BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Identifikasi terhadap penelitian dan studi terdahulu mengenai pembuatan model deteksi diabetes retinopati dengan menggunakan bantuan *machine learning* dan *deep learning* menjadi penting untuk memastikan bahwa penelitian yang dilakukan tidak menjadi sebuah redudansi pada penelitian di bidang tersebut. Mengenali penelitian dan studi terdahulu memberikan masalah yang masih terjadi dan aspek dalam penelitian yang masih perlu dikembangkan. Oleh karena itu, data temuan dari penelitian dan studi terdahulu digunakan sebagai sebuah masukan bagi penelitian ini. Pada tabel 2.1 dijabarkan data temuan dari penelitian terdahulu yang berhubungan dengan riset penelitian dengan judul "Diabetes Retinopati".

Tabel 2.1 - Penelitian Terdahulu

Diabetic Retinopathy Detection Using Deep Learning	
Artikel Jurnal #1	
Judul	Deep Learning Based Prediction of Diabetic Retinopathy Using CLAHE and ERSGAN for Enchancement[13]
Penulis	Ghadah Alwakid, Walaa Gouda, dan Mamoona Humayun
Tahun	2023
Jurnal	Healthcare MDPI
Algoritma	Inception v3
Dataset	APTOS
Hasil Penelitian	Penggunaan CLAHE dan ERSGAN meningkatkan performa model dengan *accuracy* 98,7% dibandingkan dengan tanpa menggunakan keduanya dengan *accuracy* hanya 80,87%.
Saran / Limitasi Penelitian	Pemeriksaan performa model pada dataset besar dan rumit serta penggunaan algoritma lainnya.
Artikel Jurnal #2	
Judul	Development of Revised ResNet-50 for diabetic retinopathy detection[14]
Penulis	Chun-Ling Lin dan Kun-Chi Wu
Tahun	2023
Jurnal	BMC Bioinformatics
Algoritma	Revised ResNet-50

Diabetic Retinopathy Detection Using Deep Learning	
Dataset	Diabetic Retinopathy Kaggle Competition Dataset
Hasil Penelitian	Menghasilkan *accuracy* latih 84% dan *accuracy* tes 74%
Saran / Limitasi Penelitian	Peningkatan *accuracy train* dan *test* serta penggunaan memori yang masih berlebihan.
Artikel Jurnal #3	
Judul	Pengaruh Preprocessing Terhadap Klasifikasi Diabetic Retinopathy dengan Pendekatan Transfer Learning Convolutional Neural Network[15]
Penulis	Juan Elisha Widyaya dan Setia Budi
Tahun	2021
Jurnal	Jurnal Teknik Informatika dan Sistem Informasi
Algoritma	Inception v3
Dataset	APTOS
Hasil Penelitian	Metode preprocessing enchanced green dengan menggunakan algoritma Inception v3 mendapatkan hasil *accuracy* paling baik jika dibandingkan dengan metode preprocessing lainnya.
Saran / Limitasi Penelitian	Penggunaan *hyperparameter tuning* dan optimasi model lainnya.
Artikel Jurnal #4	
Judul	Deep Learning untuk Klasifikasi Diabetic Retinopathy menggunakan Model EfficientNet[16]
Penulis	Syamsul Rizal, Nur Ibrahim, Nor Kumalasari Caesar Pratiwi, Sofia Saidah, dan Raden Yunendah Nur Fu'adah
Tahun	2020
Jurnal	ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, dan Teknik Elektronika
Algoritma	EfficientNet
Dataset	APTOS
Hasil Penelitian	Penggunaan pra-pemrosesan CLAHE, *optimizer* SGD, *learning rate* 0.001, dan momentum 0.9 membuat *accuracy* menjadi 79,8%, spesifitas 93,11%, dan *recall* 80%.
Saran / Limitasi Penelitian	Menggunakan dataset dengan jumlah seimbang pada setiap kelasnya
Artikel Jurnal #5	
Judul	Deep Learning-enhanced Diabetic Retinopathy Image Classification[17]
Penulis	Ghadah Alwakid, Walaa Gouda, Mamoona Humayun, dan Noor Zaman Jhanjh
Tahun	2023

Diabetic Retinopathy Detection Using Deep Learning	
Jurnal	Digital Health
Algoritma	DenseNet-121
Dataset	APTOS & DDR
Hasil Penelitian	Penggunaan preprocessing CLAHE+ERSGAN membuat hasil *accuracy* lebih baik bagi kedua dataset yakni 98,7% bagi APTOS dan 79,67% bagi DDR dibandingkan jika tidak menggunakan preprocessing CLAHE+ERSGAN yang menghasilkan *accuracy* 81,23% bagi APTOS dan 79,62% bagi DDR.
Saran / Limitasi Penelitian	Penambahan jumlah data pada *test* untuk mendapatkan kesimpulan yang valid dan dapat diandalkan.
Artikel Jurnal #6	
Judul	A Regression-Based Approach to Diabetic Retinopathy Diagnosis Using EfficientNet[18]
Penulis	Midhula Vijayan dan Venkatakrishnan S
Tahun	2023
Jurnal	Diagnostics
Algoritma	EfficientNet-B0
Dataset	APTOS & DDR
Hasil Penelitian	Dataset APTOS memiliki *accuracy* 86% dan nilai Kappa 93.9 sementara dataset DDR memiliki *accuracy* 85% dan nilai Kappa 90.3
Saran / Limitasi Penelitian	Mengevaluasi kinerja model menggunakan dataset lain dan menggabungkan model yang sudah ada dengan arsitektur lainnya untuk meningkatkan kinerja model.

(Sumber olahan peneliti, 2024)

Deteksi diabetes retinopati sudah pernah disinggung pada artikel jurnal penelitian sebelumnya. Pada artikel yang ditulis oleh Ghadah Alwakid et al. tahun 2023 pada jurnal Healthcare MDPI menggunakan pra-pemrosesan CLAHE dan ERSGAN pada algoritma Inception v3 dan menghasilkan *accuracy* sebesar 98,7% dibandingkan tanpa menggunakan pra-pemrosesan hanya mendapat *accuracy* di angka 80,87%[13]. Sementara itu Chun-Ling Lin dan Kun-Chi Wu yang menggunakan algoritma ResNet-50 pada jurnal BMC Bioinformatics pada tahun 2023 menghasilkan *accuracy* latih 84% dan *accuracy* tes 74%[14]. Penelitian lain yang dilakukan oleh Juan Elisha Widyaya dan Setia Budi menggunakan berbagai metode pra-pemrosesan dengan algoritma Inception v3 dan menghasilkan metode *enhanced green* sebagai metode pra-pemrosesan terbaik[15].

