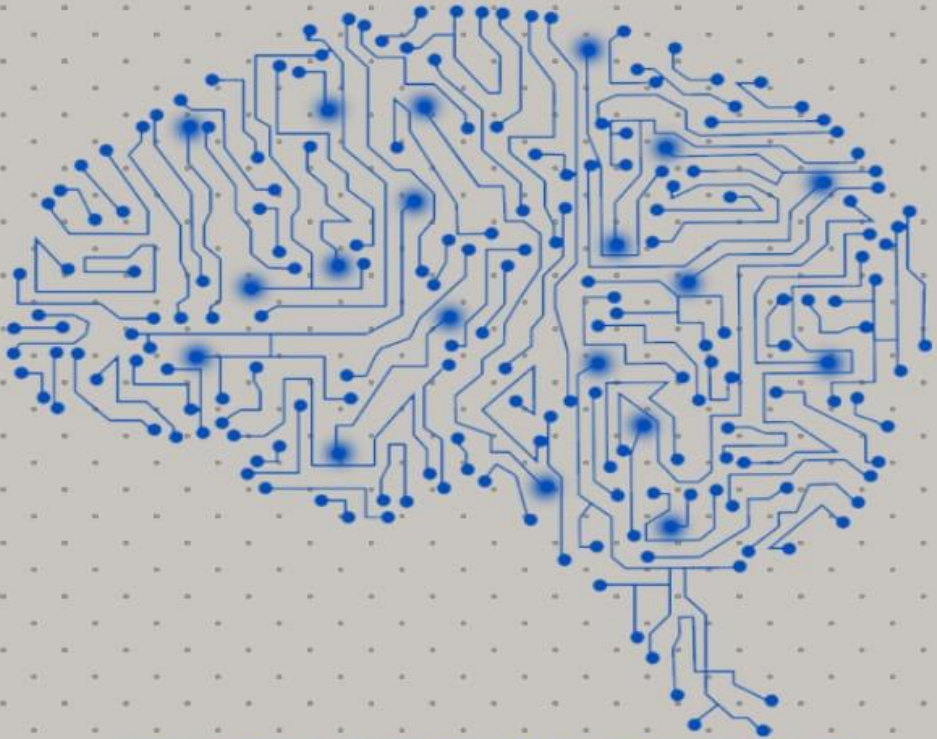




BUKU TUTORIAL KERJA PRAKTEK



BELAJAR KLASIFIKASI GAMBAR MULTICLASS MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)

**PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
2021**

NOVITA ANGGRAINI (180441100093)

LEMBAR KESEPAKATAN
PEMBUATAN BUKU TUTORIAL

“BELAJAR KLASIFIKASI GAMBAR MULTICLASS
MENGUNAKAN METODE CONVOLUTIONAL
NEURAL NETWORK (CNN)”

Nama Mahasiswa : Novita Anggraini

NIM : 180441100093

Program Studi : Sistem Informasi

Jenis : Buku Tutorial

Topik / Judul Buku : Belajar Klasifikasi gambar Multiclass
menggunakan metode Convolutional Neural Network (CNN)

Outline Buku :

BAB 1 : Pengenalan Deep Learning

- Pengertian *DeepLearning*
- Kategori *DeepLearning*
- Parameter *DeepLearning*
- Langkah-langkah evaluasi kinerja model *Deep Learning*

BAB 2 : Pengenalan Tensorflow

- Pengertian Tensorflow
- Sejarah Tensorflow
- Cara Kerja Tensorflow
- Manfaat Tensorflow

BAB 3 : Pengenalan Klasifikasi

- Pengertian Klasifikasi
- Tujuan Klasifikasi
- Metode – Metode Klasifikasi
- Pengertian Klasifikasi Multiclass

BAB 4 : Pengenalan Convolutional Neural Network (CNN)

- Pengertian Convolutional Neural Network
- Tujuan Convolutional Neural Network

BAB 5 : Persiapan Praktek

- Alat dan Bahan
- Petunjuk Penggunaan Google Colab
- Instalasi Library

BAB 6: Klasifikasi gambar Multiclass menggunakan metode Convolutional Neural Network (CNN)

- Langkah – Langkah klasifikasi gambar Multiclass menggunakan metode Convolutional Neural Network (CNN)
- Hasil klasifikasi gambar Multiclass menggunakan metode Convolutional Neural Network (CNN)

Nganjuk, 8 Juni 2021

Pengusul



Novita Anggraini

180441100093

Menyetujui,

Dosen Pembimbing



Dr. Fika Hastarita Rachman, S.T., M.Eng.

NIP. 19830305 200604 2 002

LEMBAR PENGESAHAN

Telah diperiksa dan diuji oleh Pembimbing Kerja Praktek

Pada Tanggal :

Dengan Nilai :

Mengetahui,

Koordinator Kerja Praktek

Menyetujui,

Dosen Pembimbing

Sri Herawati, S.Kom., M.Kom.

NIP. 19830828 200812 2 002

Dr. Fika Hastarita Rachman, S.T., M.Eng.

NIP. 19830305 200604 2 002

KATA PENGANTAR

Puji syukur kehadiran Tuhan Yang Maha Esa yang telah memberikan rahmat dan hidayah-Nya. Sehingga, buku tutorial dengan judul “Belajar Klasifikasi Gambar Multiclass Menggunakan Metode Convolutional Neural Network (CNN)” ini tersusun secara tuntas. Adapun, tujuan dari pembuatan buku tutorial ini untuk menyelesaikan tugas kerja praktik, yang merupakan salah satu syarat untuk memperoleh gelar Strata Satu Jurusan Sistem Informasi Universitas Trunojoyo Madura. terselesainya buku tutorial ini, tidak lepas atas arahan serta bimbingan dari berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih banyak kepada Ibu Sri Herawati S.Kom, M.Kom selaku koordinator kerja Pratik, Ibu Dr. Fika Hastarita Rachman selaku dosen pembimbing yang telah meluangkan waktu untuk membimbing penulis.

Masih banyak kekurangan dalam kepenulisan maupun susunan kalimat dalam buku tutorial ini. Sehingga, diperlukan kritik dan saran yang membangun dari pembaca sebagai acuan dalam proses perbaikan.

Nganjuk, 23 juni 2021

Penulis

DAFTAR ISI

LEMBAR KESEPAKATAN	i
LEMBAR PENGESAHAN.....	iv
KATA PENGANTAR	v
DAFTAR ISI.....	vi
DAFTAR GAMBAR	viii
BAB 1.....	1
Pengenalan Deep Learning.....	1
1.1. Pengertian Deep Learning	1
1.2. Kategori Deep Learning	3
1.3. Parameter DeepLearning	4
1.4. Langkah-langkah evaluasi kinerja model <i>Deep Learning</i>	6
BAB 2.....	11
Pengenalan Tensorflow	11
2.1. Pengertian Tensorflow	11
2.2. Sejarah Tensorflow	12
2.3. Cara Kerja Tensorflow	13
2.4. Manfaat Tensorflow	29
BAB 3.....	32
Pengenalan Klasifikasi	32
3.1. Pengertian Klasifikasi	32
3.2. Tujuan Klasifikasi	33

3.3.	Metode – Metode Klasifikasi	33
3.4.	Pengertian Klasifikasi Multiclass	39
BAB 4.....		40
PENGENALAN CONVOLUTIONAL NEURAL NETWORK (CNN).....		40
4.1.	Pengertian Convolutional Neural Network	40
4.2.	Tujuan Convolutional Neural Network	51
BAB 5.....		52
PERSIAPAN PRAKTEK.....		52
5.1.	Alat dan Bahan	52
5.2.	Petunjuk Penggunaan Google Colabs	52
5.3.	Instalasi Library.....	59
BAB 6.....		60
KLASIFIKASI GAMBAR MULTICLASS MENGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN).....		60
6.1.	Langkah – Langkah klasifikasi gambar Multiclass menggunakan metode Convolutional Neural Network (CNN) 60	
6.2.	Hasil klasifikasi gambar Multiclass menggunakan metode (CNN).....	73
BAB 7.....		74
PENUTUP.....		74
7.1.	Kesimpulan	74
7.2.	Saran.....	74
DAFTAR PUSTAKA		76

DAFTAR GAMBAR

Gambar 1.1 Proses Deep Learning.....	1
Gambar 2.1 Logo Tensorflow	11
Gambar 2.2 Tampilan Tensorboard	17
Gambar 2.3 Proses Natural Structured Learning.....	21
Gambar 2.4 Hierarki toolkit Tensorflow.....	29
Gambar 3.1 Klasifikasi.....	32
Gambar 3.2 Klasifikasi Naïve Bayes	33
Gambar 3.3 Support vectore machine	35
Gambar 3.4 Decission tree	36
Gambar 3.5 Struktur Neural Network	37
Gambar 3.6 Before after klasifikasi KNN.....	38
Gambar 3.7 Klasifikasi Multiclass	39
Gambar 4.1 Cara kerja CNN	11
Gambar 4.2 Perbedaan Gambar RGB	42
Gambar 4.3 Macam actifation function.....	44
Gambar 4.4 Feature map 4x4	45
Gambar 4.5 macam pooling	46
Gambar 4.6 Fully connected layer	47
Gambar 4.7 Proses CNN	47
Gambar 5.1 Tampilan Google Colab	54
Gambar 5.2 Tampilan Notebook.....	55

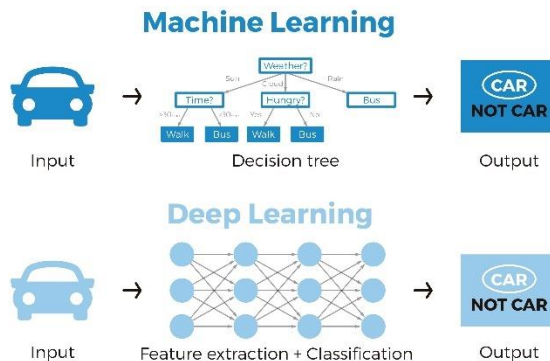
Gambar 5.3 Tampilan Notebook setting	55
Gambar 5.4 Perintah untuk menggabungkan google drive	56
Gambar 5.5 Tampilan Output Autentikasi Google drive .	56
Gambar 5.6 Tampilan jika perintah berhasil	57
Gambar 5.7 perintah upload file.....	58
Gambar 5.8 Tampilan tema yang berbeda.....	58
Gambar 5.9 contoh perintah pip.....	58
Gambar 5.10 contoh perintah instalasi library	60
Gambar 6.1 Import library dan package.....	62
Gambar 6.2 Class name.....	62
Gambar 6.3 Load data 1	63
Gambar 6.4 Load data 2	63
Gambar 6.5 Pengacakan train dan test data.....	64
Gambar 6.6 Eksplorasi data 1	64
Gambar 6.7 Eksplorasi data 2	64
Gambar 6.8 Eksplorasi data 3	65
Gambar 6.9 Normalisasi pixel.....	65
Gambar 6.10 Menampilkan 1 gambar acak.....	66
Gambar 6.11 Menampilkan 25 gambar acak 1.....	66
Gambar 6.12 Menampilkan 25 gambar acak 2.....	67
Gambar 6.13 Membuat model block 1-4	68
Gambar 6.14 Membuat model blok 5, FC layer, output layer dan compile	68

Gambar 6.15 Membuat model implement callbacks dan train	68
Gambar 6.16 Output layer model 1	69
Gambar 6.17 Output layer model 2	69
Gambar 6.18 Epoch 1-5	70
Gambar 6.19 Epoch 6-10	70
Gambar 6.20 Membuat fungsi plot accuracy dan loss	70
Gambar 6.21 Memanggil fungsi plot accuracy dan loss	71
Gambar 6.22 plot accuracy dan loss.....	71
Gambar 6.23 evaluasi model	71
Gambar 6.24 Menyimpan model.....	71
Gambar 6.25 Membuat classification report	72
Gambar 6.26 output classification report	72
Gambar 6.27 Membuat confusion matrix	72
Gambar 6.28 Membuat plot confusion matrix.....	73
Gambar 6.29 Plot confusion matrix 1	73
Gambar 6.30 Plot confusion matrix 2.....	74
Gambar 6.31 Hasil klasifikasi gambar dengan cnn.....	74

BAB 1

PENGENALAN DEEP LEARNING

1.1. Pengertian Deep Learning



Gambar 1.1 Proses Deep Learning

Deep Learning adalah kelas algoritma machine learning yang memanfaatkan beberapa lapisan secara progresif untuk mengekstrak fitur tingkat yang lebih tinggi dari input mentah[4]. Misalnya, pada pemrosesan gambar, lapisan bawah bisa mengetahui tepi, sedangkan lapisan yang lebih tinggi bisa mengetahui konsep yang sesuai dengan manusia seperti huruf, wajah atau angka.

