DAFTAR PUSTAKA

Anubhav, S., & Bhadani, R. (2020). *Mobile Deep Learning with Tensorflow Lite, ML Kit and Flutter*. Birmingham, UK: Packt Publishing Ltd.

Bhuma, C., & Kongara, R. (2020). *Childhood Medulloblastoma Classification Using EfficientNets. IEEE Bombay Section Conference (IBBSC).*

Bintang, & Thunder. (2018). Implementasi Algoritma *Convolutional Neural Network* di *Microsoft Azure* Untuk mendeteksi Jenis Kebutaan Yang diAlami Penderita Diabetes.

Ichi. (2020). Cara cerdas untuk meningkatkan performa *Convolutional Neural Network: EfficientNet Google AI*.

IDF 10th Edition. (2021). IDF Atlas 10th Edition.

Institut National Eye (NEI). (2022, July 8). *Diabetic Retinopathy.*

Jackson, P. (2019). *Style Augmentation : Data Augmentation via Style Randomization. (p. 85). UK: IEEE Xplore*.

Jeremy, H., & Gugger, S. (2020). *Deep Learning for Codes with Fastai and PyTorch.* Canada: O Reilly Media,INC.

Kandel, I., & Mauro, C. (2020). Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification A Review. *MDPI.*

Kirtika.B., P. V. (2015). Android Operating System: A Review. *international Journal of Trend in Research and Development, Volume 2(5), ISSN 2394-9333 www.ijtrd.com.*

Levy, J. J. (2020). PathFlowAI: A High-Throughput Workflow for Preprocessing, *Deep Learning* and Interpretation in Digital Pathology. *b3805 Pacific Symposium on Biocomputing 20208.5"x11"403 Pacific Symposium on Biocomputing 202* (p. 404). World scientific.

Rahman, S., Marwan, R., Fitri, A., Rusdha, M., Muhammad, Z., & Muhammad, I. (2021). *Convolutional Neural Networks Untuk Visi Komputer* . Yogyakarta: Cv Budi Utama.

Russo, A. M. (2019). Classification of sports videos with combination of *Deep Learning* model and transfer learning. *International Conference on Electrical, Computer and Communication Engineering (ECCE)* (p. 2). Ulsan, Korea: IEEE.

Sabery.E.F.,dkk. (2021). *Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier.* Spanyol: IEEE Access.

Sari, N. M., & Saraswati, M. R. (2013). *Prevalensi Retinopati Diabetika Pada Pasien Diabetes Melitus Tipe 2 di Poliklinik Penyakit Dalam RSUP Sanglah Denpasar.* Denpasar.

Shaniaputri.,dkk. (2022). Prevalensi Retinopati Diabetik di Puskesmas di Bandung Raya Periode Januari 2019-2020. *eJKI Vol. 10, No. 1, April 2022*.

Suartika.,dkk. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) PAda Caltech 101. *JURNAL TEKNIK ITS Vol. 5, No. 1, (2016) ISSN: 2337-3539 (2301-9271 Print)*, A65-A66.

Tan, M., & Le, Q. V. (2019). "EfficientNet: Rethinking Model Scalling for Convolutional Neural Networks". *International Conference on Machine `Learning*, vol. 97.

Xu., dkk. (2020). Three-way confusion matrix for classification: A measure driven view. In *Information Sciences, Vol 507* (pp. 772-794). Chine: Science Direct, https://doi.org/10.1016/j.ins.2019.06.064.

LAMPIRAN

Code.ipynb

In [1]

```
#import library
import numpy as np
import tensorflow as tf
import pandas as pd
import random
import os
import matplotlib.pyplot as plt

from tensorflow.keras.layers import Dense,
Flatten, Input, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from tensorflow.keras.callbacks import
EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.applications.efficientnet
import EfficientNetB0, EfficientNetB1,
preprocess_input
import efficientnet.keras as efn

from sklearn.metrics import confusion_matrix,
classification_report,
precision_recall_fscore_support, accuracy_score
```

In [2]

```
#cek versi tensorflow
print(tf.__version__)
print(tf.config.list_physical_devices('GPU'))
```

In [3]

```
#membuat Seed (Inisiasi Generator Bilangan Acak)
seed = 101
np.random.seed(seed)
tf.random.set_seed(seed)
random.seed(seed)
os.environ['PYTHONHASHSEED'] = str(seed)
```

In [4]

```
#load dataset
orig_path = 'gaussian_filtered_images/'

paths = []
labels = []

for root, dirs, files in os.walk(orig_path):
    path = root.split(os.sep)
    for file in files:
        current_path = '/'.join(path)+'/'+file
        paths += [current_path]
        labels += [current_path.split('/')[-2]]
```

In [5]

```
#menggabungkan path dan labels
list_combined = map(list, zip(*[paths, labels]))
```

In [6]

```
#membuat data frame
df = pd.DataFrame(list_combined,columns=['path',
'label'])
```

In [7]

```
df.loc[df['label'] != 'No_DR', 'label'] = 'DR'
#df.loc[df['label'] != '0', 'label'] = '1'
df
```

In [8]

```
#Acak dataframe & menyetel ulang index
df = df.sample(frac=1,
random_state=seed).reset_index(drop=True)
df
```

In [9]

```
#Membuat sample data train&test dari datafreame
train_index = df.index.isin(df.sample(frac=0.8,
random_state=seed).index)
train_df = df[train_index]
test_df = df[~train_index]
print(f'train: {len(train_df)}, test:
{len(test_df)}')
```

