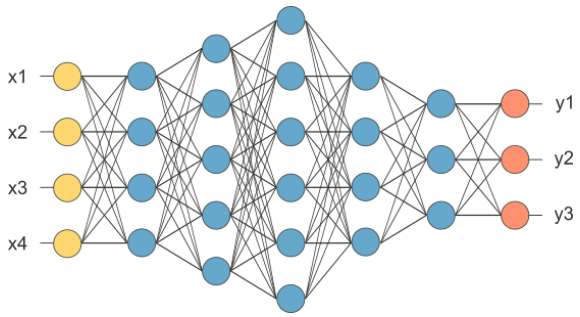


Gambar 2.9 ***Operasi Pooling Layer***

Lapisan Pooling akan beroperasi pada setiap irisan kedalaman volume input secara bergantian. Pada gambar di atas, lapisan pooling menggunakan salah satu operasi maksimal yang merupakan operasi yang paling umum. Gambar 2.9 menunjukkan operasi dengan langkah 2 dan ukuran filter 2x2. Dari ukuran input 4x4, pada masing-masing 4 angka pada input operasi mengambil nilai maksimalnya dan membuat ukuran output baru menjadi 2x2.

E. Fully Connected layer

Lapisan fully connected layer adalah suatu lapisan yang terbentuk dari kumpulan hasil proses konvolusi. Lapisan ini memiliki input dan proses sebelumnya untuk menentukan fitur manakah yang memiliki hubungan atau korelasi dengan kelas tertentu. Selain itu fungsi yang diperoleh dari fully connected layer berfungsi untuk menyatukan semua node menjadi satu dimensi. Umumnya proses fully connected ini terletak pada akhir arsitektur, prosesnya direpresentasikan sebagai perkalian matriks sederhana yang diikuti dengan penambahan vektor bias dan menerapkan fungsi non-linear sebagai berikut: $y = (WTx + b)$ dimana y merupakan vektor output yang berisi fungsi dari W yakni matriks sederhana yang memiliki bobot koneksi antar unit, lalu dikalikan dengan x sebagai vektor input, dan ditambah dengan b sebagai vektor bias.

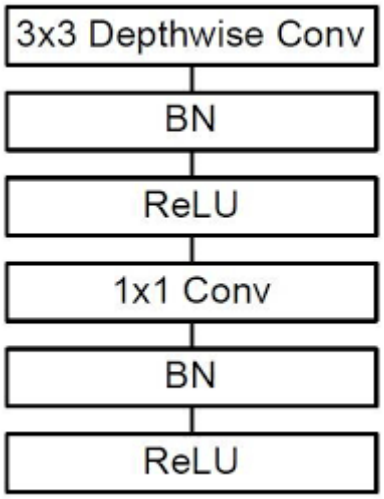


Gambar 2.10 *Fully Connected Layer*

2.6 Arsitektur MobilenetV1

MobileNet atau MobileNetV1 merupakan model yang memiliki ukuran kecil baik dari jumlah parameter maupun ukuran model yang dihasilkan. Seperti namanya, Mobile, para peneliti dari Google membuat arsitektur CNN yang dapat

digunakan untuk kebutuhan mobile. MobileNetV1 adalah sebuah arsitektur model yang dikembangkan untuk pengembangan aplikasi pada perangkat mobile ataupun perangkat lain yang memiliki keterbatasan sumber daya perangkat keras dengan mengurangi ukuran dan kompleksitas model menggunakan depthwise separable convolutions. Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan input image. Gambar 2.11 menunjukan gambar depthwise separable convolutions, yakni blok layer yang tersusun dari depthwise convolution dan pointwise convolution, masing-masing layer tersebut diikuti oleh batch normalization dan ReLU nonlinearity berikut adalah gambar arsitektur depthwise separable convolutions.



Gambar 2.11 ***Depthwise separable convolution dengan ReLU dan BN***

Blok layer tersebut kemudian disusun secara berulang membentuk arsitektur MobileNet. Berdasarkan pustaka keras, MobileNet memiliki jumlah layer sebanyak 28 convolution layer dan 1 fully connected layer yang diikuti oleh lapisan softmax.