Artikel lain yang ditulis oleh Ghadah Alwakit et al. tahun 2023 di jurnal Digital Health menggunakan algoritma DenseNet-121 dan menggunakan pra-pemrosesan CLAHE dan ERSGAN meghasilkan *accuracy* 98,7% bagi APTOS dan 79,67% untuk DDR[17]. Lalu Syamsul Rizal et al. pada tahun 2020 di jurnal ELKOMIKA juga membuat sebuah model menggunakan algoritma EfficientNet B0 dan menghasilkan *accuracy* 79,8%, spesifitas 93,11%, dan sensivitas 80%[16]. Sementara itu, Midhula Vijayan dan Vekatakrishnan pada tahun 2023 di jurnal Diagnostics menggunakan EfficientNet B0 dan menghasilkan *accuracy* 86% dan nilai Kappa 93.9 untuk dataset APTOS dan *accuracy* 85% dan nilai Kappa 90.3 untuk dataset DDR[18].

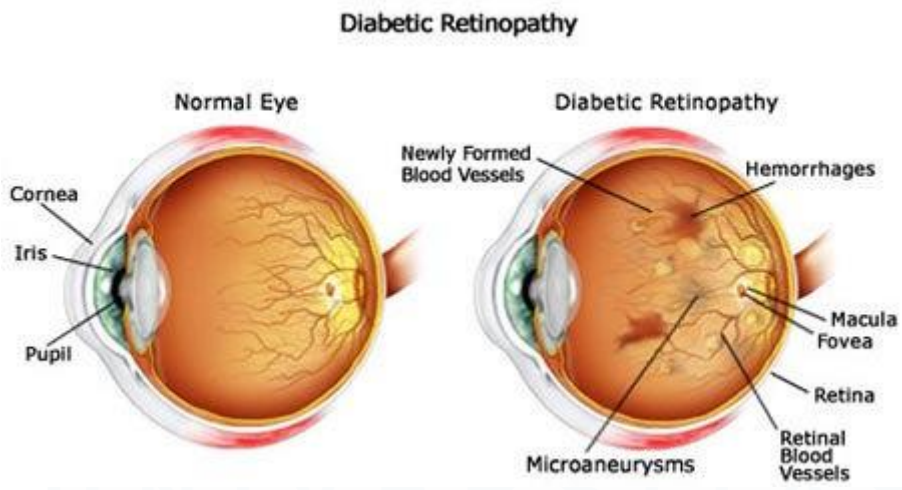
Artikel jurnal yang telah disebutkan diatas mayoritas menggunakan dataset APTOS lalu disusul DDR dan Diabetic Retinopathy Kaggle Competition Dataset. Penelitian ini juga menggunakan dataset APTOS namun berbeda dari artikel penelitian sebelumnya penelitian ini menggunakan pra-pemrosesan GABOR, algoritma CNN dan ConvNeXt, serta penggunaan metrik evaluasi *accuracy*, *precision, recall,* dan f1-*score* sebagai kebaharuan dari penelitian.

2.2 Diabetes Retinopati

Diabetes retinopati (DR) adalah sebuah penyakit kronis yang merupakan hasil komplikasi dari diabetes melitus tipe 2 yang terjadi di berbagai organ. Diabetes retinopati juga menjadi alasan paling umum terjadinya penyakit pembuluh darah retina[19]. Diabetes retinopati juga menjadi penyebab kelima terbanyak dari kebutaan yang dapat diobati dan penyebab gangguan penglihatan sedang hingga berat pada orang berumur 50 tahun atau lebih tua pada tahun 2020[20]. Semua pasien penderita diabetes memiliki kemungkinan untuk terkena diabetes retinopati namun ada beberapa faktor yang dapat membuat peluang terkena diabetes retinopati lebih tinggi yakni kadar kolesterol tinggi, tekanan darah tinggi, pasien sedang hamil, pasien yang memiliki kebiasaan merokok, dan kadar gula darah pasien yang tidak terkontrol. Diabetes retinopati juga akan menunjukkan gejalanya seiring dengan waktu berjalan seperti penglihatan menurun, adanya bercak hitam pada

penglihatan, adanya noda melayang, penglihatan yang berbayang, dan rasa nyeri pada mata[21].

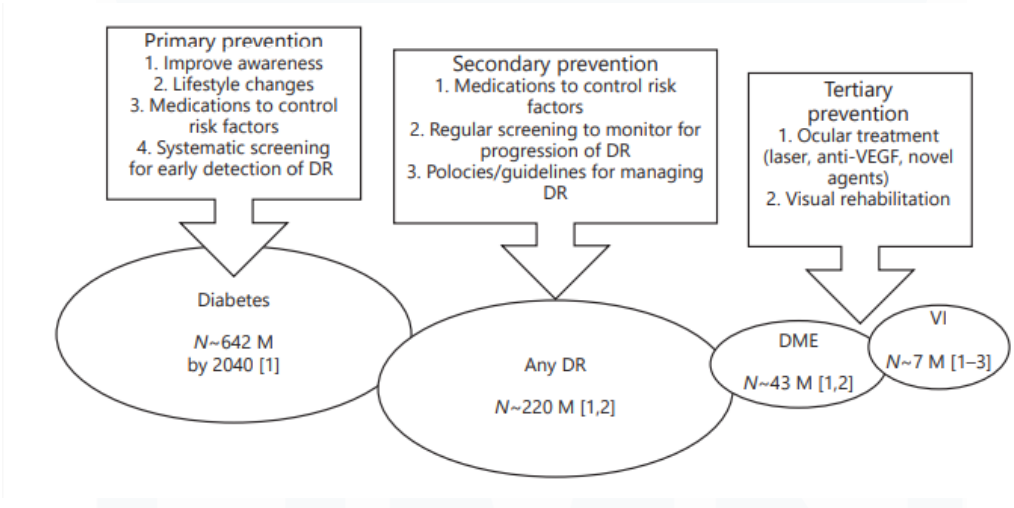
Diabetes retinopati kemudian diklasifikasikan menjadi dua tahap yakni diabetes retinopati proliferasi dan diabetes retinopati non-proliferasi, dimana pada non-proliferasi memiliki tiga level yakni ringan, sedang, dan berat[22]. Pada tiap level diabetes retinopati, terdapat diagnosa yang dapat ditemukan pada retina pasien, pada level ringan ditemukan adanya *microaneurysm* pada retina pasien, kemudian pada level sedang retina pasien memiliki *hemorrhages* atau *microaneurysm* dan/atau adanya *hemorrhages retina, exudates* keras, *cotton wool spots*, dan *venomous beading*, kemudian pada level berat retina pasien dapat ditemukan menggunakan aturan 4-2-1 yang memenuhi salah satu syarat lebih dari 20 pendarahan pada intraretina mata di empat kuadran, adanya *venomous beading* pada 2 atau lebih kuadran, atau kelainan mikrovaskular intraretinal (IRMA/*Intraretinal Microvascular Abnormality*) pada 1 kuadran[22]. Gambar 2.1 di bawah ini menunjukkan perbedaan dari mata orang normal dan mata penderita diabetes retinopati.



