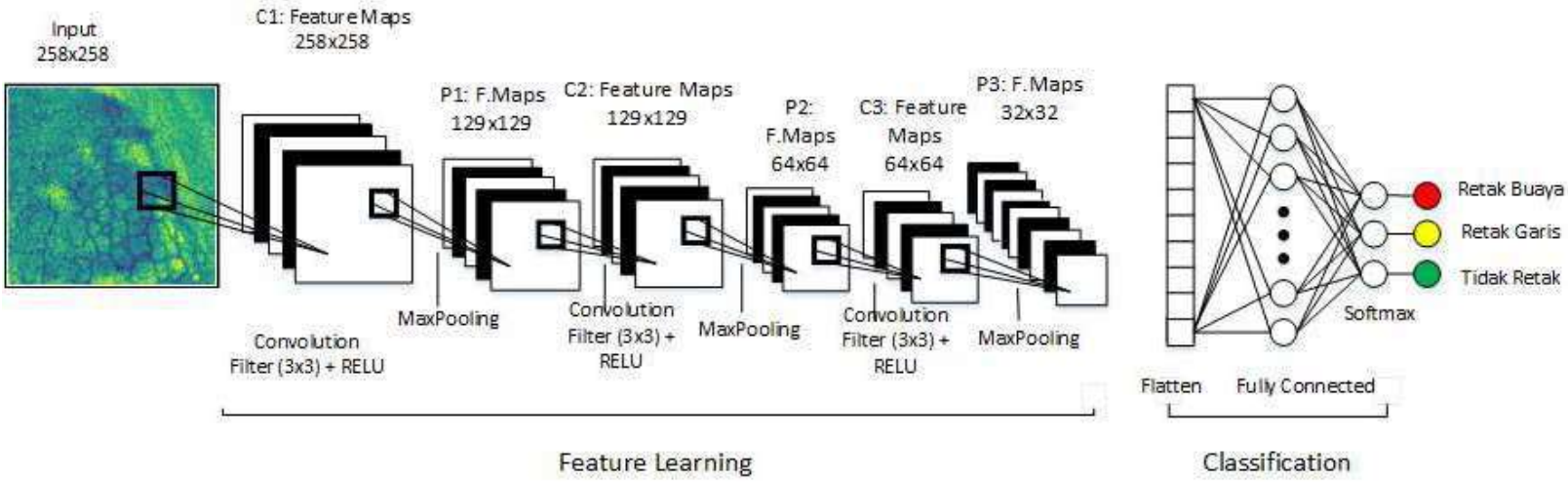
LAMPIRAN

LAMPIRAN A Arsitektur CNN



Lampiran A.1 Gambar Arsitektur CNN

Lampiran A. 2 Tampilan GUI

Lampiran B *Listing Code*

```
import os
import pickle
from tqdm import tqdm

import tkinter as tk
from tkinter.constants import ANCHOR, BOTTOM, CENTER, INSERT, LEFT,
RIGHT, S, TOP, X
from tkinter import ttk, filedialog

import numpy as np
import pandas as pd
from pandastable import Table

import cv2
import pywt
from PIL import Image, ImageTk

from tensorflow import keras

# ==== Constants ==== #
BACKGROUND = "#4bacc6"
ALTBACKGROUND = "#4bacb0"
shape = (512,512)
le = pickle.load(open(os.path.join("model","le.sav"),"rb"))
model = keras.models.load_model('model')
# ==== Constants ==== #

# ==== GUI ==== #
class root(tk.Tk):
    def __init__(self):
        # head
        super(root, self).__init__()
        self.title("Deteksi Keretakan Jalan")
        self.geometry("1024x768")
        self['background'] = BACKGROUND
        # top frame
        self.header = tk.Frame(self, background=BACKGROUND)
        self.header.pack(side=TOP)
        self.headerL = tk.Frame(self.header, background=BACKGROUND)
        self.headerL.pack(side=LEFT)
        self.headerR = tk.Frame(self.header, background=BACKGROUND)
        self.headerR.pack(side=LEFT)
        self.top = tk.Frame(self.headerR, background=BACKGROUND)
        self.top.pack(side=TOP)

        self.logo =
Image.open(os.path.join("assets","logo.png")).resize((150,150))
        self.logotk = ImageTk.PhotoImage(image=self.logo)
        self.logo_frame = tk.Label(self.headerL, image= self.logotk,
```

```
background=BACKGROUND)
        self.logo_frame.pack(side=LEFT)

        self.label = tk.Label(self.top, text='Deteksi Keretakan
Jalan', fg='black', bg='#4bacc6', padx=5, pady=5)
        self.label.pack(anchor='n', side='top', padx=20, pady=3)
        self.label.config(font=("Comic Sans", 32, 'bold'))

        # define frame
        self.sub = tk.Frame(self.headerR, background=BACKGROUND,
padx=10, pady=3)
        self.sub.pack(side=TOP, pady=3)
        self.subL = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subL.pack(side=LEFT, pady=3)
        self.subC = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subC.pack(side=LEFT, pady=3)
        self.subR = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subR.pack(side=LEFT, pady=3)
        self.bottom = tk.Frame(self, background=BACKGROUND, padx=10,
pady=3)
        self.bottom.pack(side=TOP)
        self.frame = tk.Frame(self, background=BACKGROUND, padx=10,
pady=3)
        self.frame.pack(side=TOP, pady=3)
        self.frameL = tk.Frame(self.frame, background=BACKGROUND,
padx=10, pady=3)
        self.frameL.pack(side=LEFT, pady=3)
        self.frameR = tk.Frame(self.frame, background=BACKGROUND,
padx=10, pady=3)
        self.frameR.pack(side=LEFT, pady=3)
        self.frTop = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frTop.pack(side=TOP, pady=3)
        self.frMid = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frMid.pack(side=TOP, pady=3)