2.7 Ekstraksi Fitur

Pemanfaatan (pretrained network) yaitu jaringan yang sudah dilatih sebelumnya sangat efektif untuk deep learning pada dataset gambar dalam jumlah sedikit. Jaringan pretrained adalah jaringan yang disimpan yang sebelumnya dilatih pada dataset dengan skala yang besar, biasanya dimanfaatkan pada kasus klasifikasi

gambar skala besar [15]. Jika dataset itu cukup besar dan cukup umum, maka hierarki spasial dari fitur yang dipelajari oleh jaringan yang belum dilatih dapat secara efektif bertindak sebagai model generik dari dunia visual atau dapat digunakan sebagai model dasar, dan karenanya fitur-fiturnya dapat terbukti berguna untuk banyak masalah komputer visual, meskipun masalah-masalah baru ini mungkin melibatkan yang benar-benar berbeda dari yang ada di tugas aslinya. Misalnya, digunakan untuk melatih jaringan di ImageNet (di mana kelas sebagian besar adalah binatang dan benda di kehidupan sehari-hari) dan kemudian menggunakan kembali jaringan yang terlatih ini untuk masalah yang lain seperti mengidentifikasi benda mati dalam gambar. Portabilitas fitur-fitur yang dipelajari di berbagai masalah yang berbeda merupakan keuntungan utama dari deep learning dibandingkan dengan konsep lainnya, dan itu membuat deep learning sangat efektif untuk masalah dataset kecil. Feature extraction terdiri dari penggunaan representasi yang dipelajari oleh jaringan sebelumnya untuk mengekstraks fitur menarik dari sampel baru [15] Fitur-fitur ini kemudian dijalankan melalu classifier baru yang dilatih dari awal.

Algoritma CNN yang digunakan untuk klasifikasi gambar terdiri dari dua bagian yaitu dimulai dengan sebaris pooling layer dan convolution layer, dan berakhir dengan pengklasifikasi yang terhubung erat (densely connected classifier). Bagian pertama disebut basis konvolusional (convolutional base) dari model. Dalam CNN, ekstraksi fitur mengambil basis konvolusional dari jaringan yang sebelumnya dilatih, menjalankan data baru melaluinya, dan melatih pengklasifikasian baru di atas output-nya. Hal tersebut dikarenakan representasi yang dipelajari oleh convolutional base cenderung lebih generik atau umum dan oleh karena itu dapat digunakan kembali. Peta fitur (feature map) dari sebuah CNN adalah peta keberadaan konsep-konsep umum pada gambar, yang mungkin berguna terlepas dari masalah visual komputer. Tetapi representasi yang dipelajari oleh pengklasifikasi tentu akan spesifik untuk set kelas di mana model dilatih. Mereka hanya akan berisi informasi tentang kemungkinan kehadiran kelas ini atau itu di seluruh gambar Selain itu, representasi yang ditemukan di lapisan yang terhubung erat tidak lagi berisi informasi tentang di mana objek berada pada gambar yang

dimasukkan. Lapisan ini menghilangkan gagasan ruang, sedangkan lokasi obyek masih dapat digambarkan oleh convolutional feature map. Untuk masalah dimana lokasi obyek penting, fitur-fitur yang terhubung secara rapat sebagian besar tidak berguna.

2.8 Akurasi

Akurasi adalah sebuah matrik untuk mengevaluasi hasil dari model klasifikasi. Akurasi merupakan pembagian dari prediksi model yang dianggap benar dengan jumlah total yang diprediksi, perhitungan tersebut didefinisikan sebagai berikut:

$$\text{Akurasi} = \frac{Jumlah\ prediksi\ benar}{Jumlah\ total\ prediksi} \text{ x } 100\%$$

2.9 Perangkat Lunak Pendukung

Perangkat lunak pendukung adalah alat yang memiliki fungsi sebagai pendukung dalam proses dalam pembangunan perangkat lunak dalam penelitian. Dapat berupa program atapun aplikasi, bahasa pemrograman, dan lain-lain. Berikut adalah beberapa perangkat pendukung yang digunakan dalam penelitian ini.