Intinya *Deep Learning* adalah sebuah model yang dapat mempelajari teknik komputasinya sendiri dengan 'otaknya' sendiri. Deep learning adalah metode machine learning yang belajar dengan cara meniru prosedur sistem dasar otak manusia beroperasi[7]. Sistem dasar otak manusia yang beroperasi ini disebut neural networks. Itulah mengapa deep learning disebut memanfaatkan artificial neural networks dengan kata lain memakai neural networks buatan. Deep learning merupakan teknologi yang digunakan pada computer vision dan image recognition.

Istilah *Deep Learning* diperkenalkan pertama kali ke komunitas *Machine Learning* pada tahun 1986 oleh Rina Dechter. Tahun 2009, Nvidia, perusahaan teknologi asal Amerika turut campur dalam "big bang" dari *deep learning*, *deep learning* dengan *neural training* terus dilatih oleh Nvidia *graphics processing units* (GPUs). Di tahun yang sama, untuk menciptakan *deep neural network* (DNN). Nvidia GPU dipakai oleh Google Brain. *Deep learning* dapat sebagai kepingan puzzle utama yang bisa mendorong terciptanya AI yang lebih cerdas dan manusiawi. Semua bagian AI dapat di

tingkatkan menggunakan *Deep Learning*, mulai dari pengoperasian bahasa alami sampai machine vision.

Artificial Neural Network (ANN) digunakan dalam *Deep learning* untuk memproses informasi yang didasarkan pada susunan dan tindakan jaringan saraf biologis pada otak [4]. ANN atau singkatan dari *Artificial Neural Networks* adalah bagian yang paling unik dari deep learning. Kinerja otak manusia yang terdiri dari jaringan saraf yang disebut neuron di ilustrasikan dalam ANN. Sama halnya pada sistem otak manusia, Titik-titik yang terdapat informasi yang disebut nodes pada susunan artificial neural network yang terhimpun pada satu layer yang selanjutnya diproses ke layer berikutnya yang disebut hidden layers.

1.2. Kategori Deep Learning

Secara umum kategori *deep learning* terbagi menjadi tiga, yaitu:

1) Deep Networks for Unsupervised Learning

Kategori ini mengetahui koneksi tingkat tinggi dari data yang terlihat saat ditinjau untuk menganalisis pola atau objek sintesis ketika tidak

ada informasi mengenai label kelas target yang ada.

2) Deep Networks for Supervised Learning

Kategori ini secara langsung memiliki kekuatan diskriminatif yang memiliki tujuan klasifikasi pola, biasanya dengan melakukan karakterisasi proses distribusi posterior kelas yang kemudian disesuaikan pada data yang ada. Data label target selalu tersaji dalam bentuk tidak langsung atau langsung. Kategori ini juga disebut dengan *discriminative deep networks*.

3) Hybrid Deep Networks

Kategori ini memiliki tujuan diskriminasi yang dibantu secara signifikan, dengan hasil *unsupervised deep networks* atau *generative*. Tujuan lainnya dicapai jika kriteria diskriminatif untuk *supervised learning* yang dipakai untuk mentaksir parameter pada salah satu *unsupervised deep networks* atau *deep generative* pada kategori di atas.

1.3. Parameter DeepLearning

Parameter yang digunakan pada deep learning adalah sebagai berikut:

- Hidden Layer

Hidden Layer digunakan untuk memastikan jumlah *neuron* dan jumlah lapisan tersembunyi di setiap lapisan arsitektur deep learning[4]. Informasi input menuju lapisan hidden layer pertama dan output dari lapisan ini ditransfer menjadi masukan ke lapisan hidden layer kedua dan seterusnya. Setiap output lapisan sebelumnya akan menjadi input lapisan selanjutnya, sehingga sinyal input merambat maju pada basis lapis tiap lapis hingga lapisan output.

- Epochs

Epochs digunakan untuk menentukan jumlah perulangan yang akan dilaksanakan pada set data[4]. Satu siklus proses algoritma *deep learning* yang mempelajari seluruh *training dataset* disebut dengan *Epochs* . Satu *epochs* berarti menunjukkan *training dataset* secara keseluruhan telah dipelajari sebuah algoritma *deep learning* .

- Learning rate

Learning rate adalah nilai koreksi bobot ketika proses training yang dihitung menggunakan satu

parameter training[4]. Nol (0) sampai (1) adalah range pada nilai learning rate ada. Proses training akan berjalan cepat jika nilai learning rate semakin besar. Nilai learning rate semakin tinggi, jika ketelitian jaringan semakin berkurang, namun berlaku sebaliknya, apabila semakin rendah learning rate-nya, maka semakin bertambah ketelitian jaringannya dengan akibat akan memakan waktu untuk proses training yang semakin lama.

- Evaluate Performance

Evaluate Performance digunakan untuk mengevaluasi performa kinerja model berdasar pada learning rate dan parameter epochs yang telah diatur hingga mendapatkan hasil dari performa model yang paling ideal dengan berdasarkan TPR/sensitivity, tingkat akurasi dan Kurva Receiver Operating Characteristic (ROC)[4].

1.4. Langkah-langkah evaluasi kinerja model *Deep*

Learning

Berdasarkan Pandey (2017), dalam evaluasi kinerja model Deep learning terdapat 4 langkah, yaitu:

- Data Integration

Data integration adalah metode untuk penggabungan 2 (dua) data atau lebih dari basis data atau sumber data yang berbeda dalam 1 (satu) basis data. Tujuan data integration adalah untuk mempermudah memproses analisis data untuk pengambilan keputusan serta untuk mengindari adanya redudansi data.

- Data cleaning

Data cleaning adalah teknik yang dipakai antara lain untuk menemukan missing value (nilai yang hilang dari data) dan memperbaiki inkonsisten data supaya hasil dari model cukup baik. Data cleaning penting karena proses ini meningkatkan kualitas data yang juga berpengaruh terhadap produktivitas kerja secara keseluruhan. Data yang tidak akurat akan berpengaruh buruk terhadap akurasi dan performa model. Saat melakukan proses data cleaning, membuang data dan informasi yang tidak dibutuhkan sehingga akan mendapatkan data yang berkualitas. Data yang akurat dan berkualitas akan berpengaruh positif terhadap proses pengambilan keputusan. Dalam

proses cleaning data ada beberapa hal yang perlu diperhatikan, yaitu:

1) Consistency Format

Sebuah variabel mungkin tidak memiliki format yang konsisten seperti penulisan tanggal 10-Okt-2020 versus 10/10/20. Format jam yang berbeda seperti 17.10 versus 5.10 pm. Penulisan uang seperti 17000 versus Rp 17.000. Data dengan format berbeda tidak akan bisa diolah oleh model machine learning. Solusinya, format data harus disamakan dan dibuat konsisten terlebih dahulu.

2) Skala Data

Jika sebuah variabel memiliki jangka dari 1 sampai 100, pastikan tidak ada data yang lebih dari 100. Untuk data numerik, jika sebuah variabel merupakan bilangan positif, maka pastikan tidak ada bilangan negatif.

3) Duplikasi Data

Data yang memiliki duplikat akan mempengaruhi model machine learning, apalagi jika data duplikat tersebut besar

jumlahnya. Untuk itu kita harus memastikan tidak ada data yang terduplikasi.

4) Missing Value

Missing value terjadi ketika data dari sebuah record tidak lengkap. Missing value sangat mempengaruhi performa model machine learning. Ada 2 (dua) opsi untuk mengatasi missing value, yaitu menghilangkan data missing value atau mengganti nilai yang hilang dengan nilai lain, seperti rata-rata dari kolom tersebut (mean) atau nilai yang paling sering muncul (modus), atau nilai tengah (median).

5) Skewness Distribution

Skewness adalah kondisi di mana dataset cenderung memiliki distribusi data yang tidak seimbang. Skewness akan mempengaruhi data dengan menciptakan bias terhadap model. Apa itu bias? Sebuah model cenderung memprediksi sesuatu karena ia lebih sering mempelajari hal tersebut. Misalkan ada sebuah model untuk pengenalan buah di mana jumlah jeruk 92 buah dan apel 8 buah. Distribusi yang tidak

imbang ini akan mengakibatkan model lebih cenderung memprediksi jeruk daripada apel. Cara paling mudah untuk mengatasi skewness adalah dengan menyamakan proporsi kelas mayoritas dengan kelas minoritas.

- Data reduction

Data reduction adalah metode yang digunakan untuk mereduksi representasi dari dataset. Tujuan Data reduction adalah untuk menghandel keterbatasan penyimpanan data pada database/data warehouse serta mengatasi lamanya waktu yang diperlukan untuk menganalisa data yang kompleks pada keseluruhan dataset.

- Feature selection

Feature selection adalah Teknik yang digunakan untuk menentukan subset fitur yang optimal setara dengan kriteria tertentu. Teknik ini bertujuan untuk menghapus fitur yang tidak relevan dan berlebihan serta mereduksi jumlah fitur dalam model. Feature selection memiliki tujuan untuk mengoptimalkan kinerja (kesederhanaan model, daya prediksi dan interval kecepatan) serta menurunkan dimensi.

BAB 2

PENGENALAN TENSORFLOW

2.1. Pengertian Tensorflow



Gambar 2.1 Logo Tensorflow

TensorFlow merupakan pustaka software open source & gratis yang diperuntukkan pemelajaran mesin[1]. Pustaka ini bisa dipakai sebagai aneka macam keperluan namun mempunyai penekanan spesifik dalam pembinaan & inferensi jaringan saraf dalam (Deep neural networks). Tensorflow adalah pustaka matematika simbolis yang dibentuk menurut dataflow & pemrograman diferensial[10].

2.2. Sejarah Tensorflow

TensorFlow merupakan sistem generasi ke 2 Google Brain. Versi 1.0.0 dirilis dalam 11 Februari 2017. Meskipun implementasi prototipe berjalan dalam satu perangkat, TensorFlow bisa berjalan pada beberapa CPU & GPU (menggunakan perluasan CUDA & SYCL opsional buat komputasi tujuan general dalam unit pemrosesan grafis). TensorFlow dapat digunakan pada Linux 64-bit , macOS , Windows, & platform komputasi seluler termasuk Android & iOS .

Nama TensorFlow berdasar menurut operasi yang dilaksanakan jaringan saraf sejenis itu dalam baris data multidimensi, yang diklaim menjadi tensor . Selama Pertemuan Google I/O yang dilaksanakan Juni 2016, Jeff Dean mengutarakan bahwa 1.500 repositori pada GitHub menyuarakan TensorFlow, yang mana hanya terdapat lima yang bersumber menurut Google. Pada bulan Desember 2017, pengembang yang berasal dari Google, RedHat, CaiCloud, Cisco, & CoreOS melansir Kubeflow pada sebuah Pertemuan. Kubeflow memungkinkan operasionalisasi & implementasi TensorFlow pada Kubernetes .

Pada bulan Maret 2018, Google merilis TensorFlow.js versi 1.0 untuk pembelajaran mesin di JavaScript . Pada Januari 2019, Google mengumumkan TensorFlow 2.0. Ini menjadi rilis secara resmi pada September 2019. Pada Mei 2019, Google mengumumkan TensorFlow Graphics untuk pembelajaran mendalam dalam grafik komputer.

2.3. Cara Kerja Tensorflow

TensorFlow memungkinkan developer menciptakan grafik aliran struktur data yang menggambarkan pergerakan data berdasarkan grafik, atau berdasarkan rentetan node pemrosesan [1]. Setiap node pada grafik terdapat prosedur matematika, dan setiap relasi atau batas antar node adalah baris data multidimensi, atau tensor.

Nama TensorFlow didasarkan pada **tensor**, yang merupakan array dari dimensi arbitrer [10]. Dengan menggunakan TensorFlow, Anda dapat memanipulasi tensor dengan sejumlah dimensi yang sangat tinggi. Dengan demikian, Anda akan banyak berurusan dengan satu atau beberapa tensor berdimensi rendah berikut:

- **skalar** adalah array berdimensi 0 (tensor dengan urutan 0). Misalnya, "Howdy" atau 5
- **vektor** adalah array berdimensi 1 (tensor dengan urutan 1). Misalnya, [2, 3, 5, 7, 11] atau [5]
- **matriks** adalah array berdimensi 2 (tensor dengan urutan 2). Misalnya, [[3.1, 8.2, 5.9][4.3, -2.7, 6.5]]

Operasi TensorFlow membuat, menghancurkan, dan memanipulasi tensor. Sebagian besar baris kode dalam kebanyakan program TensorFlow merupakan operasi.