Gambar 2.1 - Perbedaan Mata Normal dan Penderita DR[23]

Diabetes retinopati yang diprediksi akan mencapai angka 642 juta pasien di tahun 2040 tentunya dapat dicegah menggunakan berbagai cara. Pencegahan yang dilakukan sendiri kemudian dibagi menjadi 3 tahap pencegahan yakni pencegahan

primer, pencegahan sekunder, dan pencegahan tersier. Pencegahan primer diberikan pada pasien penderita diabetes yang tidak memiliki diabetes retinopati dengan tujuan menghindari atau menunda terjadinya diabetes retinopati. Pencegahan sekunder diberikan pada pasien yang sudah terkena diabetes retinopati dan bertujuan untuk mencegah diabetes retinopati berkembang ke tahap selanjutnya. Sementara itu, pencegahan tersier adalah pencegahan pada penderita diabetes level berat atau sudah mengancam penglihatan (VTDR/*Vision Threatening Diabetic Retinopathy*) dengan tujuan mencegah kebutaan, mengembalikan penglihatan, dan meningkatkan kualitas hidup pasien yang telah terkena kebutaan[24].



Gambar 2.2 - Prevensi Diabetes[24]

Prevensi diabetes sendiri dapat dilihat pada gambar 2.2, dimana pada prevensi primer, dilakukan beberapa tindakan seperti perubahan gaya hidup menjadi hidup sehat sebelum terkena diabetes, meningkatkan kesadaran akan adanya diabetes retinopati, pengelolaan diabetes dan hipertensi secara mandiri, dan melakukan skrining diabetes retinopati secara rutin untuk melakukan deteksi dini ada atau tidaknya diabetes retinopati. Selanjutnya pada prevensi sekunder dilakukan beberapa tindakan seperti intervensi medis untuk melakukan kontrol pada glikemik dan tekanan darah, pemantauan diabetes retinopati agar tidak berkembang, penerapan pedoman dan kebijakan untuk sistem manajemen diabetes retinopati di

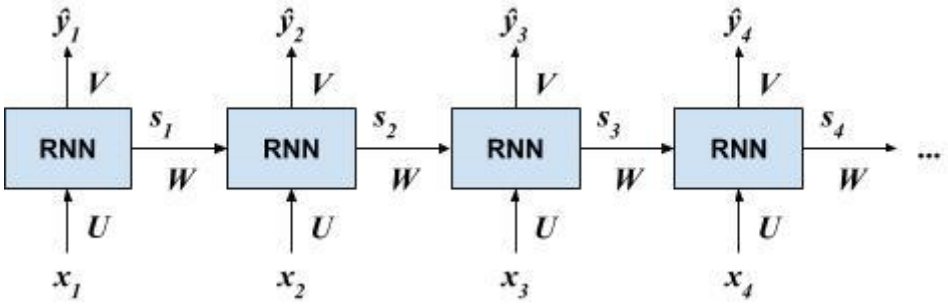
setiap negara. Pada prevensi tersier sendiri dilakukan beberapa tindakan seperti laser untuk diabetes retinopati proliferatif, terapi anti-VEGF (*Vascular Endothelial Growth Factor*), vitrektomi untuk diabetes retinopati tingkat lanjut, dan peningkatan kualitas hidup dengan rehabilitasi visual dengan kebutaan akibat diabetes retinopati[24].

2.3 Deep Learning

Deep learning adalah sebuah cabang dari *machine learning* yang memiliki lapisan hierarki dari tahap pemrosesan non-linear yang akan digunakan untuk membantu pembelajaran fitur tanpa pengawasan yang dapat mengklasifikasikan sebuah pola. *Deep Learning* menjadi salah satu metode diagnosis menggunakan bantuan komputer dan dapat digunakan untuk analisis citra medis seperti klasifikasi, segmentasi, deteksi gambar, dan lain-lain[25].

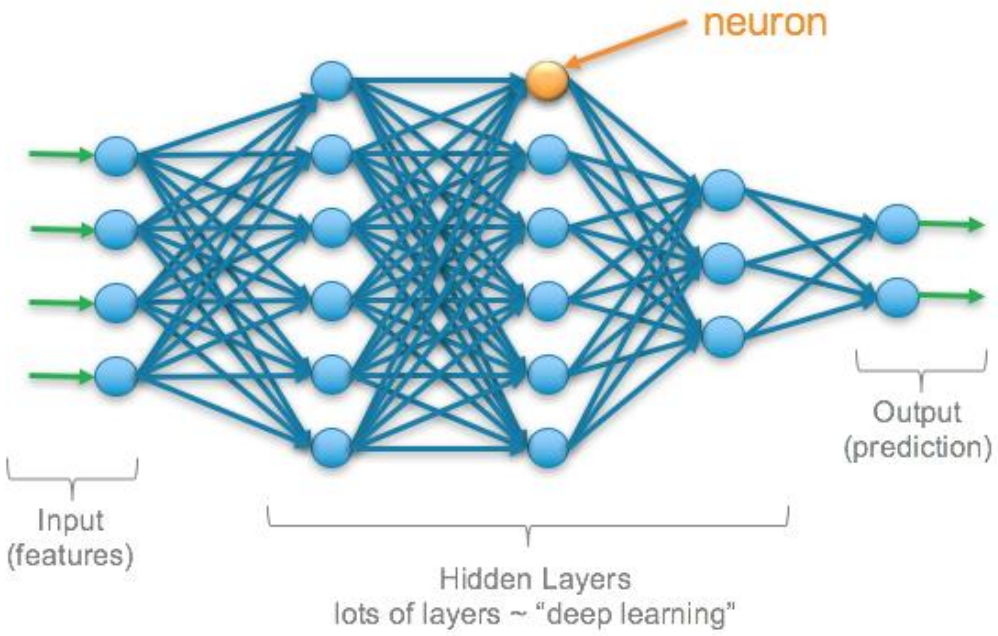
Deep learning memiliki beberapa alasan untuk digunakan yakni *deep learning* memiliki kemampuan untuk bekerja di hampir semua domain aplikasi sehingga biasanya disebut kemampuan pembelajaran universal, kemudian *deep learning* juga memiliki kekokohan dengan mempelajari fitur yang dioptimalkan dengan cara otomatis, lalu *deep learning* juga dapat digunakan pada tipe data atau aplikasi yang berbeda dimana pendekatan ini sering disebut *transfer learning*(TL). Alasan lainnya adalah skalabilitas dimana model yang dibuat dengan *deep learning* memiliki kemampuan untuk menangani data dan sumber daya yang besar sambil memperhatikan kinerja model tersebut[26].