        self.frBtm = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frBtm.pack(side=BOTTOM, pady=3)

        self.folderPath = tk.StringVar()
        self.inputlabel = tk.Label(self.subL,text="Masukkan Folder")
        self.inputlabel.pack(side=LEFT, pady=10)
        self.inputtext =
tk.Entry(self.subC,textvariable=self.folderPath, width=50)
        self.inputtext.pack(side=LEFT, pady=10)
        self.btnFind = ttk.Button(self.subR, text="Browse
```

```
Folder",command=self.folderpath)

          self.btnFind.pack(side=LEFT, pady=10)
        self.options_list = ["Approximation", "Horizontal Detail",
"Vertical Detail", "Diagonal Detail"]
        self.value_inside = tk.StringVar()
        self.value_inside.set("Pilih Transformasi")

        self.button1 = tk.Button(self.bottom, text="Selesai",
command=self._quit)
        self.button1.pack(side=RIGHT, pady=2, padx=10)
        self.button2 = tk.Button(self.bottom, text="Proses Gambar",
command=self.doStuff)
        self.button2.pack(side=LEFT, pady=2, padx=10)

        self.question_menu = tk.OptionMenu(self.bottom,
self.value_inside, *self.options_list)
        self.question_menu.pack(side=TOP, pady=2, padx=10)

        self.imgfilename = tk.StringVar()

        self.num = tk.IntVar()
    def _quit(self):
        self.quit()
        self.destroy()

    def folderpath(self):
        self.folder_selected = filedialog.askdirectory()
        self.folderPath.set(self.folder_selected)

    def doStuff(self):
        if isinstance(self.num,int):
            self.filename.destroy()
            self.resultlabel.destroy()
            self.image_frame.destroy()
            self.prev.destroy()
            self.next.destroy()
        root_folder = self.folderPath.get()
        transform = self.value_inside.get()
        print("Doing stuff with folder", root_folder, transform)
        self.image_path = []
# membuat list kosong untuk lokasi image
        for img in os.listdir(root_folder):
# loop setiap folder yang ada dalam folder data
            self.image_path.append(os.path.join(root_folder,img))
# tambahkan path gambar ke list image_path
        self.image = []
# membuat list kosong untuk image array
        self.imageH = []
# membuat list kosong untuk image array
        self.imageV = []
```

```
# membuat list kosong untuk image array
        self.imageD = []

# membuat list kosong untuk image array
        for img in tqdm(self.image_path):
# loop setiap image path
            arr = cv2.imread(img)
# membaca image
            arr = cv2.resize(arr,shape)
# resize image
            arr = cv2.cvtColor(arr, cv2.COLOR_BGR2LAB)
# convert BGR ke CIE Lab
            l, a, b = cv2.split(arr)
# split image channels
            clahe = cv2.createCLAHE(clipLimit=4.0,
tileGridSize=(8,8))                                                    #
membuat fungsi adaptive histogram equalization, untuk memperbaiki
contrast
            cl = clahe.apply(l)
# apply fungsi adaptve histogram equalization terhadap channel l
            arr = cv2.merge((cl,a,b))
# merge kembali ketiga channle gambar
            arr = cv2.cvtColor(arr, cv2.COLOR_LAB2BGR)