Sebuah **grafik** TensorFlow (juga dikenal sebagai **grafik komputasional** atau **grafik dataflow**) memang merupakan struktur data grafik. Simpul grafik merupakan operasi (di TensorFlow, setiap operasi dikaitkan dengan grafik). Banyak program TensorFlow terdiri dari satu grafik, tetapi program TensorFlow dapat secara opsional membuat beberapa grafik. Simpul grafik merupakan operasi; edge grafik merupakan tensor. Tensor mengalir melalui grafik, dimanipulasi pada setiap simpul oleh operasi. Tensor keluaran dari sebuah operasi biasanya menjadi tensor masukan pada operasi berikutnya. TensorFlow

menerapkan **model eksekusi lambat**, ini berarti simpul hanya dihitung jika diperlukan, berdasarkan kebutuhan simpul yang dikaitkan.

Tensor dapat disimpan dalam grafik sebagai **konstanta** atau **variabel**. Seperti yang Anda ketahui, konstanta berisi tensor yang nilainya tidak berubah, sementara variabel berisi tensor yang nilainya dapat berubah. Namun, hal yang mungkin tidak Anda ketahui adalah bahwa konstanta dan variabel hanyalah operasi lain dalam grafik tersebut. Konstanta merupakan operasi yang selalu menampilkan nilai tensor yang sama. Variabel merupakan operasi yang akan menampilkan tensor mana saja yang telah ditetapkan ke variabel tersebut.

TensorFlow merancang ini untuk programmer melalui Bahasa pemrograman Python. Python merupakan bahasa pemrograman yang mudah dipelajari serta diimplementasikan, dan memperiapkan cara yang mudah untuk mengimplementasikan abstraksi tingkat tinggi agar bisa diaplikasikan bersama. Tensor dan node pada TensorFlow merupakan objek Python, dan TensorFlow itu sendiri adalah perangkat Python.

Prosedur matematika yang sesungguhnya, bagaimanapun, tidak hanya dioperasikan dengan Python. Library yang disuguhkan oleh TensorFlow ditulis menggunakan biner C ++ yang mempunyai kinerja yang tinggi. Python hanya sebagai pengatur lalu lintas antar komponen, dan menciptakan generalisasi pemrograman kelas tinggi untuk merelasikannya.

Komponen Tensorflow ada 2 yaitu:

- 1) Buffer protokol grafik
- 2) Waktu untuk memproses grafik (terdistribusi)

Komponen diatas memiliki sifat analog terhadap penafsir Python dan script Python. Sama halnya kode python yang dijalankan pada beberapa platform hardware yang merupakan implementasi dari penafsir Python, grafik tensorflow bisa dijalankan di beberapa platform hardware, seperti TPU, GPU, dan CPU.

Dalam pemecahan suatu masalah bisa menggunakan API berdasarkan tingkat abstraksi. Semakin tinggi Tingkat abstraksi maka akan lebih mudah digunakan, tetapi akan kurang fleksibel. Di sisi

desain. Sebaiknya mulai dari memastikan semuanya berfungsi dari API tingkat tertinggi. Bisa berpindah ke tingkat yang lebih rendah jika beberapa masalah pemodelan khusus memerlukan fleksibilitas tambahan. Setiap API yang digunakan pada tingkat harus diperhatikan, sehingga akan cukup mudah dalam menurunkan hierarki.

Berikut ini merupakan library dan extensi pada Tensorflow:

1) TensorBoard



Gambar 2.2 Tampilan Tensorboard

TensorBoard Merupakan Rangkaian alat visualisasi untuk memahami, men-debug, dan mengoptimalkan program TensorFlow.

TensorBoard menyediakan visualisasi dan fitur yang diperlukan untuk eksperimen machine learning:

- Melacak dan memvisualisasikan metrik seperti kehilangan dan akurasi
- Memvisualisasikan grafik model (ops dan layer)
- Melihat histogram bobot, bias, atau tensor lain saat berubah seiring waktu
- Memproyeksikan embeddings ke ruang dimensi yang lebih rendah
- Menampilkan gambar, teks, dan data audio
- Membuat profil program TensorFlow

2) Tensorflo Hub

TensorFlow Hub adalah repositori dari machine learning model terlatih yang tersaji untuk dikembangkan sesuai keinginan dan dapat diterapkan di manapun. Model yang sudah terlatih dapat digunakan Kembali, contohnya Faster R-CNN dan BERT .

3) Model Optimization

Toolkit Pengoptimalan Model TensorFlow adalah seperangkat alat untuk mengoptimalkan model ML untuk penerapan dan eksekusi. Di antara banyak kegunaan, toolkit mendukung teknik yang digunakan untuk:

- Kurangi biaya latensi dan inferensi untuk perangkat cloud dan edge (mis. Seluler, IoT).
- Terapkan model ke perangkat edge dengan batasan pada pemrosesan, memori, konsumsi daya, penggunaan jaringan, dan ruang penyimpanan model.
- Aktifkan eksekusi dan optimalkan perangkat keras yang ada atau akselerator tujuan khusus baru.

4) Tensorflow Federated

TensorFlow Federated (TFF) adalah framework open source untuk machine learning dan komputasi lain pada data yang didesentralisasi. TFF telah dikembangkan untuk memfasilitasi penelitian dan eksperimen terbuka dengan Federated Learning (FL) , sebuah pendekatan

pembelajaran mesin di mana model global bersama dilatih di banyak klien yang berpartisipasi yang menyimpan data pelatihan mereka secara lokal. Misalnya, FL telah digunakan untuk melatih model prediksi untuk keyboard seluler tanpa mengunggah data pengetikan sensitif ke server.

TFF memungkinkan pengembang untuk mensimulasikan algoritma pembelajaran federasi yang disertakan pada model dan data mereka, serta bereksperimen dengan algoritma baru. Peneliti akan menemukan titik awal dan contoh lengkap untuk berbagai jenis penelitian. Blok penyusun yang disediakan oleh TFF juga dapat digunakan untuk mengimplementasikan komputasi non-pembelajaran, seperti analitik federasi . Antarmuka TFF diatur dalam dua lapisan utama:

a) Federated Learning (FL) API

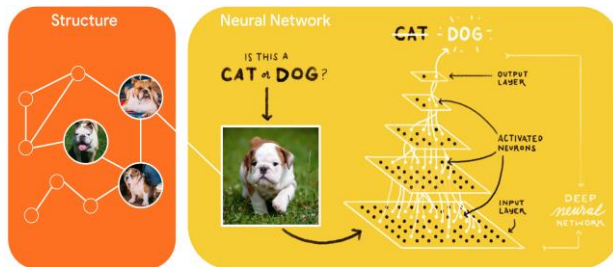
Lapisan ini menawarkan sekumpulan antarmuka tingkat tinggi yang memungkinkan developer untuk menerapkan implementasi pelatihan federasi dan evaluasi

yang disertakan ke model TensorFlow yang ada.

b) Federated Core (FC) API

Inti dari sistem ini adalah sekumpulan antarmuka tingkat rendah untuk mengekspresikan algoritme federasi baru secara ringkas dengan menggabungkan TensorFlow dengan operator komunikasi terdistribusi dalam lingkungan pemrograman fungsional yang diketik dengan kuat. Lapisan ini juga berfungsi sebagai fondasi di mana kita telah membangun Pembelajaran Federasi.

5) Neural Structured Learning



Gambar2.3 Proses Neural Structured Learning

Neural Structured Learning (NSL) adalah paradigma pembelajaran baru untuk melatih jaringan saraf dengan memanfaatkan sinyal terstruktur selain masukan fitur. Struktur dapat eksplisit seperti yang diwakili oleh grafik atau implisit yang disebabkan oleh gangguan permusuhan.

Sinyal terstruktur biasanya digunakan untuk merepresentasikan hubungan atau kesamaan di antara sampel yang mungkin diberi label atau tidak berlabel. Oleh karena itu, memanfaatkan sinyal ini selama pelatihan jaringan neural memanfaatkan data berlabel dan tidak berlabel, yang dapat meningkatkan keakuratan model, terutama bila jumlah data berlabel relatif kecil . Selain itu, model yang dilatih dengan sampel yang dihasilkan dengan menambahkan gangguan permusuhan telah terbukti kuat terhadap serangan jahat , yang dirancang untuk menyesatkan prediksi atau klasifikasi model.

NSL menggeneralisasi Pembelajaran Grafik Neural serta Pembelajaran Adversarial .

Framework NSL di TensorFlow menyediakan API dan fitur yang mudah digunakan berikut bagi developer untuk melatih model dengan sinyal terstruktur:

- Keras API untuk mengaktifkan pelatihan dengan grafik (struktur eksplisit) dan gangguan permusuhan (struktur implisit).
- Operasi dan fungsi TF untuk mengaktifkan pelatihan dengan struktur saat menggunakan TensorFlow API level yang lebih rendah
- Alat untuk membuat grafik dan membuat input grafik untuk pelatihan

6) Tensorflow Graphics

TensorFlow Graphics bertujuan membuat fungsi grafik yang berguna dapat diakses secara luas oleh komunitas dengan menyediakan serangkaian lapisan grafik yang dapat dibedakan (misalnya kamera, model reflektansi, konvolusi mesh) dan fungsi penampil 3D (misalnya Papan Tensor 3D)

yang dapat digunakan dalam model pembelajaran mesin Anda. pilihan.

Beberapa tahun terakhir telah melihat peningkatan lapisan grafis baru yang dapat dibedakan yang dapat dimasukkan ke dalam arsitektur jaringan saraf. Dari transformator spasial hingga penyaji grafis yang dapat dibedakan, lapisan baru ini memanfaatkan pengetahuan yang diperoleh selama bertahun-tahun dari visi komputer dan penelitian grafis untuk membangun arsitektur jaringan yang baru dan lebih efisien. Memodelkan prior dan batasan geometris secara eksplisit ke dalam model pembelajaran mesin membuka pintu ke arsitektur yang dapat dilatih dengan kuat, efisien, dan yang lebih penting, dengan cara yang diawasi sendiri.

7) Tensorflow Datasets

Tensorflow datasets merupakan kumpulan set data yang siap digunakan dengan TensorFlow. TFDS menyediakan kumpulan set data yang siap digunakan untuk digunakan dengan TensorFlow, Jax, dan framework Machine Learning lainnya.

8) Tensorflow Serving

TensorFlow serving adalah sistem penyajian berperforma tinggi yang fleksibel untuk model pembelajaran mesin, yang dirancang untuk lingkungan produksi. TensorFlow Serving memudahkan penerapan algoritme dan eksperimen baru, sekaligus mempertahankan arsitektur server dan API yang sama. TensorFlow Serving menyediakan integrasi yang siap pakai dengan model TensorFlow, tetapi dapat dengan mudah diperluas untuk menyajikan jenis model dan data lainnya.

9) Tensorflow Probability

TensorFlow Probability (TFP) adalah library Python yang dibangun di atas TensorFlow yang memudahkan penggabungan model probabilistik dan deep learning pada hardware modern (TPU, GPU). Ini untuk ilmuwan data, ahli statistik, peneliti ML, dan praktisi yang ingin menyandikan pengetahuan domain untuk memahami data dan membuat prediksi. TFP meliputi:

- Berbagai pilihan distribusi probabilitas dan bijector.
- Alat untuk membangun model probabilistik yang dalam, termasuk lapisan probabilistik dan abstraksi ``JointDistribution``.
- Variasi inferensi dan rantai Markov Monte Carlo.
- Pengoptimal seperti Nelder-Mead, BFGS, dan SGLD.

10) MLIR

MLIR menyatukan infrastruktur untuk model ML berperforma tinggi di TensorFlow. Proyek MLIR mendefinisikan representasi menengah umum (IR) yang menyatukan infrastruktur yang diperlukan untuk menjalankan model machine learning berperforma tinggi di TensorFlow dan framework ML serupa. Proyek ini akan mencakup penerapan teknik HPC, bersama dengan integrasi algoritma pencarian seperti pembelajaran penguatan. MLIR bertujuan untuk mengurangi biaya untuk menghadirkan perangkat keras baru,

dan meningkatkan kegunaan bagi pengguna TensorFlow yang sudah ada.

11) XLA

Kompiler khusus domain untuk aljabar linier yang mempercepat model TensorFlow dengan kemungkinan tidak ada perubahan kode sumber.