Deep learning memiliki beberapa tipe arsitektur yang dapat digunakan untuk membangun model *deep learning*. Beberapa contoh arsitektur antara lain Convolutional Neural Network (CNN) yang dapat dilihat pada gambar 2.6, Recurrent Neural Network (RNN) yang dapat dilihat pada gambar 2.3 dimana RNN adalah salah satu jenis arsitektur yang dapat digunakan untuk memproses data urutan atau rangkaian seperti data teks, audio, dan data waktu. RNN juga memiliki kemampuan untuk mengingat informasi dan kemudian menggunakannya untuk menghasilkan *output*.



Gambar 2.3 - Arsitektur RNN[27]

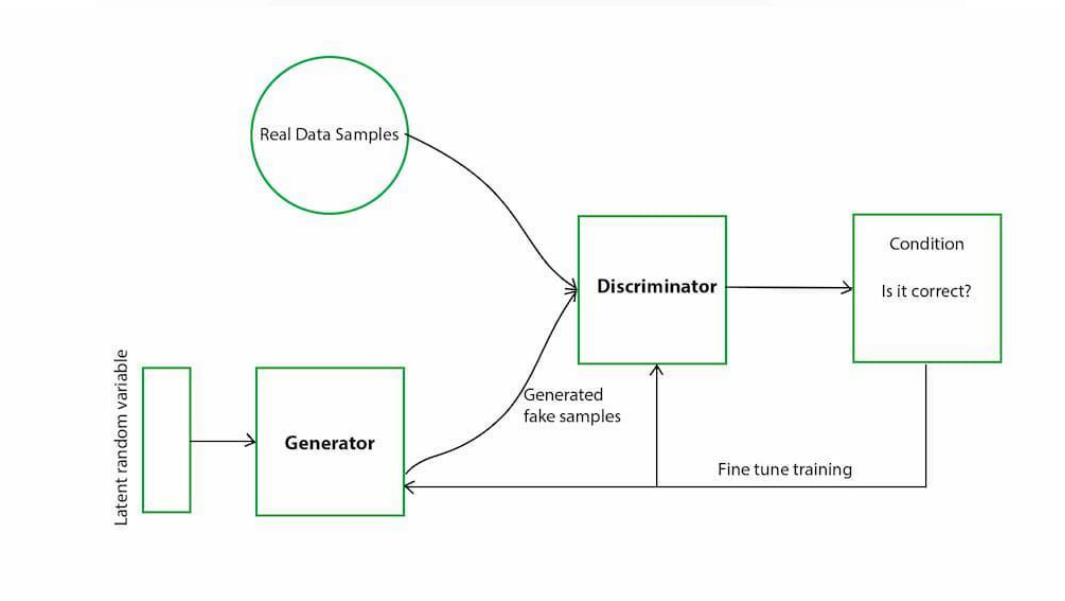
Recursive Neural Network (RvNN) yang dapat dilihat pada gambar 2.4 adalah salah satu contoh jaringan lainnya yang biasa digunakan untuk *natural language processing*, RvNN akan mempelajari informasi yang detail dan terstruktur.



Gambar 2.4 - Arsitektur RvNN[28]

Generative Adversarial Network (GAN) adalah contoh lainnya dari jaringan yang merupakan salah satu pendekatan mutakhir untuk membuat sebuah model generatif menggunakan bantuan *deep learning*, GAN mampu melakukan identifikasi pada pola dalam data yang menjadi *input* secara mandiri dan hal ini memungkinkan model untuk menghasilkan contoh baru yang menyerupai dataset

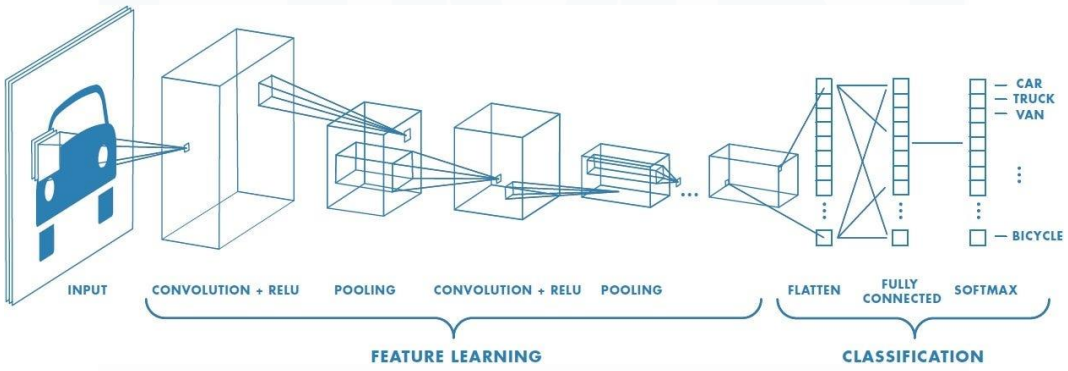
asli, arsitektur dari GAN sendiri dapat dilihat pada gambar 2.5 di bawah ini, dan masih banyak lagi[26].



Gambar 2.5 - Arsitektur GAN[29]

2.4 Convolutional Neural Network

Convolutional Neural Network atau CNN adalah salah satu algoritma *deep learning* yang paling banyak digunakan di dunia. CNN dibangun dengan menggunakan 4 lapisan yakni *convolutional*, *pooling*, *fully connected*, dan *nonlinearity layer*[30].



Gambar 2.6 - Arsitektur CNN[31]

Pada gambar 2.6 dapat dilihat arsitektur dari algoritma CNN dimana lapisan pertama yakni *convolutional layer* mesin akan melihat gambar sebagai matriks dimana angka-angka pada matriks tersebut merepresentasikan intensitas cahaya

pada sebuah poin yang biasa disebut piksel. *Convolutional layer* sendiri didefiniskan melalui tiga faktor yakni ukuran kernel, *stride length*, dan *padding*. Ukuran kernel adalah ukuran *FILTER* dari kernel atau kernel geser, *stride length* adalah jumlah kernel yang bergeser sebelum pembuatan titik produk dan mengeluarkan piksel, sementara itu *padding* adalah ukuran frame yang disusun disekitar peta input fitur[32].

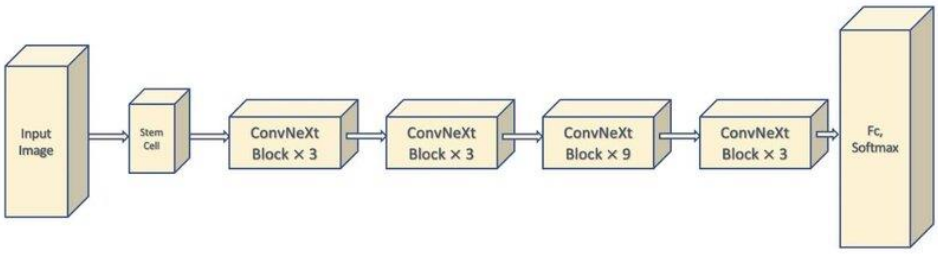
Lapisan kedua dari CNN adalah *pooling layer* yang akan menggabungkan dua *convolutional layer*, *pooling layer* akan mengurangi jumlah parameter dan beban komputasi menggunakan teknik *down-sampling* dan kemudian akan menghasilkan nilai yang sudah dimaksimalkan atau nilai rata-rata, selain itu *pooling layer* juga membantu pengurangan *overfitting* atau bobot perhitungan[32].