# convert CIE Lab ke BGR
            arr = cv2.cvtColor(arr,cv2.COLOR_BGR2GRAY)
# convert ke grayscale
            coeffs2 = pywt.dwt2(arr, 'bior1.3')
# transformasi Wavelet
            LL, (LH, HL, HH) = coeffs2
# membuat variable untuk setiap approximation
            self.image.append(LL.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageH.append(LH.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageV.append(HL.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageD.append(HH.tolist())
# add hasil transformasi Wavelet ke list image
        if transform == "Horizontal Detail":
            X = np.array(self.imageH)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        elif transform == "Vertical Detail":
            X = np.array(self.imageV)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        elif transform == "Diagonal Detail":
            X = np.array(self.imageD)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
```

```
else:
    X = np.array(self.image)/255
```

```
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        input_shape = (X.shape[1],X.shape[2])

# membuat variable input_shape sebagai input shape model
        X = X.reshape(-1, input_shape[0], input_shape[1], 1)
# reshape X
        predictions = model.predict(X)
# membuat prediksi terhadap data
        predictions = np.argmax(predictions, axis=-1)
# konversi hasil prediksi sesuai dengan bentuk array mula-mula
        self.results = le.inverse_transform(predictions)
# convert ke label semula
        df = pd.DataFrame({
            "image":self.image_path,
            "prediction":self.results})
        df['image'] =
df['image'].str.split("/",expand=True).iloc[:,-1]
        self.table = pt = Table(self.frameL, dataframe=df,
                                    showtoolbar=True,
showstatusbar=True, width=500)
        pt.show()
        self.num = 0
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)
        self.img =
Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)
        self.prev = tk.Button(self.frBtm, text="Previous",
command=self._previous)
        self.prev.pack(side=LEFT, pady=2, padx=10)
        self.next = tk.Button(self.frBtm, text="Next",
command=self._next)
        self.next.pack(side=LEFT, pady=2, padx=10)
    def _next(self):
        if self.num == len(self.image_path):
```

```
    self.num = self.num
        else:
            self.num+=1
        self.filename.destroy()
        self.resultlabel.destroy()
        self.image_frame.destroy()
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)

self.img = Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)
    def _previous(self):
        if self.num == 0:
            self.num = self.num
        else:
            self.num-=1
        self.filename.destroy()
        self.resultlabel.destroy()
        self.image_frame.destroy()
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)
        self.img =
Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)

road = root()
road.mainloop()
# ==== GUI ==== #
```