12) SIG Addons

Fungsionalitas tambahan untuk TensorFlow, dikelola oleh SIG Addons. Tensorflow Addons adalah kontribusi pola API yang sudah siap pada sebuah repositori, tetapi perlu untuk mengembangkan fungsionalitas jika pada TensorFlow core tidak tersedia. TensorFlow secara native mendukung sejumlah besar operator, lapisan, metrik, kerugian, dan pengoptimal. Namun, dalam bidang yang bergerak cepat seperti ML, ada banyak perkembangan baru yang menarik yang tidak dapat diintegrasikan ke dalam inti TensorFlow (karena penerapannya yang luas belum jelas, atau sebagian besar digunakan oleh sebagian kecil komunitas).

13) SIG IO

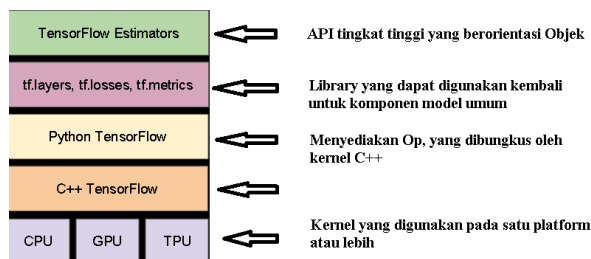
Ekstensi kumpulan data, streaming, dan sistem file, dikelola oleh SIG IO.

TensorFlow dapat diaplikasikan di nyaris seluruh sasaran yang sesuai: cluster di cloud, mesin lokal, CPU atau GPU, Android dan perangkat iOS. Jika memakai cloud dari Google, untuk akselerasi lebih lanjut bisa menggunakan TensorFlow Processing Unit (TPU) pada Google. Karena, dapat diimplementasikan di hampir seluruh perangkat jika model TensorFlow berhasil dibuat.

TensorFlow 2.0, diumumkan pada September 2019, mengganti Acuan kerja dengan berbagai cara yang bersumber dari saran pengguna, untuk menjadikannya lebih mudah digunakan (contohnya, dengan memakai API Keras yang sederhana untuk train model) dan lebih berperforma. API baru akan mempermudah pengimpenetasian Pelatihan terdistribusi, dan di dukung menggunakan TensorFlow Lite memudahkan implementasi model pada beraneka platform. Namun, code pemrograman versi TensorFlow sebelumnya terkadang harus sedikit ditulis ulang, terkadang secara keseluruhan untuk memaksimalkan fitur TensorFlow 2.0.

2.4. Manfaat Tensorflow

Tensorflow merupakan kerangka komputasional untuk menciptakan model machine learning. TensorFlow menyajikan beraneka macam toolkit yang memudahkan untuk menciptakan model di tingkat abstraksi yang diinginkan [10]. Dengan menentukan serangkaian prosedur matematis pengguna bisa memanfaatkan API dengan tingkat yang lebih rendah untuk membuat model. Sebagai alternatif, pengguna dapat memakai API dengan tingkat yang lebih tinggi (seperti `tf.estimator`) untuk menerapkan arsitektur yang telah ditentukan, seperti regresi linier atau neural network. Berikut ini merupakan hierarki pada toolkit Tensorflow:



Gambar2.4 Hirarki Toolkit Tensorflow

Manfaat signifikan yang disediakan TensorFlow pada pengembangan machine learning yaitu abstraksi.

Dengan menggunakan Tensorflow, pengguna tidak perlu terlibat dengan kerumitan penerapan algoritme, atau mencari opsi yang tepat untuk merelasikan output antar fungsi ke input fungsi lainnya, pengguna hanya cukup fokus pada logika pemrograman secara keseluruhan. Karena TensorFlow yang akan menangani rincian di balik layar.

TensorFlow memberikan bonus kemudahan untuk developer yang hendak menjalankan debug dan melaksanakan pemeriksaan dalam aplikasi TensorFlow[10]. Mode eager execution memudahkan developer menguji dan mengedit setiap operasi grafik secara eksklusif dan ekspilist, tanpa menjadikan semua grafik menjadi satu objek buram dan sekaligus memeriksanya. Prosedur visualisasi TensorBoard memudahkan Developer mengecek dan menyajikan profil secara grafik yang dilaksanakan lewat dasbor interaktif berbasis web.

TensorFlow pun menghasilkan berbagai profit dari dukungan sebuah perusahaan komersial A-list di Google. Google tak hanya mempercepat laju pengembangan di latar belakang proyek, tetapi juga membuat banyak penawaran mengenai TensorFlow

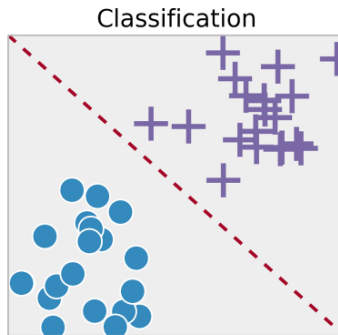
yang menjadikanya lebih mudah untuk diimplementasikan dan lebih mudah diaplikasikan.

Beberapa rincian penerapan TensorFlow memperrunyam hasil train model yang seluruhnya deterministik untuk sejumlah tugas pelatihan. Terkadang model yang ditrain di suatu sistem akan agak berbeda dari model yang ditrain di sistem lain, walaupun model tersebut diberikan data yang serupa.

BAB 3

PENGENALAN KLASIFIKASI

3.1. Pengertian Klasifikasi



Gambar3.1 Klasifikasi

Klasifikasi adalah teknik pengelompokan data berdasarkan karakteristik yang terdapat pada objek yang akan diklasifikasi [5]. Dalam prosesnya, klasifikasi bisa dilakukan dengan berbagai cara baik secara manual ataupun teknis. Klasifikasi yang dilaksanakan secara manual yaitu klasifikasi yang dilaksanakan oleh manusia tanpa bantuan teknis. Sedangkan klasifikasi yang dilaksanakan dengan teknologi, memiliki beberapa algoritma, diantaranya

Support Vector Machine, Naïve Bayes, Fuzzy, Decision Tree dan Jaringan Saraf Tiruan.

3.2. Tujuan Klasifikasi

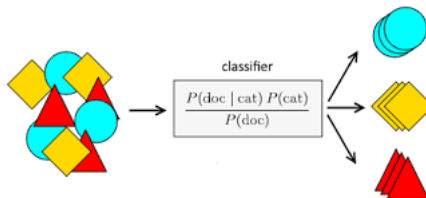
Tujuan klasifikasi adalah sebagai berikut:

- 1) Mengelompokkan data yang memiliki sifat-sifat yang sama kedalam kelompok-kelompok tertentu
- 2) Mempermudah membandingkan data
- 3) Mengelompokkan data berdasarkan variasi yang menonjol dan menghilangkan yang tidak perlu
- 4) Mempermudah analisis data
- 5) Mempermudah melakukan secara statistik terhadap data, misalnya: untuk analisa, interpretasi atau menyusun laporan.

3.3. Metode – Metode Klasifikasi

Berikut ini merupakan beberapa metode klasifikasi, antara lain:

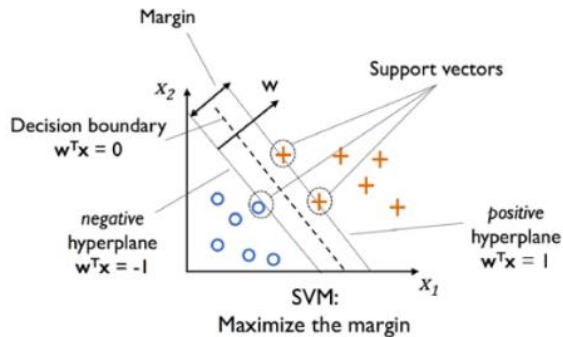
- 1) Naïve Bayes



Gambar 3.2 Klasifikasi Naive Bayes

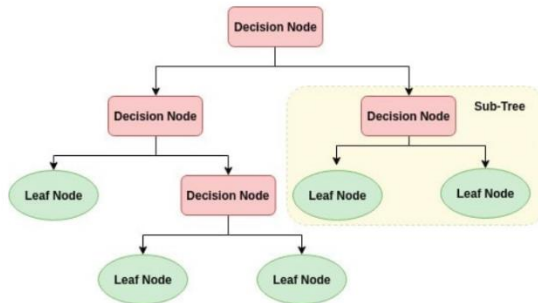
Naïve Bayes Classifier merupakan sebuah metode klasifikasi yang didasarkan pada teorema Bayes. Karakteristik utama dari metode Naïve Bayes ini yaitu dugaan yang sangat kuat (naïf) terhadap kebebasan dari masing-masing peristiwa / kejadian [6]. Naïve Bayes berdasarkan pada teorema bayes yang memiliki kapasitas klasifikasi yang sama dengan pohon keputusan dan jaringan saraf tiruan. Metode Naïve Bayes (NB) adalah salah satu Bayesian sederhana yang terdiri dari jaringan untuk klasifikasi. Dengan lebih dari satu set diskrit dan variabel stokastik yang telah di distribusi menggunakan probabilitas gabungan, jaringan Bayes dapat diarahkan sebagai tabel (Liao, 2007) . Karena beberapa alasan Metode ini di anggap penting, yaitu., Parameter estimasi skema berulang sangat mudah untuk dibangun. Ini berarti pada Data set yang besar dapat dengan mudah diimplementasikan . Sangat mudah untuk dipahami, sehingga pengguna tidak terampil dalam teknologi classifier ini dapat memahami metode ini dengan mudah.

2) Support Vector Machine



SVM atau Support Vector Machine adalah model linier untuk masalah klasifikasi dan regresi. Ini dapat memecahkan masalah linier dan non-linier dan bekerja dengan baik untuk banyak masalah praktis. Ide SVM sederhana: Algoritma membuat garis atau hyperplane yang memisahkan data ke dalam kelas. Pada pendekatan pertama yang dilakukan SVM adalah menemukan garis pemisah (atau hyperplane) antara data dua kelas. SVM adalah algoritma yang mengambil data sebagai input dan mengeluarkan garis yang memisahkan kelas-kelas tersebut jika memungkinkan.

3) Decission Tree

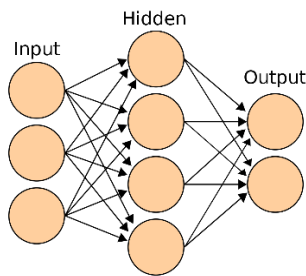


Gambar3.4 Decision Tree

Pohon keputusan adalah diagram atau bagan yang membantu menentukan tindakan atau menunjukkan probabilitas statistic[6]. Bagan ini disebut pohon keputusan karena kemiripannya dengan tanaman senama, biasanya digambarkan sebagai diagram tegak atau horizontal yang bercabang. Mulai dari keputusan itu sendiri (disebut "simpul"), setiap "cabang" dari pohon keputusan mewakili kemungkinan keputusan, hasil, atau reaksi. Cabang terjauh pada pohon mewakili hasil akhir dari jalur keputusan tertentu dan disebut "daun". Dengan menampilkan urutan langkah, pohon keputusan memberi orang cara yang efektif dan mudah untuk memvisualisasikan dan memahami efek potensial dari suatu

keputusan dan berbagai kemungkinan hasil. Pohon keputusan juga membantu orang mengidentifikasi setiap opsi potensial dan menimbang setiap tindakan terhadap risiko dan imbalan yang dapat dihasilkan oleh setiap opsi.

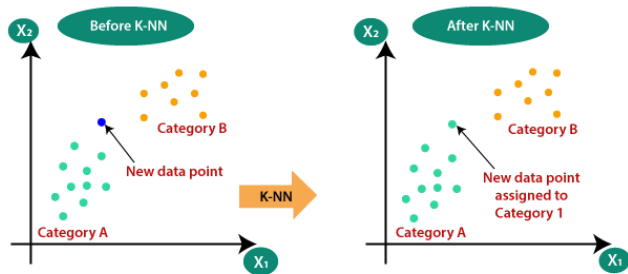
4) Jaringan Saraf Tiruan



Gambar3.5 Struktur Neural Network

Jaringan saraf adalah serangkaian algoritme yang berupaya mengenali hubungan mendasar dalam sekumpulan data melalui proses yang meniru cara otak manusia beroperasi. Dalam pengertian ini, jaringan saraf mengacu pada sistem neuron, baik organik atau buatan di alam. Jaringan saraf dapat beradaptasi dengan perubahan input; sehingga jaringan menghasilkan hasil terbaik tanpa perlu mendesain ulang kriteria keluaran.