Lapisan ketiga dari CNN adalah *fully connected layer* yang biasanya disebut dengan *convolutional output layer* dan biasa ditemukan di bawah sebuah *network*, lapisan ini akan menerima *input* dari *convolutional output layer* dan kemudian dilakukan *flattening* sebelum akhirnya akan dikirim ke lapisan berikutnya[32].

Lapisan terakhir dari CNN yakni lapisan keempatnya adalah *nonlinearity layer* yang memegang peranan penting pada lapisan CNN dimana hasil yang telah di-*filter* akan menghasilkan sebuah perhitungan fungsi matematika yang disebut aktivasi. ReLU (*Rectified Linear Unit*) adalah salah satu contoh aktivasi yang paling banyak digunakan pada model berbasis CNN. Fungsi dari lapisan terakhir ini adalah memutuskan bagaimana *output* dari sebuah *neural network* seperti ya atau tidak. Fungsi aktivasi sendiri dibagi menjadi dua kategori yakni *Linear Activation Function* adalah sebuah ekspresi matematis yang disederhanakan dari fungsi aktivasi linear yang dapat ditulis seperti F(x) = CY dimana nilai *input* akan dikalikan dengan parameter konstan dengan c sebagai bobot tiap *neuron* dan akan menghasilkan *output* sebanding dengan *input* dan *Non-Linear Activation Function* digunakan dalam jaringan syaraf *neuron* dan memungkinkan model untuk *mapping* yang rumit antara *input* dan *output* yang penting untuk sistem pembelajaran dan pemodelan kompleks[32].

2.5 ConvNeXt

ConvNeXt adalah salah satu jenis model *neural network* yang dikembangkan oleh tim pengembang penelitian Facebook menggunakan modul ConvNet standar yang memiliki kinerja kuat di berbagai variasi kondisi[33]. ConvNeXt juga sudah dikembangkan menjadi ConvNeXt v2 dengan mengubah rancangan arsitektur yang dengan menggunakan *masked autoencoder* (MAE) dan lapisan *Global Response Architecture* (GRA). ConvNeXt v2 juga sudah ditingkatkan kinerjanya pada beberapa *benchmark* seperti ImageNet, Coco, dan ADE20K[34]. Contoh dari model ConvNeXt dapat dilihat pada gambar 2.7 di bawah ini.



Gambar 2.7 - Contoh model ConvNeXt[35]

2.6 Python

Python adalah sebuah bahasa pemograman *object-oriented* yang diciptakan oleh Guido Van Rossum pada tahun 1991. Python biasanya digunakan untuk pengembangan situs web pada sisi server, pengembangan *software*, komputasi matematika, analisis data, *machine learning*, dan masih banyak lagi. Python juga dilengkapi dengan *library*, dokumentasi lengkap, dan grup komunitas yang dapat ditemukan di internet sehingga dapat dijadikan panduan bagi pemula yang ingin mempelajari Python[36].

2.7 Tensorflow

Tensorflow adalah sebuah *software library* yang bersifat gratis dan *open source* yang dapat digunakan untuk komputasi numerik dengan performa yang tinggi. Tensorflow memiliki arsitektur yang fleksibel dan hal ini memungkinkan para penggunanya untuk melakukan penerapan komputasi numerik menggunakan

berbagai *platform* seperti CPU, GPU, bahkan TPU. Pengguna dari Tensorflow juga mampu menggunakan berbagai *device* untuk melakukan komputasi dari komputer, server, hingga perangkat seluler[37].

Tensorflow sendiri awalnya dikembangkan oleh tim Google Brain dan ditujukan untuk membantu *machine learning* dan *deep learning* dan mendukungnya dengan menggunakan komputasi numerik yang dapat digunakan pada domain ilmiah lainnya[37].

2.8 Keras

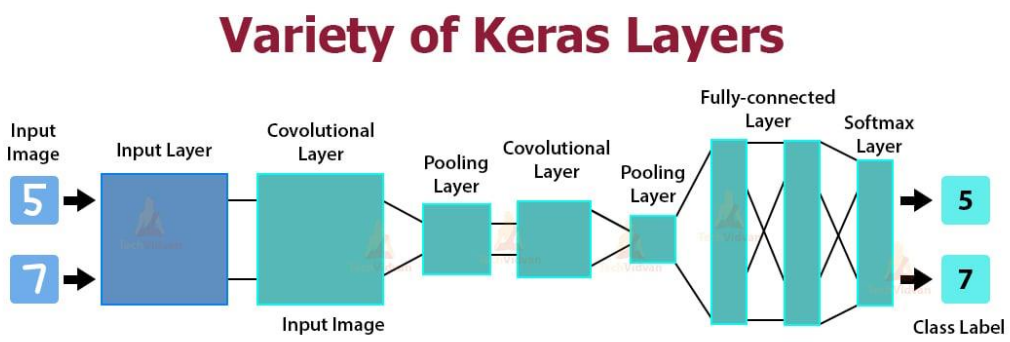
Keras adalah sebuah API dari Tensorflow bertingkat tinggi. Keras juga memiliki tampilan antarmuka yang mudah didekati dan sangat produktif untuk memecahkan masalah yang ada pada *machine learning*. Keras juga memfokuskan *framework*-nya pada *deep learning* modern. Keras juga telah mencakup semua langkah dalam pembelajaran *machine learning* dimulai dari pemrosesan data hingga *deployment*. Pengembangan Keras difokuskan pada kecepatan dalam melakukan eksperimen[38].

Penggunaan Keras memungkinkan penggunanya untuk memiliki skalabilitas dan kemampuannya untuk dapat digunakan secara *cross-platform*, dimana hal ini memungkinkan pengguna Keras untuk mampu menjalankan Keras di pod TPU, GPU, bahkan pengguna Keras juga dapat menjalankannya di perangkat seluler dan juga di *browser* menggunakan API[38].

Keras sendiri dirancang untuk mencapai beberapa tujuan seperti memiliki tampilan antarmuka yang sederhana dan konsisten, meminimalisasi jumlah tindakan yang dibutuhkan dalam *use cases* umum, memberitahu pesan *error* secara detail dan dapat ditindaklanjuti, mudah untuk dipelajari dan pengguna dapat belajar secara bertahap sambil menggunakan Keras, dan membantu pengguna dalam menulis kode yang ringkas dan mudah dibaca[38].