Lampiran C Citra Pengujian

Tabel C.1 Gambar retak jalan raya untuk pengujian GUI

No	Foto kondisi jalan	Keterangan
1		Retak Buaya
2		Retak Buaya
3		Retak Buaya

4		Retak Buaya
5		Retak Buaya
6		Retak Buaya
7		Retak Garis

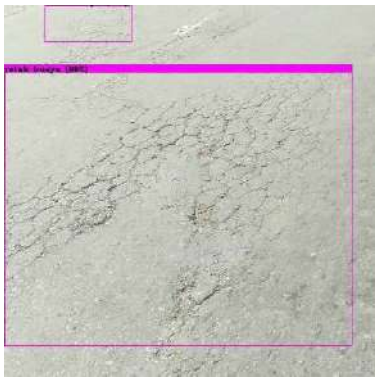
8		Retak Garis
9		Retak Garis
10		Retak Garis
11		Retak Garis

12		Retak Garis
13		Tidak Retak
14		Tidak Retak
15		Tidak Retak

16		Tidak Retak
17		Tidak Retak

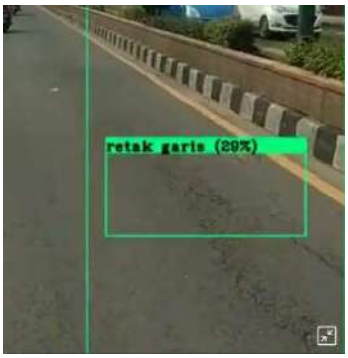
Tabel B.2 Gambar retak jalan raya untuk pengujian YOLO

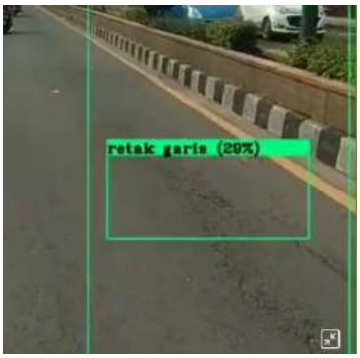
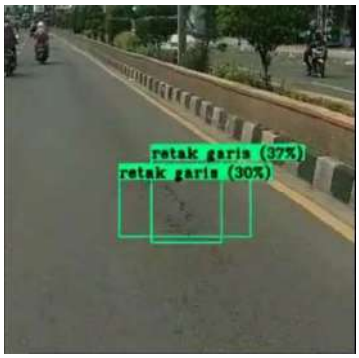
No	Foto kondisi jalan	Keterangan
1		Retak Garis

2		Retak Buaya
3		Retak Buaya
4		Retak Buaya

5		Retak Buaya
6		Retak Buaya
7		Retak Buaya

8		Retak Buaya
9		Retak Buaya
10		Retak Garis

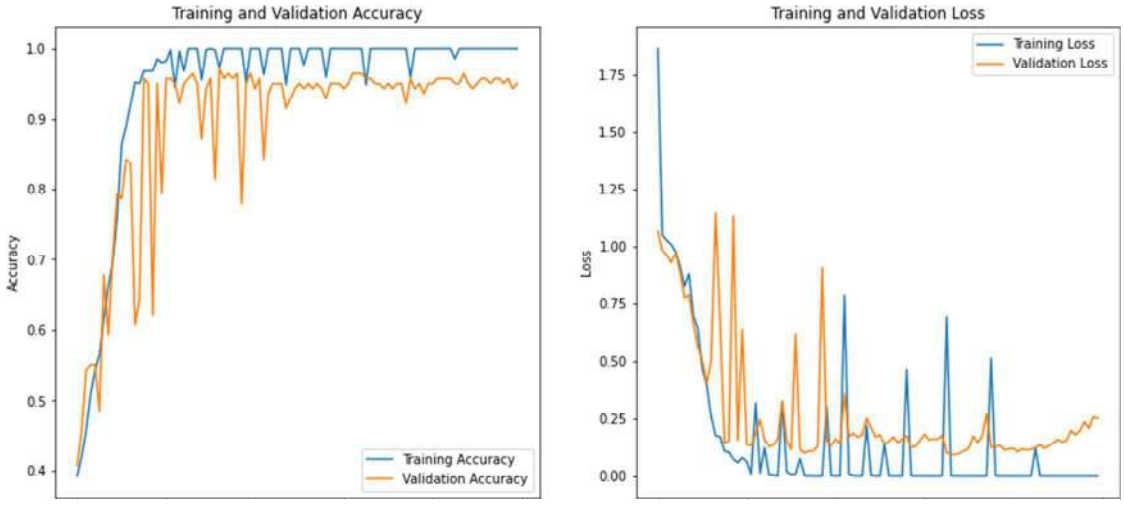
11		Retak Garis
12		Retak Garis
13		Retak Garis
14		Retak Garis