5) K-Nearest Neighbor

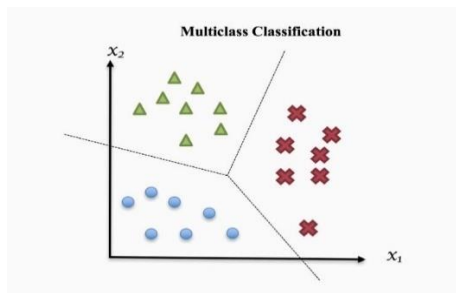


Gambar3.6 Before After klasifikasi KNN

Algoritma K-nearest neighbor (KNN) adalah jenis algoritma ML terawasi yang dapat digunakan baik untuk klasifikasi maupun masalah prediksi regresi [2]. Namun, ini terutama digunakan untuk masalah prediksi klasifikasi di industri. Dua properti berikut akan mendefinisikan KNN dengan baik. Algoritma Lazy learning KNN merupakan algoritma pembelajaran malas karena tidak memiliki fase pelatihan khusus dan menggunakan semua data untuk pelatihan saat klasifikasi. Algoritma pembelajaran non-parametrik KNN juga merupakan algoritma pembelajaran non-parametrik karena tidak mengasumsikan apa pun tentang data yang

mendasarinya... Metode Nearest Neighbor menetapkan nilai prediksi untuk pengamatan baru berdasarkan pluralitas atau mean (rata-rata berbobot) dari k "Nearest Neighbors" di set pelatihan.

3.4. Pengertian Klasifikasi Multiclass



Gambar3.7 Klasifikasi Multiclass

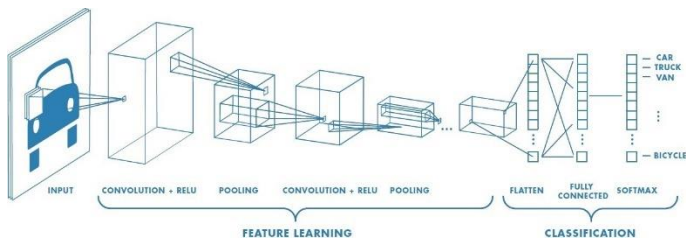
Klasifikasi Multiclass adalah tugas klasifikasi dengan lebih dari dua kelas. Dimana setiap sampel hanya dapat diberi label sebagai satu kelas. Misalnya, klasifikasi menggunakan fitur yang diekstraksi dari sekumpulan gambar sel darah, di mana setiap gambar dapat berupa Eosinofil, Limfosit, Monosit, dan Neutrofil. Setiap gambar adalah satu sampel dan diberi label sebagai salah satu dari 4 kemungkinan kelas. Klasifikasi multikelas membuat asumsi bahwa setiap

sampel ditugaskan ke satu dan hanya satu label - satu sampel tidak dapat lebih, misalnya, menjadi Limfosit dan Monosit.

BAB 4

PENGENALAN CONVOLUTIONAL NEURAL NETWORK (CNN)

4.1. Pengertian Convolutional Neural Network



Gambar 4.1 Cara Kerja CNN

Jaringan saraf convolutional, juga disebut ConvNets, pertama kali diperkenalkan pada 1980-an oleh Yann LeCun, seorang peneliti ilmu komputer pascadoktoral. LeCun telah dibangun di atas pekerjaan yang dilakukan oleh Kunihiro Fukushima, seorang ilmuwan Jepang yang, beberapa tahun sebelumnya, telah menemukan neocognitron, jaringan saraf pengenalan gambar yang sangat dasar.

Versi awal CNN, yang disebut LeNet (setelah LeCun), dapat mengenali angka tulisan tangan. CNN menemukan ceruk pasar di perbankan dan layanan pos dan perbankan, di mana mereka membaca kode pos pada amplop dan angka pada cek.

Namun terlepas dari kecerdikan mereka, ConvNets tetap berada di sela-sela visi komputer dan kecerdasan buatan karena mereka menghadapi masalah serius: Mereka tidak dapat menskalakan. CNN membutuhkan banyak data dan sumber daya komputasi untuk bekerja secara efisien untuk gambar besar. Pada saat itu, teknik ini hanya berlaku untuk gambar dengan resolusi rendah.

Convolutional Neural Network adalah algoritma deep learning yang dapat mengenali atau mendeteksi sebuah object dari inputan yang berupa gambar [11]. Pra-pemrosesan yang diperlukan dalam CNN jauh lebih rendah dibandingkan dengan algoritma klasifikasi lainnya. Sementara dalam metode primitif, filter direkayasa dengan tangan, dengan pelatihan yang cukup, CNN memiliki kemampuan untuk mempelajari filter/karakteristik ini. Arsitektur CNN analog dengan pola konektivitas Neuron di Otak Manusia dan

terinspirasi oleh organisasi Korteks Visual [3]. Neuron individu merespons rangsangan hanya di wilayah terbatas bidang visual yang dikenal sebagai Bidang Reseptif. Kumpulan bidang tersebut tumpang tindih untuk menutupi seluruh area visual.

Perbedaan ANN dan CNN adalah pada ANN, titik data konkret harus disediakan. Misalnya, dalam model di mana kita mencoba membedakan antara anjing dan kucing, lebar hidung dan panjang telinga harus diberikan secara eksplisit sebagai titik data. Saat menggunakan CNN, fitur spasial ini diekstraksi dari input gambar. Ini membuat CNN ideal ketika ribuan fitur perlu diekstraksi. Alih-alih harus mengukur setiap fitur individual, CNN mengumpulkan fitur-fitur ini sendiri..

1) Input layer



Gambar4.2 Perbedaan Gambar RGB

Input layer adalah awal dari alur kerja untuk jaringan saraf tiruan. [12]. Untuk gambar dengan ukuran 64×64 yang memiliki 3 channel warna, RGB (Red, Green, Blue) maka yang menjadi inputan adalah piksel array yang berukuran $64 \times 64 \times 3$.

2) Convolution Layer

Lapisan konvolusi menerapkan operasi konvolusi ke input, meneruskan hasilnya ke lapisan berikutnya. Konvolusi mengubah semua piksel dalam bidang reseptifnya menjadi satu nilai. Misalnya, jika Anda akan menerapkan konvolusi ke gambar, Anda akan mengurangi ukuran gambar serta menyatukan semua informasi di bidang menjadi satu piksel.

Hasil akhir dari convolutional layer adalah sebuah vektor. Berdasarkan jenis masalah yang perlu kita pecahkan dan jenis fitur yang ingin kita pelajari, kita dapat menggunakan berbagai jenis konvolusi.

3) Activation Function

Activation Functions

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Gambar4.3 Macam Activation Function

Activation Layer adalah layer dimana feature map dimasukkan ke dalam fungsi aktivasi [12]. Fungsi aktivasi memutuskan, apakah sebuah neuron harus diaktifkan atau tidak dengan menghitung jumlah bobot dan selanjutnya menambahkan bias dengannya. Tujuan dari fungsi aktivasi adalah untuk memasukkan non-linearitas ke dalam output neuron.

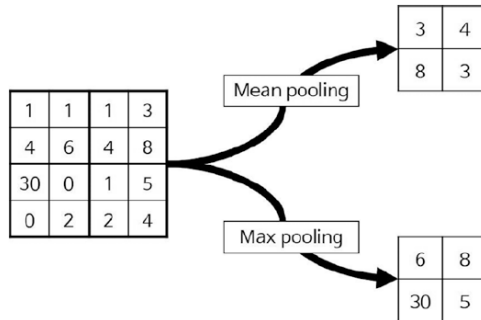
4) Pooling Layer

Pooling layer menerima input dari activation layer kemudia mengurangi jumlah paramaternya [11]. Pooling layer digunakan untuk mengurangi dimensi peta fitur. Dengan demikian, ini mengurangi jumlah parameter untuk dipelajari dan jumlah komputasi yang dilakukan dalam jaringan. Melakukan proses pooling pada feature map, sebagai contoh menggunakan feature map berukuran 4×4 berikut.

1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

Gambar4.4 Feature map 4×4

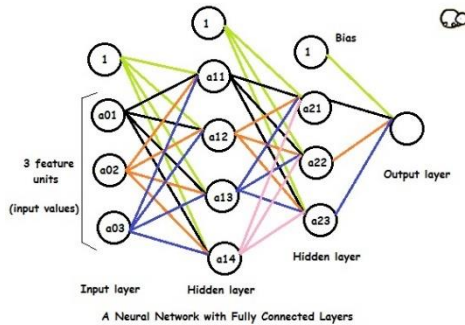
Proses pooling terbagi menjadi beberapa kategori yaitu Max pooling, Mean pooling, Sum pooling.



Gambar4.5 Macam Pooling

5) Fully Connected Layer

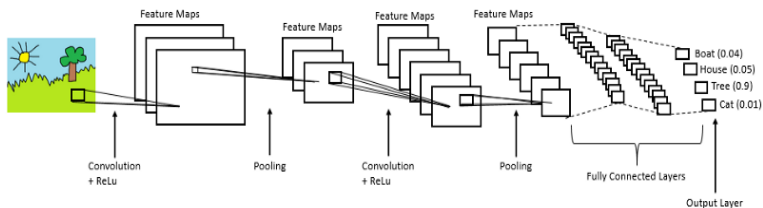
setelah melewati proses-proses diatas, hasil dari pooling layer digunakan menjadi masukan untuk Fully connected layer [12]. Fully Connected layer merupakan lapisan di mana semua input dari satu lapisan terhubung ke setiap unit aktivasi lapisan berikutnya. Dalam model pembelajaran mesin paling populer, beberapa lapisan terakhir adalah lapisan terhubung penuh yang mengkompilasi data yang diekstraksi oleh lapisan sebelumnya untuk membentuk hasil akhir. Ini adalah lapisan kedua yang paling memakan waktu setelah Convolution Layer.



Gambar 4.6 Fully Connected Layer

pada gambar diatas dapat dilihat Lapisan pertama/input memiliki 3 unit fitur dan ada 4 unit aktivasi di lapisan tersembunyi berikutnya. Angka 1 di setiap lapisan adalah unit bias .a01 , a02 dan a03 adalah nilai input ke jaringan saraf. Mereka pada dasarnya adalah fitur dari contoh pelatihan.

4 unit aktivasi lapisan tersembunyi pertama terhubung ke semua 3 unit aktivasi lapisan tersembunyi kedua Bobot/parameter menghubungkan kedua lapisan.



Gambar 4.7 Proses CNN

CNN berhasil menangkap dependensi Spasial dan Temporal dalam gambar melalui penerapan filter yang relevan. Arsitektur melakukan pemasangan yang lebih baik ke dataset gambar karena pengurangan jumlah parameter yang terlibat dan bobot yang dapat digunakan kembali. Dengan kata lain, jaringan dapat dilatih untuk memahami kecanggihan gambar dengan lebih baik. Peran CNN adalah untuk mereduksi gambar menjadi bentuk yang lebih mudah untuk diproses, tanpa kehilangan fitur yang sangat penting untuk mendapatkan prediksi yang baik.. Ini penting ketika kita mendesain arsitektur yang tidak hanya bagus dalam mempelajari fitur tetapi juga dapat diskalakan untuk kumpulan data yang sangat besar.

Penggabungan setiap neuron dengan menggunakan bobot pada iterasi tertentu yang telah diperbarui serta jumlah neuron dan jumlah ekstraksi

fitur yang dihasilkan oleh konvolusi sehingga CNN memiliki akurasi yang tinggi. Akurasi yang tinggi dihasilkan dari kombinasi terbaik. Berikut kombinasi yang sering dimodifikasi adalah [3]:

- 1) Membatasi ukuran convolution untuk mendapatkan jumlah layer. Iterasi yang semakin banyak akan menyebabkan proses menjadi lama. Akan tetapi, iterasi yang semakin sedikit juga menyebabkan pendekatan kebenaran terpengaruh akibat sedikitnya jumlah fitur.
- 2) Ukuran kernel berfungsi sebagai sub matriks.. Ukuran kernel yang semakin kecil, hasil yang dihasilkan akan semakin detail. Akan tetapi, hal tersebut menyebabkan waktu komputasi semakin lama.
- 3) Jumlah layer berfungsi sebagai penampung hasil konvolusi. layer yang semakin banyak, akan menghasilkan fitur yang semakin banyak juga. Akan tetapi, hal tersebut akan mengakibatkan waktu semakin lamanya komputasi.
- 4) Jumlah Fully-Connected berfungsi menggabungkan ekstraksi fitur ke dalam kelas. Prosedurnya adalah dengan memberikan nilai

perkalian acak dari bobot dan bias. Perkalian acak akan di perbarui dan diulang hingga mendapatkan bobot ketika nilai tersebut masih jauh dan pendekatan kelas dari bias yang bagus. Proses ini memerlukan waktu yang lumayan lama tergantung jumlah fitur yang dihasilkan.