Keras memiliki struktur inti yakni lapisan dan model. Lapisan Keras adalah sebuah transformasi input atau output sederhana yang dapat dipanggil dengan menggunakan tf.Keras.layers.Layer yang menjadi abstraksi fundamental, lapisan

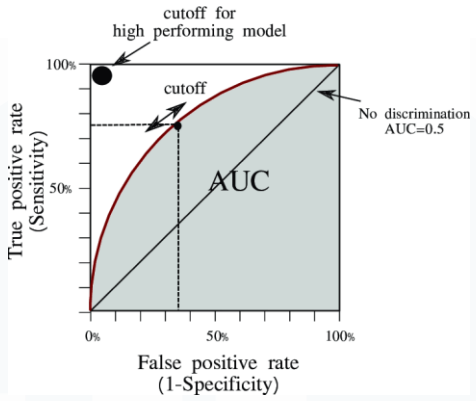
Keras mengenkapsulasi bobot dan beberapa komputasi yang didefinisikan pada metode tf.Keras.layers.Layer.call. Bobot yang dibuat oleh lapisan Keras ini dapat dilatih dan juga tidak dilatih dan lapisan Keras juga dapat disusun secara rekursif. Contoh lapisan dari Keras dapat dilihat pada gambar 2.8. Sementara itu model adalah sebuah objek yang akan menggabungkan lapisan-lapisan Keras dan dapat dilatih menggunakan data. Model Keras yang paling sederhana adalah model *Sequential* yang merupakan sebuah susunan linier dari beberapa lapisan. Model Keras juga memiliki beberapa fungsi seperti *fit* yang dapat digunakan untuk melatih model dengan menggunakan jumlah *epochs* yang telah ditentukan, selain itu ada juga fungsi *predict* yang berguna untuk memberikan *output* berupa prediksi dari sampel *input*, kemudian juga ada fungsi *evaluate* yang dapat digunakan untuk memberikan *output* berupa *loss* dan nilai beberapa metrik yang dapat dikonfigurasikan pada metode *compile*[38].



Gambar 2.8 - Contoh lapisan Keras[39]

2.9 ROC Curve

Kurva ROC atau *Receiver Operating Characteristics Curve* adalah salah satu metriks evaluasi untuk mengukur kemampuan sebuah model. ROC pertama kali digunakan pada saat perang dunia pertama untuk mempelajari kemampuan radar dalam mengenali pesawat dengan benar dengan cara menampilkan pertukaran antara tingkat *true positive rate* (TPR) dan *false positive rate* (FPR)[40]. Contoh kurva ROC dapat dilihat pada gambar 2.9.



Gambar 2.9 - Contoh Kurva ROC[57]

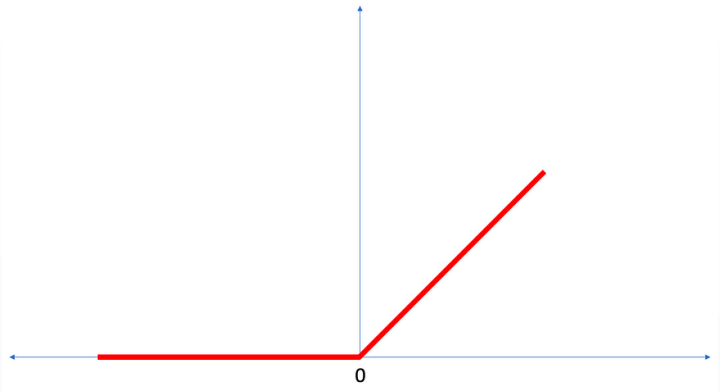
Semakin kurva ROC dekat ke arah kiri maka model semakin bagus dan begitu pula sebaliknya. Area di bawah kurva ROC sendiri disebut AUC atau *Area Under Curve* menjelaskan seberapa bagus sebuah tes tanpa memedulikan *treshold*. Nilai AUC 1 merupakan hasil pengujian yang sempurna sementara nilai 0.5 berarti nilai yang buruk karena sama saja seperti melempar koin dan nilai 0 adalah nilai terburuk karena model salah mengklasifikasi semua hasilnya[41].

2.10 ReLu (Rectified Linear Unit)

ReLu atau *Rectified Linear Unit* adalah salah satu jenis aktivasi yang sering digunakan dalam *deep learning*. Fungsi ini akan mengembalikan nilai 0 jika menerima nilai negatif dalam *input*-nya, namun jika ia menerima nilai positif *x* maka fungsi ini akan mengembalikan nilai tersebut. Fungsi ReLu sendiri dapat ditulis dengan perhitungan matematis seperti di bawah ini[42].

$$\mathbf{f(x) = max(0, x)}$$

Fungsi ReLu juga memiliki grafik seperti yang dapat dilihat pada gambar 2.10. Perhitungan dengan menggunakan ReLu sendiri akan memberikan bantuan kepada jaringan untuk mengatasi masalah non-linier[42].



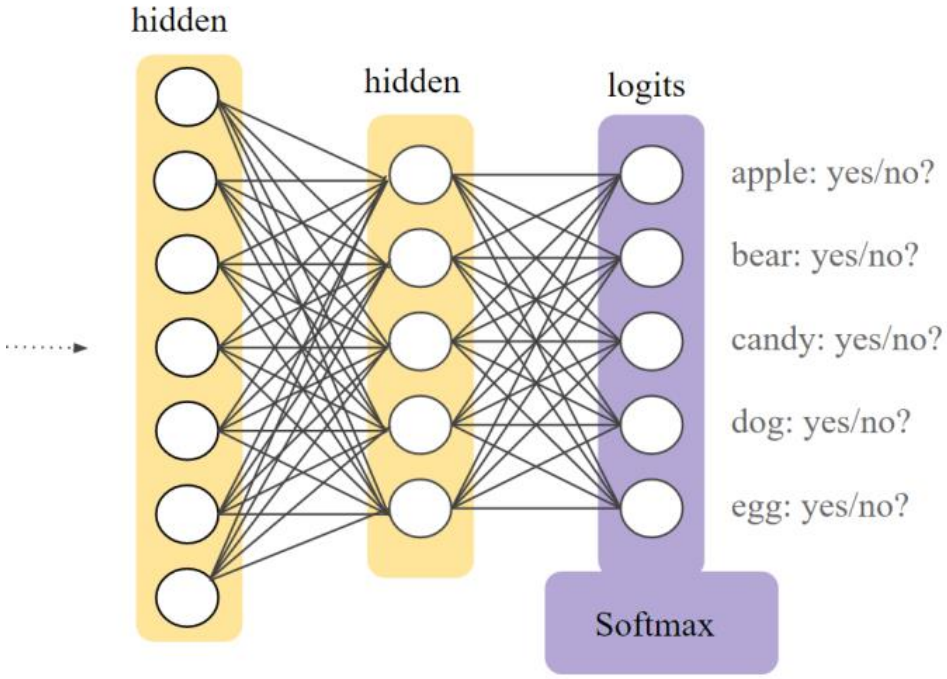
Gambar 2.10 - Grafik Fungsi ReLu[43]

2.11 Softmax

Softmax adalah sebuah fungsi matematika yang akan mengubah nilai yang awalnya berbentuk numerik menjadi sebuah probabilitas yang kemudian akan digunakan dalam klasifikasi *multiclass*. Dalam *machine learning*, softmax biasa digunakan untuk melakukan perhitungan probabilitas perbedaan kelas dari data yang di-*input*. Dengan menggunakan fungsi inilah maka sebuah model dalam *machine learning* maupun *deep learning* dapat melakukan prediksi kelas yang paling memungkinkan dari data yang telah di-*input*. Rumus dari softmax sendiri dapat diinterpretasikan sebagai berikut[44].