15		Retak Buaya
16		Retak Garis

LAMPIRAN D Pelatihan CNN

D.1 Pelatihan dengan variasi *epoch*

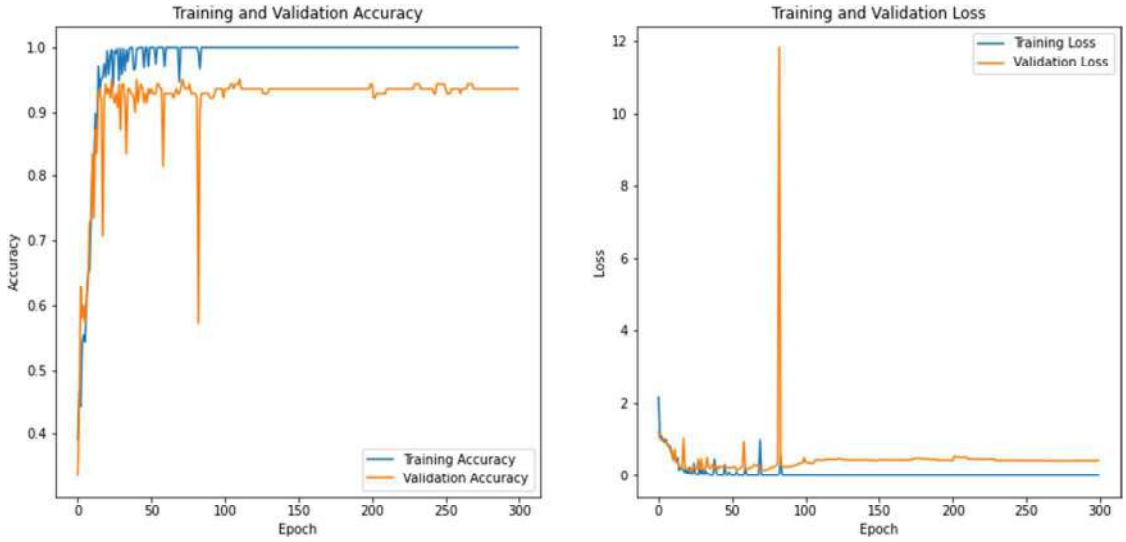
Dari grafik yang terdapat pada Gambar D.1, kita dapat melihat perkembangan *loss* dan akurasi selama pelatihan dengan total 100 *epoch*. Hasilnya menunjukkan bahwa akurasi pelatihan mencapai 100%, sedangkan akurasi validasi mencapai 95% setelah melewati 100 *epoch* dengan *learning rate* 0,001. Di sini, "*epoch*" merupakan parameter yang mengindikasikan jumlah iterasi penuh terhadap seluruh dataset yang mengalami proses pelatihan dalam model arsitektur *deep learning*. Dengan pengaturan 100 *epoch*, seluruh dataset telah menjalani pelatihan sebanyak 100 kali. Sementara itu, *loss* pelatihan mencapai 0,00, dan *loss* validasi mencapai 0,251..



Gambar D.1 Grafik Hasil *Training Validation Epoch* 100

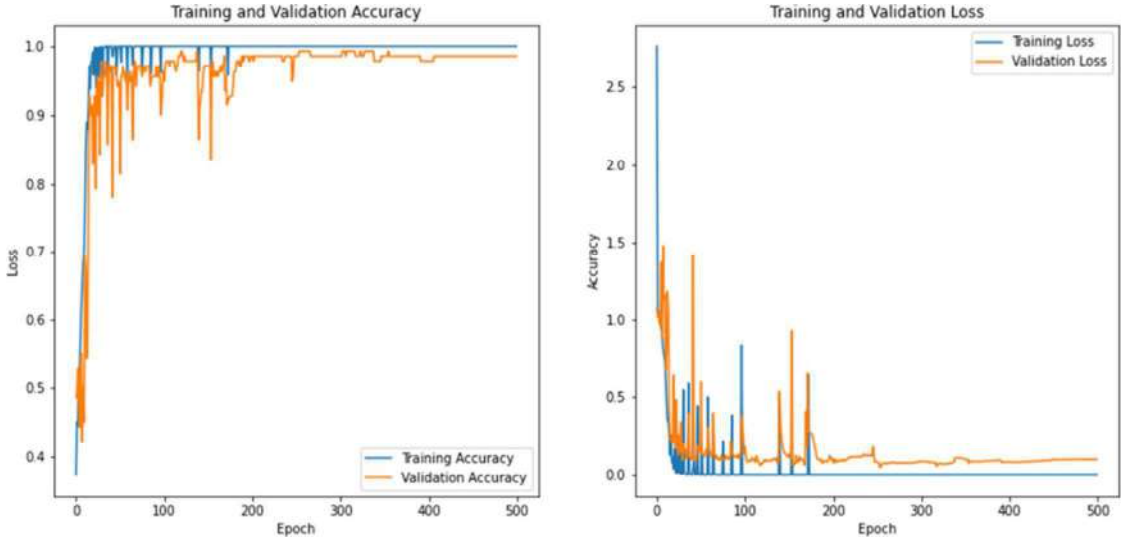
Gambar D.2 menunjukan grafik *loss* dan akurasi pelatihandengan *epoch* 300 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 93,57 % setelah melalui 300 *epochs* dengan *learning rate* 0,001. *Epoch* adalah parameter untuk seluruh dataset yang sudah melalui proses pelatihan pada model arsitektur *deep learning* untuk sekali putaran penuh, dan jika mengatur dengan 100 *epoch* artinya seluruh dataset mengalami proses pelatihan sebanyak 100 kali.

Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,421.



Gambar D.2 Grafik Hasil *Training Validation Epoch* 300

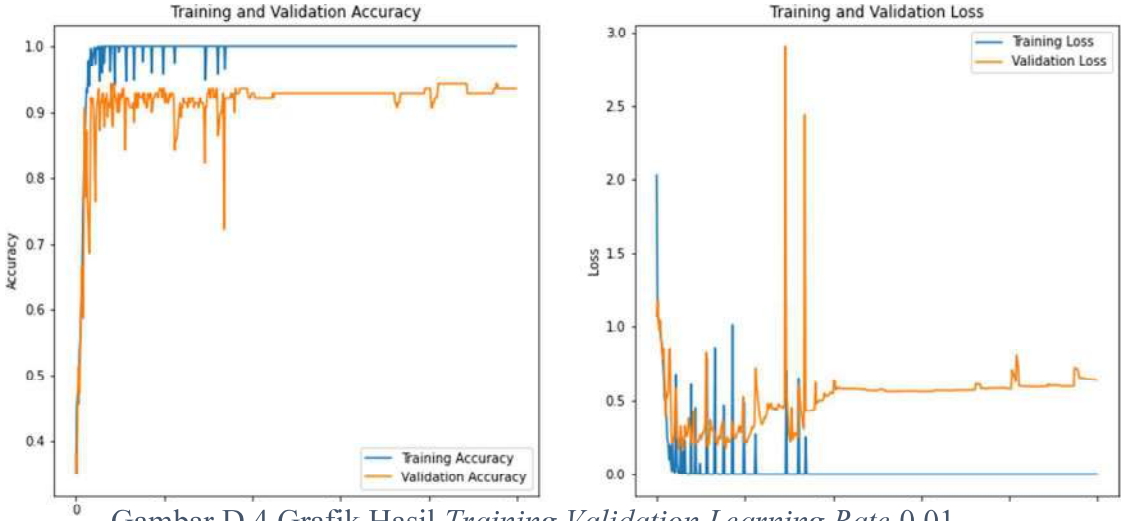
Gambar D.3 gambar grafik *loss* dan akurasi pelatihandengan *epoch* 500 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 96,43 % setelah melalui 500 *epochs* dengan *learning rate* 0,001. *Epoch* adalah parameter untuk seluruh dataset yang sudah melalui proses pelatihan pada model arsitektur *deep learning* untuk sekali putaran penuh, dan jika mengatur dengan 100 *epoch* artinya seluruh dataset mengalami proses pelatihan sebanyak 100 kali. Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,3636.