- 5) Pooling layer adalah lapisan pengurangan fitur yang dihasilkan oleh ekstraksi fitur. Dimensi yang tidak digunakan akan dieliminasi Pooling. Parameter dari pooling umumnya menggunakan rata-rata atau nilai maksimum.

Berdasarkan pemaparan di atas, untuk mendapatkan nilai presisi dan akurasi tinggi serta waktu yang singkat. Hal itu dilakukan dengan cara memaksimalkan parameter CNN serta mencari kombinasi parameter yang paling tepat [3].

4.2. Tujuan Convolutional Neural Network

Tujuan dari operasi konvolusi adalah untuk mengekstrak fitur tingkat tinggi seperti tepi, dari gambar input. CNN tidak perlu dibatasi dengan Convolutional Layer. Secara umum, Layer pertama Convolutional bertanggung jawab untuk menangkap fitur tingkat rendah seperti tepi, warna, orientasi gradien, dll. Dengan lapisan tambahan, arsitektur juga menyesuaikan dengan fitur tingkat tinggi, memberi jaringan yang memiliki pemahaman yang baik.

BAB 5

PERSIAPAN PRAKTEK

5.1. Alat dan Bahan

Berikut ini beberapa hal yang perlu dipersiapkan :

- 1) Komputer / Laptop.
- 2) Jaringan Internet.
- 3) Google Account (Untuk mengakses Google colabs).
- 4) Pengetahuan dasar mengenai Bahasa pemrograman Python.
- 5) Dataset yang akan digunakan.

5.2. Petunjuk Penggunaan Google Colabs

Google Colab adalah salah satu produk Google berbasis cloud untuk pembelajaran mesin gratis[8]. Google Colab adalah Integrated Development Environment Python dengan format “ipynb/notebook” (mirip Jupyter notebook namun berbasis cloud).

Google Colab memiliki beberapa manfaat, antara lain:

- a) Yang terpenting adalah gratis.

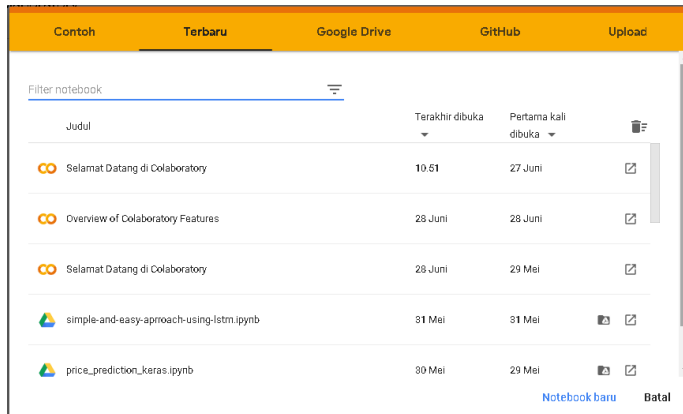
- b) Agar sesuai dengan standar industri atau jika Anda menjalankan kode yang memakan waktu dan memori, tidak mungkin memasang GPU yang mahal pada mesin pribadi Anda. Sepengetahuan saya, setiap notebook di lingkungan Colab dapat menggunakan GPU gratis (yaitu NVIDIA Tesla K80 ~ berharga £3500.00) di lingkungan Colab, terlepas dari mesin pribadi apa yang Anda gunakan. Selain itu, jika Anda tidak dapat menyelesaikan program atau tugas, Anda dapat menyelesaikannya dari tempat kerja/workstation hanya dengan masuk ke akun Colab.
- c) Colab menyediakan sistem kontrol versi bawaan menggunakan Git dan cukup mudah untuk membuat catatan dan dokumentasi, menyertakan gambar, dan tabel menggunakan penurunan harga.
- d) Seperti namanya, Google Colab hadir dengan kolaborasi yang didukung dalam produk. Faktanya, ini adalah Notebook Jupyter yang memanfaatkan fitur kolaborasi Google Documents.
- e) Itu juga berjalan di server Google menggunakan mesin virtual dan Anda tidak perlu menginstal paket apa pun, yang terkadang membuat kesulitan

Anda menggunakan sistem operasi yang berbeda seperti MAC, Windows, dan Linux.

- f) Saat ini di industri, persyaratan paling umum dalam paket pekerjaan adalah pengetahuan tentang pemrograman model pembelajaran mendalam menggunakan GPU.
- g) Tautkan ke profil GitHub Anda. Tautan GitHub berguna untuk perekrut. Anda dapat dengan mudah memamerkan pekerjaan Anda dari mana saja di seluruh dunia dan perekrut dapat menjelajahi pekerjaan yang Anda lakukan, yang membantu mereka mengidentifikasi kandidat kuat untuk pekerjaan tersebut.

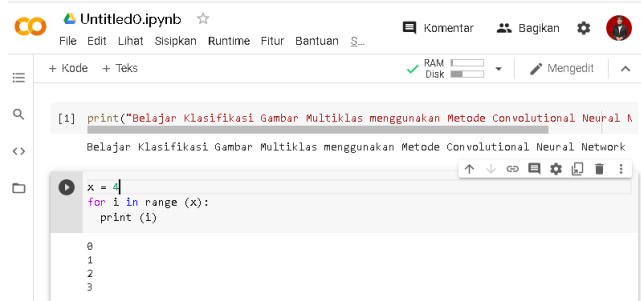
Setelah mengetahui beberapa manfaat dari Google colabs, berikut merupakan Langkah – Langkah untuk menggunakan Google Colabs:

- 1) Login akun google lalu menuju halaman resmi Google Colabs melalui tautan berikut ini <https://colab.research.google.com/>. Setelah itu akan tampil tampilan seperti berikut:



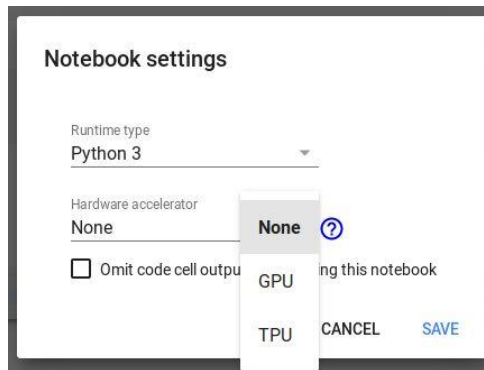
Gambar 5.1 Tampilan Google Colab

- 2) Untuk membuat notebook baru, klik pada New Python 3 Notebook (atau Python 2 tergantung pada yang ingin digunakan) . Setelah itu akan muncul tampilan mirip Jupyter Notebook. Seperti dibawah ini



Gambar 5.2 Tampilan Notebook

- 3) Untuk menjalankan program Python menggunakan GPU atau TPU, cukup klik menu 'Edit' lalu 'Notebook Settings', lalu klik bagian 'Hardware Accelerator' pilih GPU/TPU sesuai yang diinginkan.



Gambar5.3 Tampilan Notebook Setting

- 4) Untuk menghubungkan dengan Google colabs kita bisa menjalankan program dibawah ini. Karena Google Colab akan mereset maksimal 12 jam sekali pada notebook beserta semua temporary filenya.

```
1. from google.colab import drive
2. drive.mount('/content/drive')
```

Gambar 5.4 Perintah untuk menghubungkan Google Drive

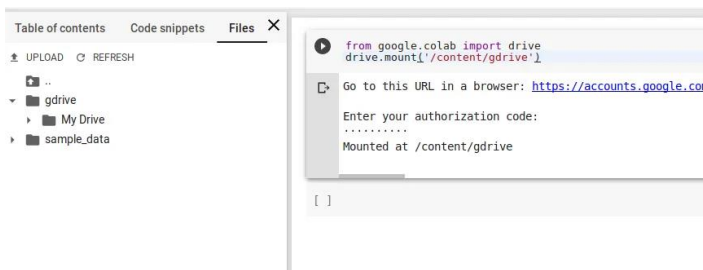
Jika perintah di atas dijalankan, maka akan diberikan alamat URL yang menuju ke halaman

permohonan akses Google Drive. Jika sudah memberi izin, akan diberikan kode yang dapat dituliskan di kolom seperti gambar dibawah.



Gambar5.5 Tampilan Output Autentikasi ke Google drive

Setelah berhasil tersambung, maka akan tampil daftar file di bagian kiri Notebook seperti dibawah ini. Untuk mengakses file tersebut, cukup arahkan proses load / save ke *path drive/My Drive/FOLDERTUJUAN*



Gambar5.6 Tampilan jika perintah berhasil

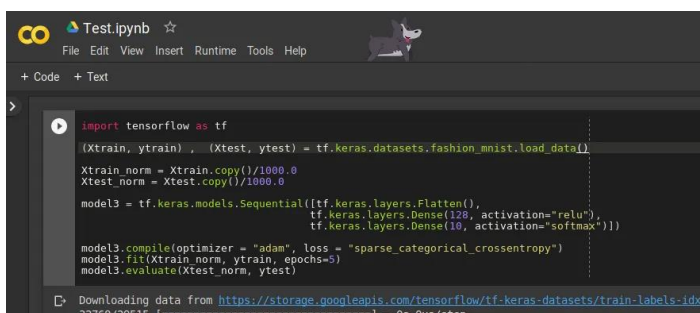
- 5) Untuk Mengupload File ke Google Colab, tersedia tempat penyimpanan file sementara pada colab. Berikut ini adalah perintah untuk mengupload file ke colab.

1.	<code>from google.colab import files</code>
2.	<code>upload = files.upload()</code>

Gambar5.7 Perintah Upload file

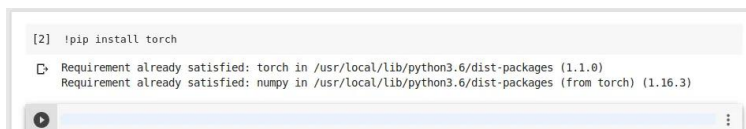
Jika perintah di atas di jalankan akan tampil kotak dialog untuk upload file. Perintah di atas cukup mudah untuk mengupload file kecil (untuk dataset besar, lebih baik diletakkan di drive)

- 6) Untuk pengaturan tema dapat di lakukan dengan klik menu ‘tools’ lalu ‘preferences’ lalu ‘site’



Gambar5.8 Tampilan tema yang berbeda

- 7) Untuk install Python package di Colab bisa menggunakan instalasi **pip**.



Gambar5.9 contoh perintah pip

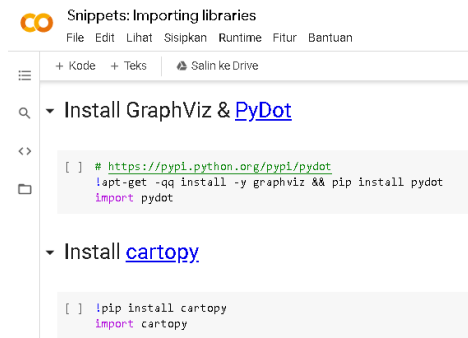
Perbedaan install pip pada umumnya dengan di google colab hanya terdapat pada tanda seru “!” di pip google colab

5.3. Instalasi Library

Untuk instalasi Library yang tidak tersedia pada Google Colab, dapat menggunakan perintah berikut.

!pip install atau *!apt-get install* .

Contohnya sebagai berikut :

A screenshot of the Google Colab 'Snippets: Importing libraries' interface. The window has a title bar with 'File', 'Edit', 'Lihat', 'Sisipkan', 'Runtime', 'Fitur', and 'Bantuan'. Below the title bar is a toolbar with '+ Kode', '+ Teks', and 'Salin ke Drive'. The main content area shows two expandable snippets. The first snippet is titled 'Install GraphViz & PyDot' and contains the following code:

```
[ ] # https://pypi.python.org/pypi/pydot
!apt-get -qq install -y graphviz && pip install pydot
import pydot
```

 The second snippet is titled 'Install cartopy' and contains the following code:

```
[ ] !pip install cartopy
import cartopy
```

Gambar5.10 contoh perintah install library

BAB 6

KLASIFIKASI GAMBAR MULTICLASS MENGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)

6.1. Langkah – Langkah klasifikasi gambar Multiclass menggunakan metode Convolutional Neural Network (CNN)

- 1) Memilih dataset yang akan digunakan, di buku tutorial ini menggunakan dataset dari Kagle.com . Dataset yang digunakan berjudul Blood cell images. Dataset ini berisi 12.500 gambar augmented sel darah (JPEG) dengan label tipe sel (CSV) yang menyertainya. Ada sekitar 3.000 gambar untuk masing-masing dari 4 jenis sel berbeda yang dikelompokkan ke dalam 4 folder berbeda (sesuai dengan jenis sel). Jenis selnya adalah Eosinofil, Limfosit, Monosit, dan Neutrofil. Dataset ini disertai dengan kumpulan data tambahan yang berisi 410 gambar asli (pra-augmentasi) serta dua label subtype tambahan (WBC vs WBC) dan juga kotak pembatas untuk setiap sel di masing-masing 410 gambar ini

(metadata JPEG + XML). Lebih khusus lagi, folder 'dataset-master' berisi 410 gambar sel darah dengan label subtype dan kotak pembatas (JPEG + XML), sedangkan folder 'dataset2-master' berisi 2.500 gambar tambahan serta 4 label subtype tambahan (JPEG + CSV).

