DAFTAR PUSTAKA

Cheung N, Mitchell P, Wong TY. Diabetic retinopathy. The Lancet 2010;376:124–36.

S. Sharma, A. Oliver-Fernandez, W. Liu, P. Buchholz, and J. Walt, "The impact of diabetic retinopathy on health-related quality of life," Curr. Opin. Ophthalmol., vol. 16, no. 3, pp. 155–159, 2005.

Mayo Foundation For Medical Eduaction And Research. (2023). Diabetic Macular Edema. https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy /multimedia/ diabetic-macular-edema/img-20124558

R. Gargeya dan T. Leng, "*Automated Identification of Diabetic Retinopathy Using Deep Learning*," Ophthalmology, vol. 124, no. 7, pp. 962–969, 2017.

Ozieh MN, Bishu KG, Dismuke CE, Egede LE. (2015). *Trends in health care expenditure in U.S. adults with diabetes*: 2002 – 2011. Diabetes Care. 2015;38:1844-1851

James Kang Hao Goh, BSc, Carol Y. Cheung, PhD, dkk. (2016). *Retinal Imaging Techniques for Diabetic Retinopathy Screening. Journal of Diabetes Science and Technology Volume 10, Issue 2*, *March 2016*, *Pages 282-294*. Doi:10.1177/1932296816629491

M. E. Brezinski et al. (1996), "*Optical coherence tomography* for *optical* biopsy: Properties and demonstration of vascular pathology," Circulation, vol. 93, no. 6, pp. 1206–1213, 1996

Ashan, Haves. (2020). Karakteristik Ketebalan Makula Sentral Sebelum dan Sesudah Injeksi IntravitrealBevacizumab di RSKM Padang Eye Center Periode Januari 2018 – Januari 2019. *Health & Medical Journal*. 2. 01-07. 10.33854/heme.v2i2.451.

Managing Diabetic Eye Disease in Clinical Practice. (2015). Germany: Springer International Publishing.

State of the Art in Neural Networks and Their Applications: Volume 1. (2021). Belanda: *Elsevier Science.*

I. Goodfellow, Y. Bengio, A. Courville, *Convolutional networks, Deep Learning*, MIT Press, 2016. Ch. 9.

Wang, Yu, Lin, Zhong, Zhai, G., Ding, Xiao Xia, wen, liang, li, dong, Zou, Bo, Mi Feng, Ke, Bo Liang, Bo, xie, cong. (2020). *Prevalence of and risk factors for diabetic retinopathy and diabetic macular edema in patients with early and late onset diabetes mellitus. Ophthalmic Research.* doi:10.1159/000508335

Klein R, Knudtson MD, Lee KE, Gangnon R, Klein BE. The Wisconsin

Epidemiologic Study of Diabetic Retinopathy: XXII the twenty-five-year progression of retinopathy in persons with type 1 diabetes. *Ophthalmology.* 2008;115(11):1859–1868.

Deniston, Alistair K. O., dan Philip I. Murray (eds). (2018). *Oxford Handbook of Opthamology*, 4 edn, *Oxford Medical Handbook*. doi:10.1093/med/9780198804550.001.0001

LeCun, Yann, Yoshua Bengio, Geoffrey Hinton. (2015). *Deep Learning*. New York University Vol 521.

Bridge Maintenance, Safety, Management, Life-Cycle Sustainability and Innovations: Proceedings of the Tenth International Conference on Bridge Maintenance, Safety and Management (IABMAS 2020), 28 Juni - 2 Juli, 2020, Sapporo, Jepang. United Kingdom: CRC Press, 2021.

Sewak, M., Karim, M. R., Pujari, P. (2018). Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python. United Kingdom: Packt Publishing.

Rajappa, S., Paneerselvam, S., Kumar, L. A., Sumathi, S. (2022). *Machine Learning for Decision Sciences with Case Studies in Python*. United States: CRC Press.

Kermany, Daniel S., et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning." *Cell* 172.5 (2018): 1122-1131.