2.9.1. Android Studio

Android Studio merupakan sebuah software atau aplikasi Integrated Development Environment (IDE) resmi yang dapat digunakan untuk proses development aplikasi android, berdasarkan Intellij IDEA. Android Studio merupakan aplikasi IDE yang banyak digunakan dan familiar dikalangan developer atau pengembang aplikasi android Disamping sebagai editor serta alat pengembang intellij yang baik, Android Studio juga memberikan penawaran yang menarik dengan banyak fitur yang dapat meningkatkan produktivitas pengguna saat development aplikasi yang berbasis android. Berikut beberapa fitur Android Studio :

a. Support beberapa bahasa pemrograman

b. Device Emulator yang lengkap

c. Integrasi dengan GitHub yang dapat membantu dokumentasi kode dan kolaborasi dalam pembangunan aplikasi.

d. Gradle yang fleksibel

e. Support integrasi dengan banyak library dan API

f. dan lain- lain.

Gambar 2.12 ***Logo Android Studio***

Beberapa tugas Android studio sebagai Integrated Development Environmen ialah menyediakan antarmuka guna pembangunan aplikasi dan mengelola manajemen file yang dapat dikatakan kompleks. Bahasa pemrograman yang digunakan ialah Kotlin. Beberapa hal yang dapat dilakukan oleh Android Studio adalah menulis, mengedit, dan menyimpan proyek beserta berbagai file yang berkesinambungan dengan proyek tersebut.

Selain itu, Android Studio juga memberikan akses terhadap Android Software Development Kit (SDK). SDK tersebut dapat dikatakan sebagai ekstensi dari kode java yang memperbolehkannya dapat berjalan dengan baik di Device Android. Jika Java dibutuhkan untuk menulis programnya, SDK di perlukan untuk menjalankan program di Device Android. Lalu untuk menggabungkan diperlukan

Android Studio. Disamping itu jika menemukan bug ataupun keslahan kode pada aplikasi yang dibangun maka kita dapat melihat dan memperbaikinnya dengan Android Studio.

Google terus melakukan percobaan guna Android Studio menjadi lebih optimal serta memastikan bahwa perangkat lunak ini dapat membantu membangun Aplikasi Android. Android Studio menawarkan saran kode maupun saran kesalahan jika menurut mereka terdapat kesalahan sehingga proses pembangunan aplikasi android dapat lebih efisien dan efektif.

2.9.2 Kotlin

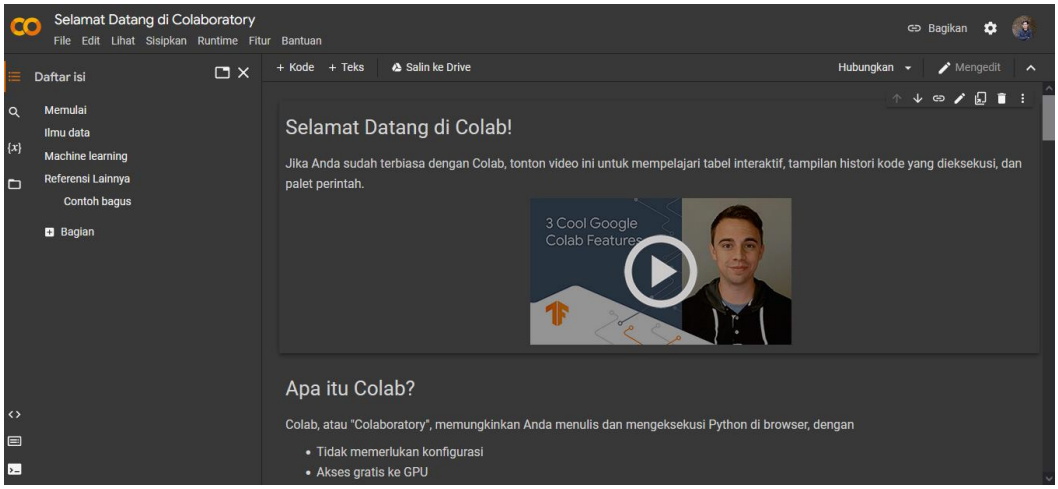
Kotlin merupakan bahasa pemrograman pengembangan dari Java yang dianggap rumit dan memiliki banyak aturan. Kotlin dibuat oleh JetBrains pada tahun 2011. Kotlin adalah bahasa pemrograman open-source yang dibuat oleh JetBrains untuk berbagai platform. JetBrains mencetus ide membuat bahasa pemrograman yang ringkas namun tetap ingin memanfaatkan keunggulan java yaitu ekosistem yang luas, sehingga kotlin diharapkan tetap dapat menggunakan seluruh ekosistem milik java.Namun seiring berjalannya waktu kotlin justru popular dimanfaatkan untuk pembangunan aplikasi Android. Bahasa kotlin dapat berjalan pada platform Java Virtua Machine (JVM), dimana sebuah platform yang dapat memungkinkan koputer menjalankan kode berbasis java, atau kode yang berasal dari bahasa lain yang dikompilasi (compile) menggunakan Java. Kotlin dapat digunakan bersamaan dengan Java, Dimana kotlin dan java dapat dimplemetasikan dalam satu proyek atau satu aplikasi Android.