Gambar D.3 Grafik Hasil *Training Validation Epoch* 500

D.2 Pelatihan Dengan Variasi *Learning rate*

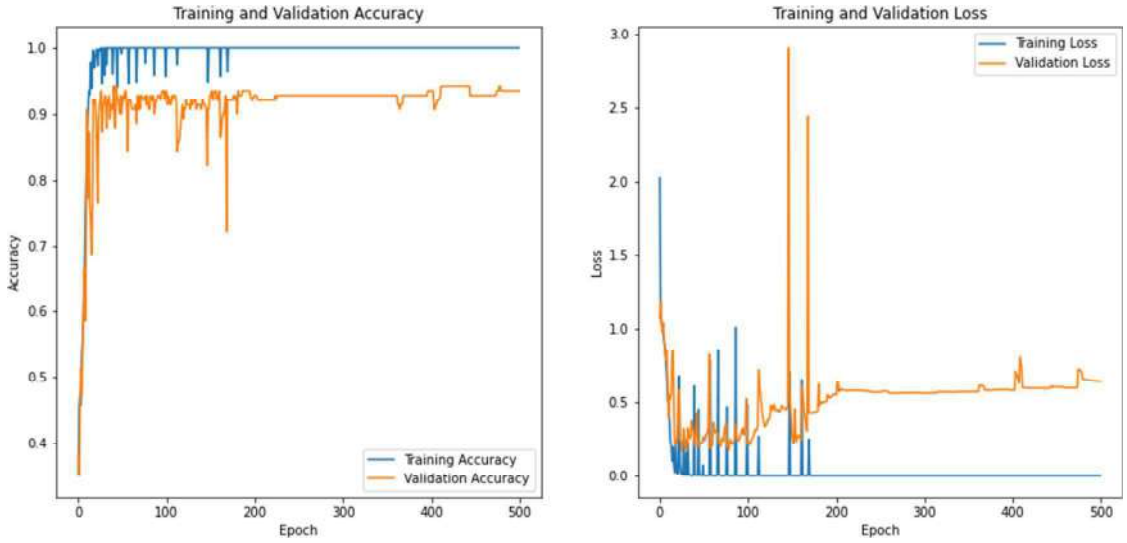
Gambar D.4 menunjukan grafik *loss* dan akurasi pelatihandengan *learning rate* 0,01 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 93,57 % setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menhitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss*mencapai 0,00 dan *validation loss*



Gambar D.4 Grafik Hasil *Training Validation Learning Rate* 0,01

mencapai 0,6413.

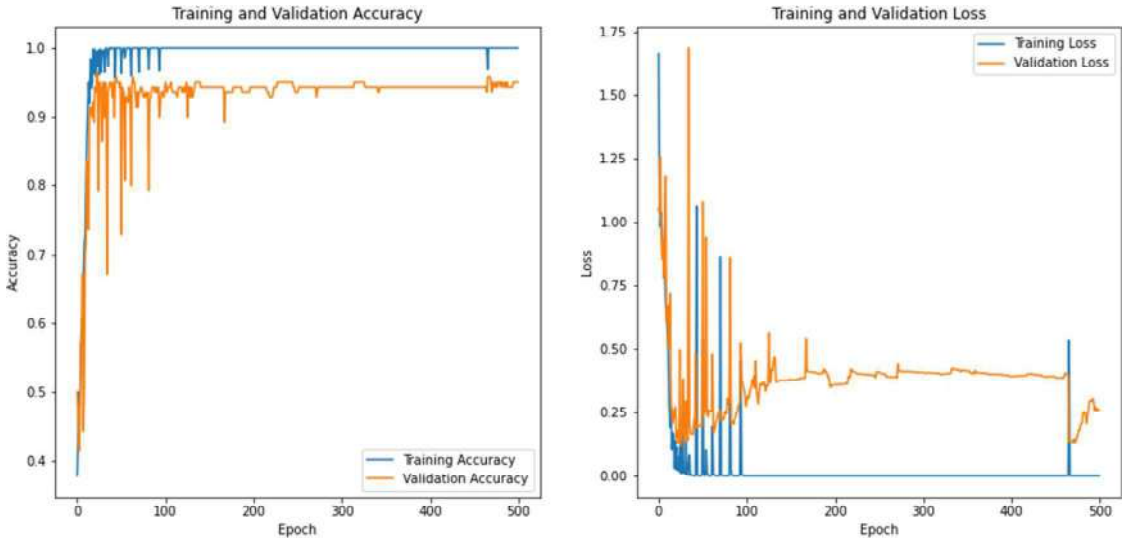
Pada gambar D.5 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate* 0,001 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 96,43 % setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menhitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,3636.



Gambar D.5 Grafik Hasil *Training Validation Learning Rate* 0,001

Dari gambar D.6 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate* 0,0001 dapat dilihat bahwa *training accuracy* mencapai angka100 % dan *validation accuracy* mencapai 95% setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menghitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss* mencapai0,00 dan *validation loss* mencapai 0,26.

Dari gambar D.5 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate*



Gambar D.6 Grafik Hasil *Training Validation Epoch* 0,0001