LAMPIRAN

Lampiran 1. *Source Code*

```
from flask import *
import numpy as np
import glob
import os
import shutil
from shutil import copyfile
from sklearn.model_selection import StratifiedShuffleSplit
from skimage.io import imread
from skimage.transform import resize
from skimage.feature import hog
from skimage import exposure
import matplotlib.pyplot as plt
import joblib
import cv2
import pathlib

import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import load_model
From sklearn.metrics import classification_report,
confusion_matrix
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras import applications
from tensorflow.keras.regularizers import l2
# import tensorflow as tf

covid_dataset="./static/dataset/COVID"
non_covid_dataset="./static/dataset/NONCOVID"
hog_covid_dataset="./static/hog_dataset/COVID"
hog_non_covid_dataset="./static/hog_dataset/NONCOVID"
resize_hog_covid_dataset="./static33/resize_dataset/COVID"
```

```
        resize_hog_non_covid_dataset="./static/resize_dataset/NONCOVID"
        k_fold_dataset='./static/'

        fold_validation = Blueprint('fold_validation', __name__,
            template_folder='templates',
            static_folder='static')

        @fold_validation.route('/')
        def index():
            static_folder=os.listdir(k_fold_dataset)
            fold_folder=[]
            for f in static_folder:
                if "fold" in f:
                    fold_folder.append(f)

            result_joblib=glob.glob("./static/result/*_result.joblib")
            data_result=[]
            table=""
            for f in result_joblib:
                d=joblib.load(f)
                data_result.append(d)
                # table="<tr>"
                # table=table+"<td>"+d[0][1]+"</td>"
                # table=table+"<td>"+d[1][1]+"</td>"
                #  table=table+"<td>"+d[2][1]+", "+d[3][1]+", "+d[4][1]+",
"+d[5][1]+"</td>"
                # table=table+"<td>"+d[6][1]+", "+d[7][1]+", "+d[8][1]+",
"+d[9][1]+", "+d[10][1]+"</td>"
                #  table=table+"<td><img  src='."+d[11][1]+"'  class='img-
center' width='150' height='150' style='cursor: pointer;'></td>"
                # table=table+"</tr>"

           return  render_template('fold_validation/fold_validation.html',
fold=fold_folder, data_result=data_result)

        @fold_validation.route('/getFold', methods=['POST'])
        def getFold():
            fold=request.form['fold']
            train0=glob.glob('./static/'+fold+"/train/NONCOVID/*.jpg")
            train1=glob.glob('./static/'+fold+"/train/COVID/*.jpg")
            test0=glob.glob('./static/'+fold+"/test/NONCOVID/*.jpg")
```

```
            ["False Negative : ", str(confusion[0][1])],
            ["Accuracy : ", Accuracy],
            ["image", './static/result/'+experiment_name+'.png']]

            filename_test=[]
            label_name=[]
            pred_name=[]
            path_with_name=[]
            for  f  in  zip((test_file_covid+test_file_non_covid),  label,
pred):
                filename_test.append(os.path.basename(f[0]))
                label_name.append(f[1])
                pred_name.append(f[2])

path_with_name.append(pathlib.PurePath(f[0]).parent.name+'\\'+os.path.bas
ename(f[0]))

            detail_result=[filename_test,   path_with_name,   label_name,
pred_name, temp_predict]

            # print(confusion)
            # print(report)

            np.savetxt("./static/result/"+experiment_name+".csv",
np.array(result, dtype='str'), fmt="%s")
            joblib.dump(result,
"./static/result/"+experiment_name+"_result.joblib")
            joblib.dump(detail_result,
"./static/result/"+experiment_name+"_detail.joblib")
            joblib.dump([history.history['acc'],
history.history['val_acc'],                      history.history['loss'],
history.history['val_loss']],
"./static/result/"+experiment_name+"_plot.joblib")

        @fold_validation.route('/train', methods=['POST'])
        def train():
            fold=request.form['fold']
            method=request.form['method']
            batch=request.form['batch']
            learning_rate=request.form['learning_rate']
            name=request.form['name']
```

```
            epoch=request.form['epoch']

            if os.path.isfile("./static/result/"+name+".png"):
                os.remove("./static/result/"+name+".png")
            if os.path.isfile("./static/result/"+name+".joblib"):
                os.remove("./static/result/"+name+".joblib")
            if os.path.isfile("./static/result/"+name+".csv"):
                os.remove("./static/result/"+name+".csv")

            # CNN(name, float(learning_rate), int(batch), int(epoch),
"./static/"+fold+"/train", "./static/"+fold+"/test", "./model", fold,
method)
            # CNN(name, float(learning_rate), int(batch), int(epoch),
"./static/"+fold+"/train", "./static/"+fold+"/test", "./model", fold,
method)

            MYCNN("./static/"+fold+"/train/COVID/*.jpg",
            "./static/"+fold+"/train/NONCOVID/*.jpg",
            "./static/"+fold+"/test/COVID/*.jpg",
            "./static/"+fold+"/test/NONCOVID/*.jpg",
            64,
            64,
            float(learning_rate),
            int(batch),
            int(epoch),
            "./static/model",
            name,
            fold,
            method)

            return jsonify("done")
```