Gambar 2.13 ***Logo Kotlin***

Kotlin termasuk bahasa pemorgraman dengan jenis (statically typed) atau ditulis secara statis, Dimana semua variable yang akan digunakan dalam pembangunan aplikasi harus terlebih dahulu dikenalkan apa jenisnya. Sehingga pada saat proses compiled, program harus sudah benar – benar dalam keadaan clean dari error. Kotlin melesat pesat akibat dari dukungan google dan saran proritas sebagai bahasa pemrograman utama untuk Android.

2.9.3 Google Colaboratory

Google Colaboratory atau kerap disebut Google Colab merupakan layanan dari google berbasisi cloud yang mereplikasi Jupyter Notebook. Google Colaboratory merupakan executable document yang bisa dimanfaatkan untuk melakukan penulisan kode, editor kode, serta menyimpan, dan membagikan program yang telah di tulis melalu Google Drive .Keuntungan menggunakan Google Colaboratory pengguna tidak usah menginstal apapun pada perangkat mereka untuk menjalankannya. Sebagian besar hal, pemanfaatan Google Colaboratory hampir mirip layaknya yang di lakukan pada instalasi Jupyter Nootebook. Berikut tampilan antar muka Google Colaboratory.



Gambar 2.14 *Tampilan Antarmuka Google Colaboratory*

Jika menjalankan sebuah notebook dalam Google Colab, Pengguna melakukan komputasi disalah satu computer Google melalui Virtual Machine (Mesin Virtual). Salah satu syarat untuk menggunakan Google Colab harus memiliki akun Google agar semua fitur yang terdapat pada Google Colaboratory dapat berfungsi dengan baik. Sama sepert Jupyter Notebook, Penggunaan Google Colaboratory dapat melakukan tugas – tugas tertentu menggunakan paradigma berorientasi sel. Disamping itu tampilan antarmuka Google Colab dan Jupyter Notebook sangat mirip. Dalam Google Colaboratory dapat membuat sel untuk menulis kode maupun membuat catatan.

2.9.4 Keras dan Tensorflow

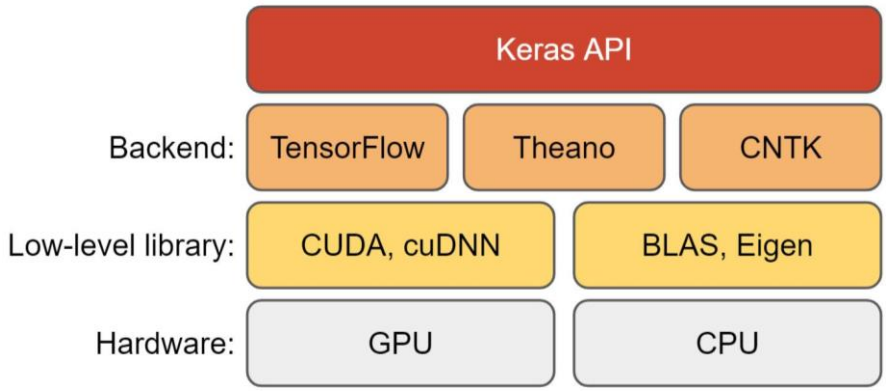
Keras merupakan sebuah *framework deep learning* yang ditulis menggunakan bahasa pemrograman Python yang menyedikan kemudahan dalam melakukan mendefinisikan dan melatih hampir seluruh jenis *model deep learning*. Keras merupakan high-level API neural network API yang dikembangkan dengan bahasa *Python* dengan tujuan utama mempercepat proses riset atau percobaan. Berikut beberapa fitur utama dari Keras:

- Mampu menjalankan *source code* yang sama menggunakan CPU atau GPU dengan lancar

- API yang *user-friendly* sehingga mempermudah penggunanya dalam proses prototipe model deep learning
- Dukungan *built-in* untuk CNN atau Convolutional Neural Networks (Computer Vision), RNN atau Recurrent Neural Networks (untuk *sequence processing*), dan kombinasi keduanya
- Dapat digunakan untuk hampir semua jenis dari model deep learning

Gambar 2.15 *Logo Keras*

Keras merupakan *library* tingkat model, meneydiakn blok-blok tingkat tinggi untuk mengembangkan model *deep learning*. *Keras* Keras tidak dapat melakukan operasi tingkat rendah seperti manipulasi tensor dan diferensiasi. Keras bergantung pada library tensorflow yang dioptimalkan dengan baik untuk menjalankannya, memiliki manfaat sebagai backend dari *Keras.* Tidak hanya *tensorflow* yang berfungsi sebagai backend, melainkan terdapat dua implementasi *backend* yang ada selain itu yaitu: *backend Theano,* dan *backend Microsoft Cognitive Toolkit (CNTK).* Beikut merupakan susunan perangkat lunak dan perangkat keras dari *deep learning*.

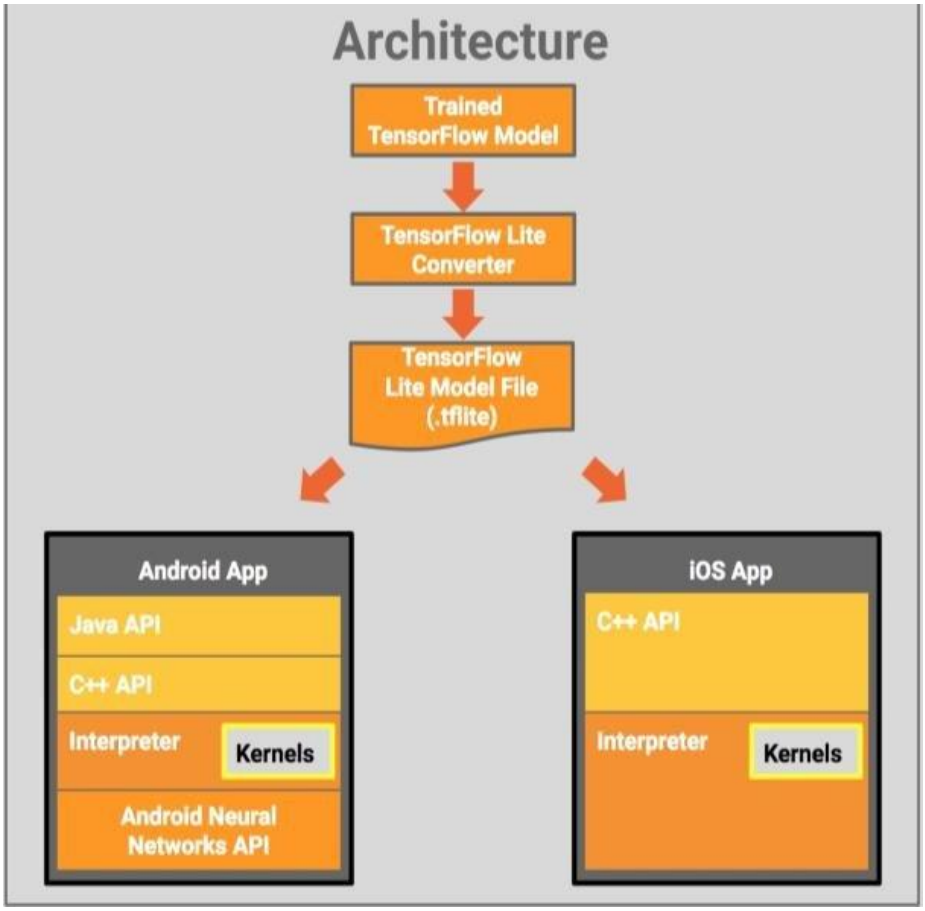


Gambar 2.16 ***Susunan perangkat lunak dan perangkat keras deep learning***

Melalui *Backend TensorFlow*, *Theano*, ataupun *CNTK Keras* dapat berfungsi dengan baik pada CPU dan GPU. Saat berjalan pada CPU, *TensorFlow* memanfaatkan *library* tingkat rendah untuk operasi *tensor* yang disebut dengan *Eigen.* Pada saat berjalan di GPU, *TensorFlow* memanfaatkan *library* operasi *deep learning* yang dioptimalkan dengan baik yang disebut *Library NVIDIA CUDA Deep Neural Netwrok (*cuDNN*)*.

2.9.5 Tensorflow Lite

Tensor Flow Lite merupakan *library machine learning* yang dirancang khusus untuk perangkat mobile (baik Android maupun IOS). *Tensor Flow Lite* memungkinkan mesin untuk "belajar" di perangkat dengan latensi rendah dan ukuran binary yang kecil. *TensorFlow* memiliki tugas sebagai converter dari model *Machine Learning TensorFlow* yang sebelumnya telah dilakukan proses