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_i}}$$

Simbol-simbol yang digunakan pada rumus softmax dapat dijelaskan sebagai berikut, softmax(x) sebagai fungsi softmax yang diterapkan pada vektor x, x adalah vektor input yang nantinya akan diubah menjadi probabilitas, i menjadi indeks dari elemen vektor x, e sebagai bilangan konstan *euler* yakni 2.71828, n sebagai jumlah elemen dalam vektor x, dan $\sum$ sebagai simbol yang menunjukkan penjumlahan. Lapisan dari softmax sendiri dapat dilihat pada gambar 2.11 di bawah ini[44].



Gambar 2.11 - Lapisan Softmax[45]

2.12 Adam

Adam atau *Adaptive Moment Estimation* adalah salah satu bentuk optimasi yang mengadopsi algoritma Adam. Optimasi Adam sendiri adalah salah satu metode *stochastic gradient descent* yang didasarkan pada estimasi momen orde pertama dan kedua yang adaptif dan kemudian dipadukan dengan RMSprop. Adam dibuat secara khusus untuk membantu pelatihan *deep neural network*. Algoritma Adam sendiri melakukan pengskalaan *learning rate* menggunakan gradien kuadrat dan kemudian memanfaatkan momentum dengan menggunakan *gradient moving average*, hal ini mirip dengan SGD namun dengan momentum[46].

Untuk melakukan perkiraan momentum, Adam akan menggunakan *exponential moving averages* yang akan dihitung berdasarkan gradien dan akan dievaluasi pada *mini batch*. Secara matermatis, perkiraan momentum ini dapat ditulis sebagai berikut:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2$$

Simbol dari rumus matematis tersebut dapat dijelaskan yakni m dan v sebagai *moving averages*, g sebagai nilai dari *gradient*, β atau beta sebagai *hyperparameter* yang memiliki nilai *default* yakni 0.9 atau 0.999. Nilai dari *moments* dan *gradient* harus sama, oleh karena itu harus dilakukan pengambilan nilai rata-rata *moments* yang dapat dilakukan dengan menggunakan rumus matematis seperti di bawah ini[47].

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

Dengan menggunakan rumus diatas maka Adam akan melakukan pembaharuan nilai bobot yang dimana proses ini mirip dengan RMSprop yang rumus matematisnya dapat dilihat pada rumus di bawah ini dengan w sebagai bobot, eta atau η sebagai *learning rate*, dan epsilon yakni nilai yang sangat kecil biasanya 10^{-8} untuk menghindari pembagian dengan nilai nol[47].

$$w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{v_t} + \varepsilon}$$

Perbedaan Adam dalam cara melakukan perhitungan *learning rate* dan momentum lalu memperbaharui nilainya secara otomatis menjadi salah satu keunggulan Adam dibandingkan dengan optimasi lainnya. Dalam beberapa penelitian yang telah dilakukan sebelumnya, Adam terbukti berhasil memberikan hasil yang lebih baik dibandingkan optimasi lain seperti SGD maupun RMSprop. Penelitian yang dilakukan oleh Doni Anggara et al. berhasil membuktikannya dimana dengan menggunakan 60 *epochs* dan *learning rate* 0.001 dengan membandingkan tiga optimasi menghasilkan optimasi Adam dengan akurasi 68,61% disusul oleh optimasi SGD dengan akurasi 57,68% dan RMSprop yang hanya mendapat akurasi 54,83%[48]. Dalam salah satu penelitian, Adam juga sudah digunakan pengembangan model *deep learning* untuk melakukan deteksi parasit

malaria pada citra mikroskopis hapusan darah mendapatkan hasil yang lebih baik dengan akurasi 95,83% dibandingkan dengan optimasi lainnya[49].

2.13 CRISP-DM

CRISP-DM atau *Cross Industry Standard Process for Data Mining* merupakan salah satu standar industri *data mining* dan merupakan model proses yang tidak bergantung pada industri. CRISP-DM memiliki enam langkah dimulai dari *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan terakhir *deployment*[50]. Langkah-langkah tersebut dapat dilihat pada gambar 2.12.

Gambar 2.12 - CRISP-DM[51]

Pada *business understanding* dilakukan pemahaman mengenai situasi bisnis untuk mengetahui sumber daya yang tersedia dan juga dibutuhkan, selain itu tujuan dan persyaratan proyek juga harus dipahami. Setelah itu dilanjutkan ke *data understanding* akan dilakukan pemahaman mengenai data. Kemudian data yang sudah dipahami akan dipersiapkan supaya dapat lanjut kebagian *modeling* seperti *cleaning, transforming*, dan integrasi data. Pada tahap *modeling* akan dilakukan pemilihan dan pembangunan model yang sesuai dengan kebutuhan dari proyek tersebut. Setelah model selesai dibuat maka model harus dievaluasi untuk

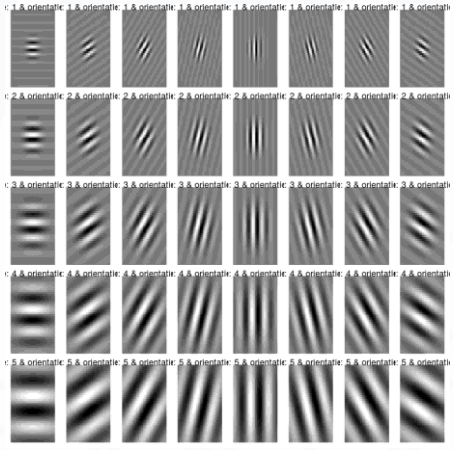
mengetahui performa proyek dengan melihat metrik evaluasi sesuai dengan tujuan dibentuknya model. Model yang sudah selesai dibuat dan dievaluasi kemudian masuk ke tahap *deployment* sehingga model tersebut dapat digunakan untuk menilai data baru dan menjadi solusi bisnis asli[50].