- 2) Import Library dan package yang akan digunakan, pada tutorial ini menggunakan library sebagai berikut.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
import seaborn as sns
from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn import decomposition
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import tensorflow as tf
import keras
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential, Model
from keras.applications.densenet import DenseNet201
from keras.initializers import he_normal
from keras.layers import Lambda, SeparableConv2D, BatchNormalization, Dropout,
| MaxPooling2D, Input, Dense, Conv2D, Activation, Flatten
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
```

Gambar 6.1 Import Library dan Package

- 3) Membuat class_names yang berisi tipe sel darah, serta menentukan ukuran gambar. Pada tutorial ini menggunakan ukuran gambar 150 x 150

```
[2] class_names = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']
nb_classes = len(class_names)
image_size = (150,150)
```

Gambar 6.2 Class name

- 4) Mempersiapkan dataset untuk memuat data train dan test menggunakan fungsi `load_data`.

```
def load_data():  
    datasets = ['/content/drive/My Drive/colab/images/TRAIN', '/content/drive/My Drive/colab/images/TEST']  
    images = []  
    labels = []  
  
    # Perulangan training and test sets  
    for dataset in datasets:  
        # perulangan melalui folders dataset  
        for folder in os.listdir(dataset):  
            if folder in ['EOSINOPHIL']: label = 0  
            elif folder in ['LYMPHOCYTE']: label = 1  
            elif folder in ['MONOCYTE']: label = 2  
            elif folder in ['NEUTROPHIL']: label = 3  
  
            # perulangan melalui setiap gambar di folder  
            for file in tqdm(os.listdir(os.path.join(dataset, folder))):  
                # mendapatkan nama path dataset gambar  
                img_path = os.path.join(os.path.join(dataset, folder), file)
```

Gambar 6.3 Load Data 1

```
# membuka dan mengubah ukuran gambar  
image = cv2.imread(img_path)  
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  
image = cv2.resize(image, image_size)  
  
# menambahkan gambar dan label yang sesuai dengan output  
images.append(image)  
labels.append(label)  
  
images = np.array(images, dtype = 'float32')  
labels = np.array(labels, dtype = 'int32')  
  
return images, labels  
  
[5] images, labels = load_data()  
  
100%|██████████| 2478/2478 [10:44<00:00, 3.85it/s]  
100%|██████████| 2499/2499 [10:17<00:00, 4.05it/s]  
100%|██████████| 2497/2497 [10:12<00:00, 4.07it/s]  
100%|██████████| 2483/2483 [10:07<00:00, 4.09it/s]  
100%|██████████| 620/620 [02:29<00:00, 4.14it/s]  
100%|██████████| 624/624 [02:36<00:00, 3.98it/s]  
100%|██████████| 623/623 [02:30<00:00, 4.15it/s]  
100%|██████████| 620/620 [02:35<00:00, 3.97it/s]
```

Gambar 6.4 Load Data 2

- 5) Semua data (Pelatihan dan Tes gabungan) diacak dan dibagi menjadi Pelatihan (80%), Validasi (10%), Tes (10%) set.

```
[6] images, labels = shuffle(images, labels, random_state=10)

train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size = 0.2)
test_images, val_images, test_labels, val_labels = train_test_split(test_images, test_labels, test_size = 0.5)
```

Gambar 6.5 pengacakan Train dan test data

- 6) Eksplorasi data melalui pemeriksaan performa, dengan mengetahui Berapa banyak contoh train, validasi, dan test yang kita miliki? Apa bentuk gambar dan labelnya? berapa proporsi masing – masing kategori? apakah dataset seimbang?

```
[7] n_train = train_labels.shape[0]
    n_val = val_labels.shape[0]
    n_test = test_labels.shape[0]

    print("Number of training examples: {}".format(n_train))
    print("Number of validation examples: {}".format(n_val))
    print("Number of testing examples: {}".format(n_test))

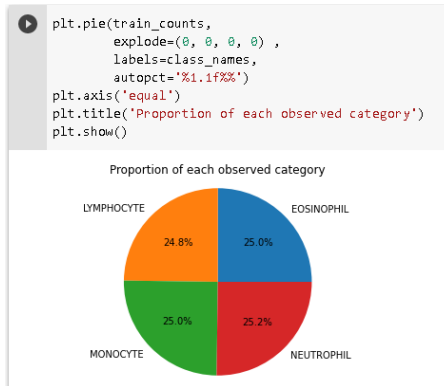
    print("Training images are of shape: {}".format(train_images.shape))
    print("Training labels are of shape: {}".format(train_labels.shape))
    print("Validation images are of shape: {}".format(val_images.shape))
    print("Validation labels are of shape: {}".format(val_labels.shape))
    print("Test images are of shape: {}".format(test_images.shape))
    print("Test labels are of shape: {}".format(test_labels.shape))

    Number of training examples: 9955
    Number of validation examples: 1245
    Number of testing examples: 1244
    Training images are of shape: (9955, 150, 150, 3)
    Training labels are of shape: (9955,)
    Validation images are of shape: (1245, 150, 150, 3)
    Validation labels are of shape: (1245,)
    Test images are of shape: (1244, 150, 150, 3)
    Test labels are of shape: (1244,)
```

Gambar6.6 Eksplorasi Data 1



Gambar6.7 Eksplorasi Data 2



Gambar6.8 Eksplorasi Data 3

- 7) Catatan singkat tentang mempersiapkan data: Nilai piksel dalam gambar RGB diwakili oleh bilangan bulat antara 0.255. Nilai bobot dalam CNN biasanya kecil dan jumlah yang besar dapat memperlambat atau mengganggu proses pembelajaran. Data dinormalisasi sehingga nilai piksel antara 0,1 untuk meningkatkan kecepatan komputasi.

```
[10] train_images = train_images / 255.0
     val_images = val_images / 255.0
     test_images = test_images / 255.0
```

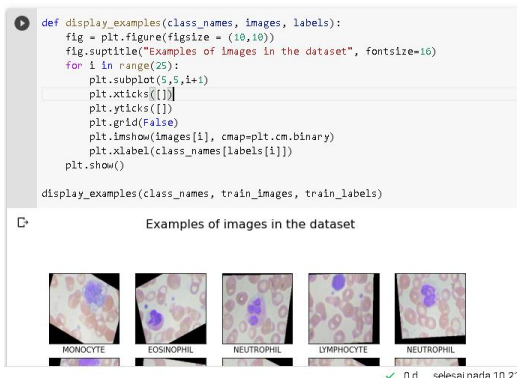
Gambar6.9 Normalisasi Piksel

- 8) Memeriksa perfoma lainnya dengan cara memvisualisasikan data dengan menampilkan satu

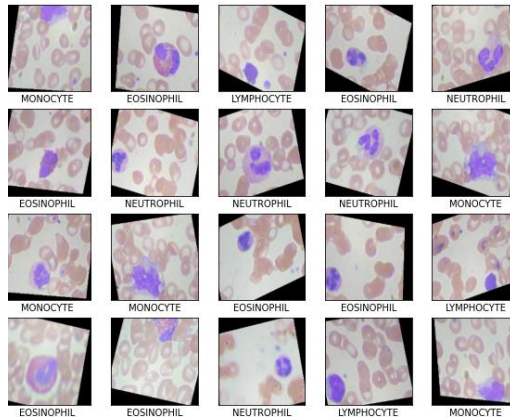
gambar acak dan 25 gambar acak dari set pelatihan dengan label yang sesuai.



Gambar6.10 Menampilkan 1 gambar acak



Gambar6.11 Menampilkan 25 gambar acak 1



Gambar6.12 .Menampilkan 25 gambar acak 2

- 9) Membuat model khusus Convolutional Neural Network (CNN). Model ini berisi urutan lima blok Conv yang berisi kombinasi lapisan SeparableConv2D, BatchNormalization, MaxPooling dan Dropout. Output dari blok Conv akhir diratakan dan diikuti oleh tiga lapisan Fully Connected (FC) masing-masing dengan lapisan Dropout sendiri. Lapisan FC akhir ditambahkan dengan empat unit dan aktivasi softmax untuk klasifikasi multikelas.

```

1 model = Sequential()

# First Conv block
model.add(Conv2D(16, (3,3), padding = 'same', activation = 'relu', input_shape = (150,150,3)))
model.add(MaxPooling2D(pool_size = (2,2)))

# Second Conv block
model.add(SeparableConv2D(32, (3,3), activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(32, (3,3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))

# Third Conv block
model.add(SeparableConv2D(64, (3,3), activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(64, (3,3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))

# Forth Conv block
model.add(SeparableConv2D(128, (3,3), activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(128, (3,3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.2))

```

✓ 1 | 23 m 45 d selesai pada 11.51

Gambar6.13 Membuat Model block 1-4

```

1 # Fifth Conv block
model.add(SeparableConv2D(256, (3,3), activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(256, (3,3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.2))

# FC layer
model.add(Flatten())
model.add(Dense(units = 512, activation = 'relu'))
model.add(Dropout(0.7))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.3))

# Output layer
model.add(Dense(units = 4, activation = 'softmax'))

# Compile
model.compile(optimizer = "adam", loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
model.summary()

```

Gambar6.14 Memmbuat Model block 5, FC layer,Ouput layer dan complie

```

# Implement callbacks
checkpoint = ModelCheckpoint(filepath='best_model.hdf5', save_best_only=True, save_weights_only=False)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=5, verbose=1, mode='min', restore_best_weights=True)
learning_rate_reduction = ReduceLROnPlateau(
    monitor = 'val_accuracy',
    patience = 2,
    verbose = 1,
    factor = 0.5,
    min_lr = 0.000001)

# Train
history = model.fit(
    train_images,
    train_labels,
    batch_size = 32,
    epochs = 10,
    validation_data=(val_images, val_labels),
    callbacks=[learning_rate_reduction])

Model: "sequential"

```

layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 16)	448

✓ 1 | 23 m 45 d selesai pada 11.51

Gambar6.15 Membuat model implement Callbacks dan Train

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 16)	448
conv2d_1 (Conv2D)	(None, 150, 150, 16)	2320
max_pooling2d (MaxPooling2D)	(None, 75, 75, 16)	0
separable_conv2d (SeparableC	(None, 75, 75, 32)	688
separable_conv2d_1 (Separabl	(None, 75, 75, 32)	1344
batch_normalization (BatchNo	(None, 75, 75, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 37, 37, 32)	0
separable_conv2d_2 (Separabl	(None, 37, 37, 64)	2400
separable_conv2d_3 (Separabl	(None, 37, 37, 64)	4736
batch_normalization_1 (Batch	(None, 37, 37, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 18, 18, 64)	0
separable_conv2d_4 (Separabl	(None, 18, 18, 128)	8896
separable_conv2d_5 (Separabl	(None, 18, 18, 128)	17664
		✓ 1129m45s

Gambar6.16 Output Layer Model 1

separable_conv2d_5 (Separabl	(None, 18, 18, 128)	17664
batch_normalization_2 (Batch	(None, 18, 18, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 9, 9, 128)	0
dropout (Dropout)	(None, 9, 9, 128)	0
separable_conv2d_6 (Separabl	(None, 9, 9, 256)	34176
separable_conv2d_7 (Separabl	(None, 9, 9, 256)	68896
batch_normalization_3 (Batch	(None, 9, 9, 256)	1024
max_pooling2d_4 (MaxPooling2	(None, 4, 4, 256)	0
dropout_1 (Dropout)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 512)	2097664
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65664
dropout_3 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256

Gambar6.17 output layer model 2

dropout_4 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 4)	260
=====		
Total params: 2,314,532		
Trainable params: 2,313,572		
Non-trainable params: 960		
=====		
Epoch 1/10		
312/312 [=====]	- 545s 2s/step - loss: 1.3253 - accuracy: 0.3141 - val_loss: 1.5619 - val_accuracy: 0.2426	
Epoch 2/10		
312/312 [=====]	- 497s 2s/step - loss: 0.7520 - accuracy: 0.6487 - val_loss: 1.5443 - val_accuracy: 0.2578	
Epoch 3/10		
312/312 [=====]	- 495s 2s/step - loss: 0.5742 - accuracy: 0.6899 - val_loss: 0.6939 - val_accuracy: 0.6578	
Epoch 4/10		
312/312 [=====]	- 493s 2s/step - loss: 0.4793 - accuracy: 0.7536 - val_loss: 36.6550 - val_accuracy: 0.3470	
Epoch 5/10		
312/312 [=====]	- 494s 2s/step - loss: 0.4106 - accuracy: 0.8287 - val_loss: 46.7933 - val_accuracy: 0.2458	

Gambar 6.18 Epoch 1-5