pelatihan agar dapat di implementasikan pada perangkat mobile. *TensorFlow* sendiri merupakan sebuah aplikasi yang bersifat *end-to-end open source platform* yang banyak dimanfaatkan untuk *machine learning*. Sehingga dapat dimanfaatkan untuk melakukan klasifikasi, pendeteksian, maupun regresi sesuai yang di hendaki tanpa perlu melakukan proses POST (pengiriman) dan GET (penerimaan) dari atau menuju *server*. Berikut gambaran arsitektur *TensorFlow Lite* dapat dlihat pada gambar 2.17 berikut.



Gambar 2.17 ***Arsitektur TensorFlow Lite***

Pada saat ini *TensorFlow Lite* support dengan *Operating System* (OS) Android dan iOS memanfaatkan C++ API (*Application Program Interface*), serta menyediakan *Java Wrapper* untuk Pengembang Android. Disamping itu, Pada *Operating System* (OS) Android yang didukung, penerjemah (*interpreter*) juga dapat memanfaatkan *Android Neural Networks* API untuk akselerasi perangkat keras, jika tidak secara default proses dengan CPU (*Central Processing Unit*) selanjutnya di eksekusi.

TensorFlow Lite terdiri dari *runtime* yang dapat dimanfaatkan untuk menjalankan model yang sudah teraltih sebelumnya, dan berbagai alat (*tools*) ysng dapat di implementasikan untuk menyiapkan model yang akan digunakan pada perangkat berbasis mobile dan sifat tertanam.

2.9.6 Database Firebase

Firebase merupakan layanan database berbasis cloud yang disediakan oleh Google. *Firebase* termasuk kedalam database dengan basis NoSQL (bukan sql), dimana data yang terdapat tidak terstruktur (tidak membutuhkan skema dan tidak memiliki relasi untuk setiap tabel). Google pertama kali memperkenalkan *Firebase* pada tahun 2016 tepatnya bulan mei pada acara tahunan Google I/O. Layanan *Firebase* mendukung berbagai platform seperti: Android, iOS, hingga website. *Firebase* mendukung Baas (*Backend as a Service*) merupakan solusi yang ditawarkan oleh Google agar mempercepat dan mepermudah pekerjaan developer. Data yang disimpan dalam database *Firebase* memiliki struktur yang mirip dengan *JSON*.