2.14 Gabor

Gabor adalah salah satu metode pra-pemrosesan yang mendapatkan namanya dari Dennis Gabor yang merupakan sebuah *filter* linear yang digunakan untuk menganalisis tekstur dalam tugas pemrosesan gambar. Cara kerja *filter* ini adalah dengan mencari apakah pada sebuah gambar sepanjang orientasi tertentu pada zona lokal di sekitar wilayah analisis memiliki frekuensi tertentu. Beberapa peneliti kontemporer juga mengatakan bahwa frekuensi dan orientasi yang dimiliki oleh *filter* Gabor sama dengan penglihatan manusia walaupun belum ada bukti eksperimen mengenai hal ini. Gabor yang memiliki kemampuan pembedaan dan penyelesaian yang beragam yang berasal dari fungsi Gabor dapat berguna untuk ekstraksi dan analisis fitur tekstur dan tepi gambar[52]. Gabor sendiri memiliki rumus matematis yang dapat dilihat di bawah ini.

$$G(x, \mathrm{y}|\mathrm{W}, \theta, \varphi, \mathrm{X}, \mathrm{Y}) = exp^{\frac{-[(x-X)^2+(y-Y)^2]}{2\sigma^2}} x \sin\left(W(x\cos\theta - y\sin\theta) + \varphi\right).$$

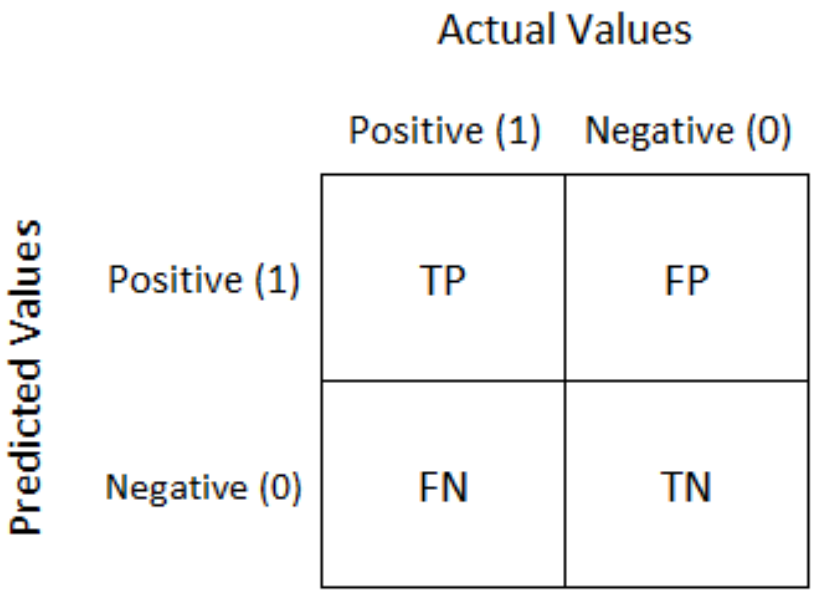
Contoh dari hasil penerapan *filter* Gabor dapat dilihat pada gambar 2.13 di bawah ini.



Gambar 2.13 - Contoh Penerapan Filter Gabor[53]

2.15 Confusion Matrix

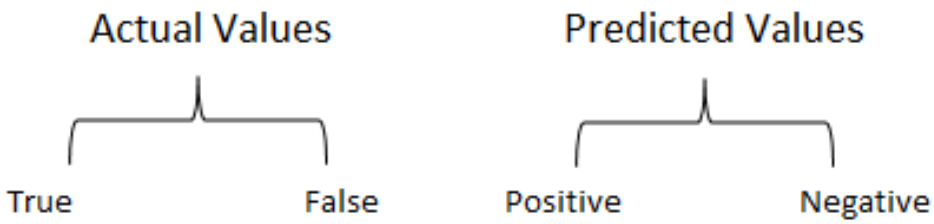
Setelah melakukan seluruh rangkaian proses *machine learning* atau *deep learning* mulai dari pembuatan model sampai pelatihan model maka akan dilakukan evaluasi kepada model. Evaluasi model dapat dilakukan dengan berbagai cara salah satunya adalah dengan menggunakan confusion matrix. Confusion matrix adalah sebuah matriks yang berfungsi sebagai pengukur performa dari sebuah model klasifikasi, prediksi, dan masih banyak lagi. Confusion matrix memiliki *output* dua kelas atau lebih dimana confusion matrix menjadi sebuah tabel dengan empat kombinasi berbeda dan digabungkan antara nilai yang diprediksi maupun nilai aslinya. Contoh dari confusion matrix dapat dilihat pada gambar 2.14 di bawah ini



Gambar 2.14 - Contoh Confusion matrix[54]

Confusion matrix dapat digunakan untuk melakukan perhitungan pada *recall*, *precision*, *accuracy*, dan F1-*Score*. Untuk melakukan perhitungan yang telah disebutkan diatas maka perlu pengertian terlebih dahulu kepada empat analogi pada gambar 2.14 yakni TP, FP, FN, TN. TP atau *True Positive* adalah nilai prediksi yang berupa benar dan nilai aktualnya berupa benar. Selanjutnya adalah TN atau *True Negative* yakni nilai prediksi yang berupa salah namun nilai aktualnya berupa

benar. Kemudian ada FP atau *False Positive* atau merupakan eror tipe 1 yakni nilai yang diprediksi berupa benar namun nilai aktualnya adalah salah. Terakhir adalah FN atau *False Negative* atau eror tipe 2 yakni kondisi dimana nilai yang diprediksi berupa salah namun nilai aktualnya adalah benar. Nilai aktualnya disebut sebagai benar atau salah sementara nilai prediksi akan menggunakan positif dan negatif seperti pada gambar 2.15.



Gambar 2.15 - Nilai Aktual vs Prediksi[54]

Setelah mengetahui empat analogi yang ada pada tabel confusion matrix, maka perhitungan *recall*, *precision*, *accuracy*, dan f1-*score*. Untuk perhitungan pertama yakni *recall* atau dapat disebut *sensitivity* yang menggambarkan keberhasilan sebuah model dalam menemukan sebuah informasi. Rumus dari *recall* dapat dilihat pada perhitungan matematis di bawah ini.

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

Setelah itu, perhitungan *precision* yang menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Rumus dari perhitungan *precision* dapat dilihat di bawah ini.

$$Precision = \frac{TP}{(TP + FP)}$$

Selain *recall* dan *precision*, akurasi juga dapat dihitung menggunakan data yang disediakan oleh confusion matrix. Akurasi akan menggambarkan seberapa akurat

model dalam melakukan klasifikasi data. Rumus dari perhitungan akurasi dapat dilihat di bawah ini.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Setelah itu dapat juga dihitung F1-*Score* yang menggambarkan perbandingan rata-rata antara *precision* dan *recall* yang diberi bobot. Penggunaan F1-*Score* sendiri biasanya digunakan ketika jumlah data FN dan FP tidak mendekati atau simetris namun jika sangat mendekati (simetris) maka penggunaan *accuracy* lebih disarankan. Rumus matematis dari F1-*Score* dapat dilihat di bawah ini.

$$F1 - Score = \frac{(2\ x\ Recall\ x\ Precision)}{(Recall + Precision)}$$