In [10]

```
#Generator augmentasi
IMG_SIZE = 224
gen = ImageDataGenerator(
    rescale=1.0 / 255,
    #rotation_range=15,
    #width_shift_range=0.15,
    #height_shift_range=0.15,
    #shear_range=0.5,
    #zoom_range=0.2,
    #vertical_flip=True,
    #horizontal_flip=True,
    preprocessing_function=preprocess_input,
    fill_mode='nearest',
    validation_split=0.20
)

train_gen = gen.flow_from_dataframe(
    train_df, x_col='path', y_col='label',
subset="training", shuffle=False, target_size=(IMG_SIZE,
IMG_SIZE),
    class_mode="categorical", seed=seed
)
gen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.20,
    fill_mode='nearest',
    preprocessing_function=preprocess_input
)
val_gen = gen.flow_from_dataframe(
    train_df, x_col='path', y_col='label',
subset="validation", shuffle=False,
target_size=(IMG_SIZE, IMG_SIZE),
    class_mode="categorical", seed=seed
)
```

In [11]

```
#Menghitung jumlah data training & validasi
classes = list(train_gen.class_indices.keys())
_, train_counts = np.unique(train_gen.classes,
return_counts=True)
_, val_counts = np.unique(val_gen.classes,
return_counts=True)
_, test_counts = np.unique(test_gen.classes,
return_counts=True)
print('number of classes in each set:')
counts = pd.DataFrame([train_counts, val_counts,
test_counts], columns=classes, index=['train count', 'val
count', 'test count']).T
counts['total'] = counts['train count'] + counts['val
count'] + counts['test count']
print('total training data:', np.sum(counts['train count']))
print('total validation data:', np.sum(counts['val count']))
print('total testing data:', np.sum(counts['test count']))
counts
```

In [12]

```
#Inisiasi Model
model = efn.EfficientNetB0(weights='imagenet',
include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
model.trainable = False #membekukan bobot
```

In [13]

```
#Fully Connected Layer
x = model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(2, activation='softmax')(x)
model = Model(inputs=model.input,
outputs=predictions)
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
loss='categorical_crossentropy',
metrics=['accuracy'])
#model.summary()
```

In [14]

```
#Menentukan jumlah epoch & batch size
EPOCHS = 30
BATCH_SIZE = 32
```

In [15]

```
#Early stopping & Checkpoint
early_stopping = EarlyStopping(monitor='val_loss',
patience=EPOCHS)
checkpoint = ModelCheckpoint('checkpoint/check',
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=True, save_freq='epoch')
```

In [16]

```
#proses train & validation model EfficientNet
history = model.fit(train_gen,
validation_data=val_gen, epochs=EPOCHS,
callbacks=[early_stopping, checkpoint])
```

In [17]

```
#dataframe hasil train % validasi per epoch
epoch_result =
pd.DataFrame([history.history['accuracy'],
history.history['val_accuracy'],
history.history['loss'], history.history['val_loss'],
], columns=np.arange(1,EPOCHS+1,step=1),
index=['accuracy', 'val_accuracy', 'loss',
'val_loss']).T
epoch_result.index.name='epoch'
for col in epoch_result:
    if col == 'loss' or col == 'val_loss':
        continue
    epoch_result[col] = epoch_result[col] * 100
epoch_result = epoch_result.round(2)
epoch_result
```

In [18]

```
#plot akurasi model efficientNet
plt.figure(figsize=(12,8))
plt.plot(epoch_result['accuracy'],'-o',
label='training', color='#e895a1')
plt.plot(epoch_result['val_accuracy'], '-o',
label='validation', color='#7070ff')
plt.title('efficientnet accuracy')
plt.ylim(80,100.5)
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks(np.arange(1, EPOCHS+1, step=1))
plt.grid()
plt.legend()
```

In [19]

```
#plot loss akurasi model efficientNet
plt.figure(figsize=(12,8))
plt.plot(epoch_result['loss'], '-o', label='training',
color='#e895a1')
plt.plot(epoch_result['val_loss'], '-o',
label='validation', color='#7070ff')
plt.title('efficientnet loss')
plt.xlabel('epoch')
plt.ylabel('loss')
#plt.xticks(np.arange(1, EPOCHS+1, step=1))
plt.grid()
plt.legend()
```

In [20]

```
accuracy = accuracy_score(test_gen.classes,
y_pred_test)
test_result =
pd.DataFrame(precision_recall_fscore_support(test_gen.
classes, y_pred_test), columns=classes,
index=['precision', 'recall', 'f1 score',
'support']).T.drop('support', axis=1)
#test_scores = [accuracy,
np.average(test_result['precision']),
np.average(test_result['recall']),
np.average(test_result['f1 score'])]
test_result
```

In [21]

```
#confusion matrix
test_cm =
pd.DataFrame(confusion_matrix(test_gen.classes,
y_pred_test))
test_cm.columns = multiindex_pred
test_cm.index = multiindex_actual
test_cm.index.name = 'actual'
test_cm
```

In [22]

```
#Menyimpan model
saved_model_dir = './aviva/model.h5'
model.save(saved_model_dir)
```

***Source* Kode Aplikasi Android :**

https://drive.google.com/drive/folders/1KuIov1fEbHOYfSCUre-trsaoRNG68ouX?usp=share_lin