Uzair, M., & Jamil, N. (2020). Effects of hidden layers on the efficiency of neural networks. In 2020 IEEE 23rd international multitopic conference (INMIC) (pp. 1-6). IEEE.

LAMPIRAN

1. Dataset

Dataset yang digunakan pada penelitian ini dapat dilihat pada pranala berikut. https://www.kaggle.com/paultimothymooney/kermany2018

2. Source Code

```
import os
from glob import glob
import matplotlib.pyplot as plt
import random
import cv2
import pandas as pd
import numpy as np
import matplotlib.gridspec as gridspec
import seaborn as sns
import zlib
import itertools
import sklearn
import itertools
import scipy
import skimage
from skimage.transform import resize
import csv
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")
from sklearn import model_selection
from sklearn.model_selection import train_test_split, KFold, cross_val_score, StratifiedKFold, GridSearchCV
from sklearn.utils import class_weight
from sklearn.metrics import confusion_matrix, make_scorer, accuracy_score, classification_report
import keras
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, Lambda, MaxPool2D, BatchNormalization
from keras.utils import np_utils
from keras.utils.np_utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
from keras import models, layers, optimizers
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.utils import class_weight
from keras.optimizers import SGD, RMSprop, Adam, Adagrad, Adadelta, RMSprop
from keras.models import Sequential, model_from_json
from keras.layers import Activation,Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.layers import MaxPooling2D,AveragePooling2D, GlobalAveragePooling2D,BatchNormalization
from keras.preprocessing.image import ImageDataGenerator #array_to_img, img_to_array, load_img
from tensorflow.keras.utils import array_to_img, img_to_array, load_img
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from keras import backend as K
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.applications.mobilenet import MobileNet
from keras.applications.inception_v3 import InceptionV3
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

```

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    plt.show()

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

image_size = 256
batch_size = 16
num_classes = 2
epochs = 10

train_datagen = ImageDataGenerator(validation_split=0.2)

train_generator = train_datagen.flow_from_directory('G:/OCT2017/Train', target_size=(image_size, image_size),
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    subset='training')

validation_generator = train_datagen.flow_from_directory('G:/OCT2017/train', target_size=(image_size, image_size),
                                                         batch_size=batch_size,
                                                         class_mode='categorical',
                                                         subset='validation')

test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow_from_directory("G:/OCT2017/test", target_size=(image_size, image_size),
                                                  batch_size=batch_size,
                                                  class_mode='categorical')
```

```
pred_datagen = ImageDataGenerator()

pred_generator = pred_datagen.flow_from_directory("G:/OCT2017/test",target_size=(image_size, image_size),
                                                  batch_size=1,
                                                  class_mode='categorical',
                                                  shuffle = False)


class_weights = dict(zip(np.unique(train_generator.classes),
                class_weight.compute_class_weight(class_weight = 'balanced',
                                                  classes = np.unique(train_generator.classes),
                                                  y = train_generator.classes)))

filepath="weights_balanced_cnn_6layered_best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

model = Sequential()
model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', input_shape=(image_size, image_size, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))
```

```
159
160  print(model.summary())
161
162  model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
163
164
165  history = model.fit(train_generator,
166                      steps_per_epoch = train_generator.samples // batch_size,
167                      validation_data = validation_generator,
168                      validation_steps = validation_generator.samples // batch_size,
169                      epochs = epochs,
170                      callbacks=callbacks_list,
171                      class_weight = class_weights
172                  )
173
174
175  score = model.evaluate_generator(test_generator,steps = test_generator.samples // batch_size)
176  print("\n\n")
177  print('Test Loss:', score[0])
178  print('Test accuracy:', score[1])
179
180  fig,ax = plt.subplots(1,1)
181  ax.set_xlabel('epoch')
182  ax.set_ylabel('Categorical Crossentropy Loss')
183
184  x = list(range(1,epochs+1))
185
186  vy = history.history['val_loss']
187  ty = history.history['loss']
188  plt_dynamic(x, vy, ty, ax)
```