```
Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 6/10
312/312 [=====] - 499s 2s/step - loss: 0.3041 - accuracy: 0.8710 - val_loss: 0.1097 - val_accuracy: 0.9285
Epoch 7/10
312/312 [=====] - 496s 2s/step - loss: 0.1806 - accuracy: 0.9311 - val_loss: 2.8011 - val_accuracy: 0.6185
Epoch 8/10
312/312 [=====] - 500s 2s/step - loss: 0.1780 - accuracy: 0.9332 - val_loss: 0.4020 - val_accuracy: 0.6313
Epoch 00008: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 9/10
312/312 [=====] - 495s 2s/step - loss: 0.1170 - accuracy: 0.9530 - val_loss: 0.3640 - val_accuracy: 0.8988
Epoch 10/10
312/312 [=====] - 495s 2s/step - loss: 0.0953 - accuracy: 0.9612 - val_loss: 0.0859 - val_accuracy: 0.9622
```

Gambar 6.19 Epoch 6-10

10) Mengevaluasi Performa, berdasarkan model training, di tutorial ini saya menggunakan fungsi pembantu untuk memplot nilai Accuracy dan Loss untuk training dan validation set.

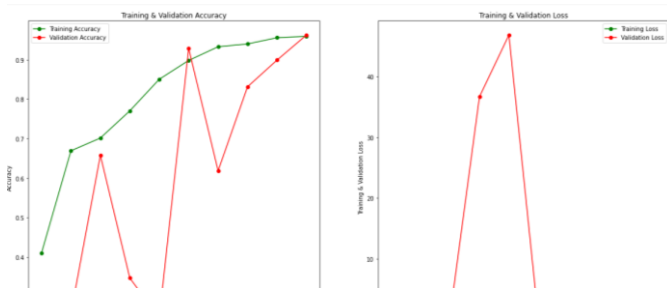
```
def plot_accuracy_loss_chart(history):
    epochs = [i for i in range(10)]
    fig, ax = plt.subplots(1, 2)
    train_acc = history.history['accuracy']
    train_loss = history.history['loss']
    val_acc = history.history['val_accuracy']
    val_loss = history.history['val_loss']
    fig.set_size_inches(20, 10)
    ax[0].plot(epochs, train_acc, 'go-', label = 'Training Accuracy')
    ax[0].plot(epochs, val_acc, 'r-o', label = 'Validation Accuracy')
    ax[0].set_title('Training & Validation Accuracy')
    ax[0].legend()
    ax[0].set_xlabel("Epochs")
    ax[0].set_ylabel("Accuracy")

    ax[1].plot(epochs, train_loss, 'g-o', label = 'Training Loss')
    ax[1].plot(epochs, val_loss, 'r-o', label = 'Validation Loss')
    ax[1].set_title('Training & Validation Loss')
    ax[1].legend()
    ax[1].set_xlabel("Epochs")
    ax[1].set_ylabel("Training & Validation Loss")
    plt.show()
```

Gambar 6.20 Membuat fungsi Plot accuracy dan loss

```
plot_accuracy_loss_chart(history1)
```

Gambar6.21 memanggil fungsi plot accuracy dan loss



Gambar6.22 plot accuracy dan loss

11) Mengevaluasi model pada test data untuk mengetahui loss dan accuracy.

```
results = model1.evaluate(test_images, test_labels)

print("Loss of the model is - ", results[0])
print("Accuracy of the model is - ", results[1]*100, "%")
```

39/39 [=====] - 15s 379ms/step - loss: 0.0961 - accuracy: 0.9542
 Loss of the model is - 0.0960891091853714
 Accuracy of the model is - 95.41808618171692 %

Gambar6.23 Evaluasi model

12) Menyimpan model

```
model1.save('Bloodcell_Classification_Model1_Custom_Build_10_epochs.h5')
```

Gambar6.24 menyimpan model

13) Membuat classification report dengan Sklearn

```
[20] from sklearn.metrics import classification_report

predictions = model1.predict(test_images)
predictions = np.argmax(predictions,axis=1)
predictions[:15]

array([2, 3, 2, 1, 2, 0, 0, 3, 1, 3, 3, 1, 3, 2, 3])
```

Gambar6.25 membuat classification report

```
print(classification_report(
    test_labels,
    predictions,
    target_names = ['EOSINOPHIL (Class 0)', 'LYMPHOCYTE (Class 1)', 'MONOCYTE (Class 2)', 'NEUTROPHIL (Class 3)']))
```

	precision	recall	f1-score	support
EOSINOPHIL (Class 0)	0.96	0.97	0.91	326
LYMPHOCYTE (Class 1)	1.00	0.98	0.99	314
MONOCYTE (Class 2)	1.00	1.00	1.00	292
NEUTROPHIL (Class 3)	0.87	0.97	0.92	312
accuracy			0.95	1244
macro avg	0.96	0.96	0.96	1244
weighted avg	0.96	0.95	0.95	1244

Gambar6.26 Output classification report

14) Memplot confusion matrix untuk mendapat hasil visualisasi yang lebih baik

```
[22] cm = confusion_matrix(test_labels, predictions)
cm = pd.DataFrame(cm, index = ['0', '1', '2', '3'], columns = ['0', '1', '2', '3'])
cm
```

	0	1	2	3
0	285	0	0	41
1	2	309	1	2
2	0	0	291	1
3	10	0	0	302

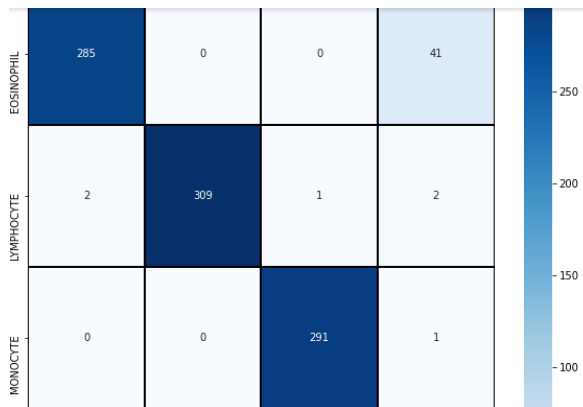
Gambar6.27 Membuat Confussion matrix

15) Membuat plot confusion matrix

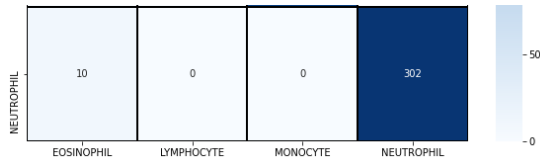
```
def plot_confusion_matrix (cm):
    plt.figure(figsize = (10,10))
    sns.heatmap(
        cm,
        cmap = 'Blues',
        linecolor = 'black',
        linewidth = 1,
        annot = True,
        fmt = '',
        xticklabels = class_names,
        yticklabels = class_names)

plot_confusion_matrix(cm)
```

Gambar6.28 Membuat plot confusion matrix



Gambar6.29 plot confusion matrix 1



Gambar6.30 plot confusion matrix 2

6.2. Hasil klasifikasi gambar Multiclass menggunakan metode (CNN)

	precision	recall	f1-score	support
EOSINOPHIL (Class 0)	0.96	0.87	0.91	326
LYMPHOCYTE (Class 1)	1.00	0.98	0.99	314
MONOCYTE (Class 2)	1.00	1.00	1.00	292
NEUTROPHIL (Class 3)	0.87	0.97	0.92	312
accuracy			0.95	1244
macro avg	0.96	0.96	0.96	1244
weighted avg	0.96	0.95	0.95	1244

Gambar6.31 Hasil klasifikasi gambar dengan CNN

Dari plot diatas dapat dilihat bahwa klasifikasi Limfosit dan Monosit mendekati sempurna bahkan sempurna dengan skor F1 0,99 dan 1,00. Model ini juga berkinerja baik dalam mengklasifikasikan Neutrofil dan Eosinofil, namun keduanya terkadang membingungkan terutama saat mengidentifikasi Eosinofil. Ini tercermin dalam skor F1 mereka masing-masing 0,92 dan 0,91.

Untuk melihat script lebih lanjut dapat Memindai Qrcode disamping ->



BAB 7

PENUTUP

7.1. Kesimpulan

Deep Learning telah menjadi metode yang dominan dalam berbagai tugas kompleks seperti klasifikasi gambar dan deteksi objek, kelas penting lain dari Neural network yang digunakan untuk mempelajari representasi gambar yang dapat diterapkan pada berbagai masalah penglihatan komputer . CNN dalam, khususnya, terdiri dari beberapa lapisan operasi linier dan non-linier yang dipelajari secara bersamaan, secara end-to-end. Untuk menyelesaikan tugas tertentu, parameter lapisan ini dipelajari selama beberapa iterasi.

Untuk mendapatkan performa model yang baik bisa dengan mengkombinasikan parameter hingga mendapatkan akurasi yang baik.

7.2. Saran

Buku tutorial ini, masih jauh dari kata baik dan sempurna. Masih banyak kekurangan dari segi tata bahasa maupun penyusunan. Selain itu, pembahasan-

pembahasan yang ada juga kurang mendalam. Berangkat dari hal itu, penulis berharap besar akan saran dan kritik dari pembaca untuk menjadi bahan evaluasi dimasa yang akan datang. Dan untuk penulis selanjutnya, diharap untuk lebih mengulas pembahasan maupun implementasi klasifikasi gambar multiklas menggunakan metode convolutional neural network secara lebih mendalam.

DAFTAR PUSTAKA

- [1] P. Mooney, "Blood Cell Images," 22 April 2018. [Online]. Available: <https://www.kaggle.com/paultimothymooney/blood-cells>.
- [2] I. A. SABILLA, "05111850010020-Master_Thesis," 15 Januari 2020. [Online]. Available: https://repository.its.ac.id/73567/1/05111850010020-Master_Thesis.pdf.
- [3] P. d. T. R. M. Astari Retnowardhani, "Apakah Deep Learning," 26 November 2019. [Online]. Available: <https://mmsi.binus.ac.id/2019/11/26/apakah-deep-learning/>.
- [4] M. G. A. P. M. F. A. F. A. D. Aji Prasetya Wibawa, "Metode-metode Klasifikasi," in *Prosiding Seminar Ilmu Komputer dan Teknologi Informasi*, Malang, 2018.
- [5] S. Dewi, "KOMPARASI 5 METODE ALGORITMA KLASIFIKASI DATA MINING," *Jurnal Techno Nusa Mandiri Vol. XIII*, pp. 61-62, 2016.
- [6] DQLAB, "Pengenalannya Konsep Algoritma Deep Learning," 22 Maret 2021. [Online]. Available:

<https://www.dqlab.id/pengenalan-konsep-algoritma-deep-learning>.

- [7] R. ADAM, "Mengal Google Colab," 7 Mei 2019. [Online]. Available: <https://structilmy.com/2019/05/mengenal-google-colab/>.
- [8] Google, "welcome to colaboratory," [Online]. Available: <https://colab.research.google.com/notebooks/welcome.ipynb?hl=id>.
- [9] Tensorflow, "Tensorflow," [Online]. Available: <https://www.tensorflow.org/>.
- [10] Hemera Academy, "TensorFlow – Machine Learning Framework buatan Google," 18 Januari 2021. [Online]. Available: <https://itlearningcenter.id/tensorflow-adalah/>.
- [11] Q. Lina, "Apa itu Convolutional Neural Network?," 2 Januari 2019. [Online]. Available: <https://medium.com/@16611110/apa-itu-convolutional-neural-network-836f70b193a4>. [Accessed 20 Juni 2021].
- [12] S. Tandungan, "Pengenalan Convolutional Neural Network – Part 1," 10 Ferbuari 2019. [Online]. Available: <http://sofyantandungan.com/pengenalan->

convolutional-neural-network-part-1/. [Accessed 26 Juni 2021].