Layanan yang tersedia dari Firebase ada 3 pilihan, diantaranya:

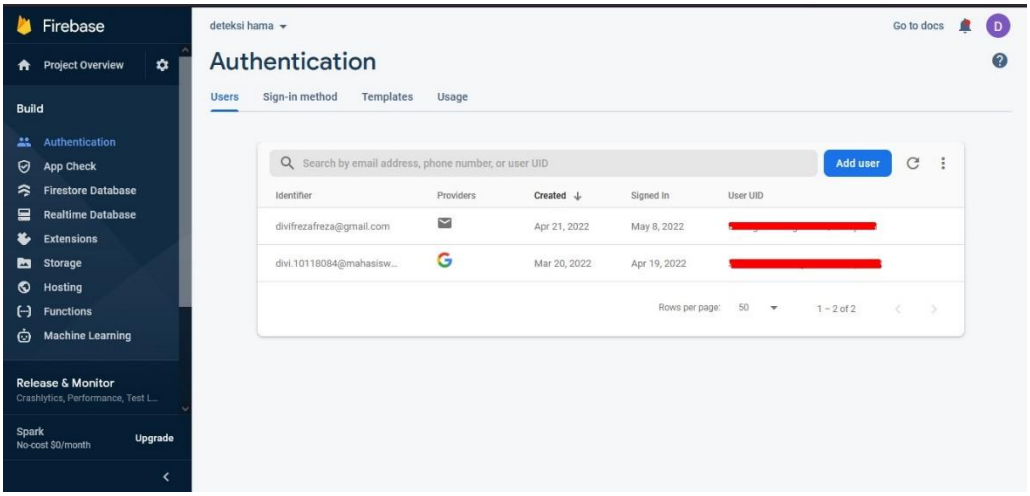
- ***SPARK*** : kita bisa menggunakan layanan ini secara gratis
- ***FLAME*** : untuk menggunakan layanan ini kita dikenakan biaya $25 / bulan
- ***BLAZE*** : sedangkan jika ingin menggunakan layanan yang ketiga ini kita dikenakan biaya sesuai dengan pemakaian

Gambar 2.18 *Logo Firebase*

Firebase menawarkan banyak fitu diantaranya: *Firebase Authentication, Firebase Cloud Firestore, Firebase Realtime Database, Firebase Hosting,* dan lain - lain.

A. *Firebase Authentication*

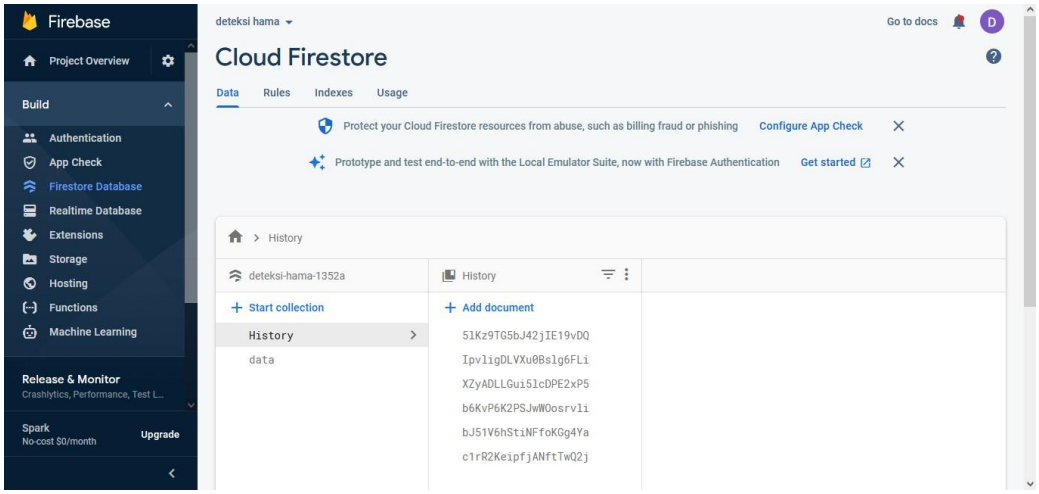
Firebase Authentication merupakan salah satu fitur layanan back-end, yang mudah digunakan support terhadap berbagai platform guna mengautentikasi pengguna ke aplikasi yang *developer* buat. Firebase Authentication mendukung autentikasi menggunakan nomor telepon, sandi, penyedia identitas gabungan populer seperti seperti Google, Facebook, dan sebagainya. *Firebase Authentication* juga sudah menerapkan berbagai jenis standardisasi industry, seperti OAuth 2.0 dan OpenId Connect, yang berguna memudahkan integrase dengan backend custom buatan developer.



Gambar 2.19 ***Antarmuka Firebase Authentication***

B. *Firebase Cloud Firestore*

Cloud Firestore adalah *database* yang bersifat fleksibel dan terukur untuk pengembangan perangkat seperti seluler, web, dan server di Firebase dan *Google Cloud Platform*. Seperti halnya Firebase Realtime Database, Cloud Firestore membuat datamu tetap terkoneksi di aplikasi *user* melalui *listener realtime* dan menawarkan layanan secara *offline* untuk aplikasi seluler dan web. Dengan begitu, kamu dapat membuat aplikasi yang *powerfull*, responsif, dan mampu bekerja tanpa bergantung pada latensi koneksi internet. Cloud Firestore merupakan database NoSQL yang dihosting di *cloud* dan dapat diakses melalui SDK *real* oleh aplikasi iOS, Android dan web.

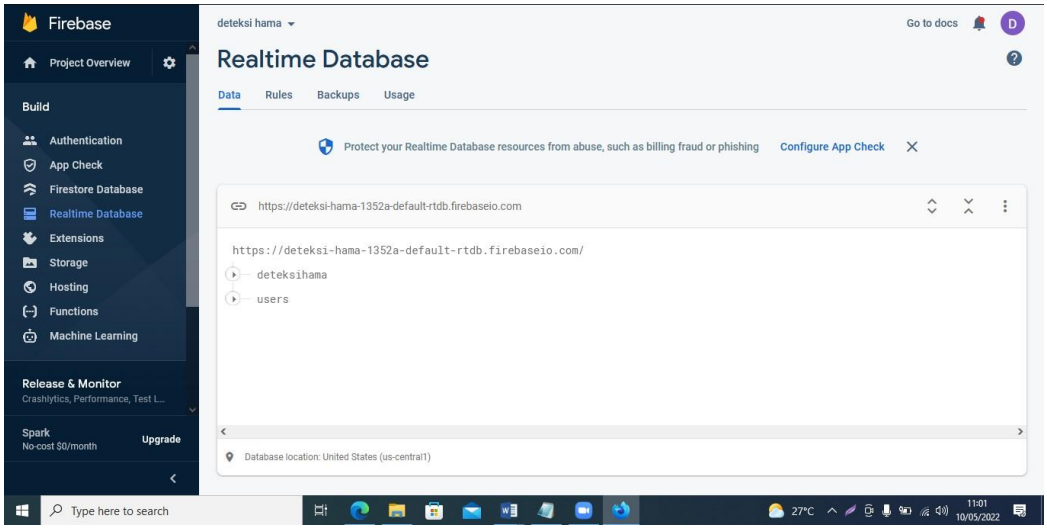


Gambar 2.20 ***Antarmuka Firebase Cloud Firestore***

C. *Firebase Realtime Database*

Firebase Realtime *Database* adalah *database* yang di-host melalui cloud. Data disimpan dan dieksekusi dalam bentuk *JSON* dan disinkronkan secara *realtime* ke setiap *user* yang terkoneksi. Hal ini berfungsi memudahkan kamu dalam mengelola suatu *database* dengan skala yang cukup besar. Ketika kamu membuat aplikasi lintas-platform/*multiplatform* menggunakan SDK Android, iOS, dan juga JS (*JavaScript),* semua pengguna akan berbagi sebuah instance *Realtime Database* dan menerima *update*-an data secara serentak dan otomatis.

Kemampuan lain dari Firebase Realtime *Database* adalah tetap responsif bahkan saat *offline* karena SDK Firebase Realtime *Database* menyimpan data langsung ke *disk device* atau memori lokal. Setelah perangkat terhubung kembali dengan internet, perangkat pengguna (*user*) akan menerima setiap perubahan yang terjadi.



Gambar 2.21 ***Antarmuka Firebase Realtime Database***