LAMPIRAN

Lampiran 1 Source Code Akuisisi Data dan Preprocessing

```
import os
def list_files(startpath, folder_lvl):
    level_val = 0
    for root, dirs, files in os.walk(startpath):
        level = root.replace(startpath, '').count(os.sep)
        indent = ' ' * 4 * (level)
        strimages = str(len(files)) + ' images'
        if len(files) > 0:
            print('{}{}{}{}'.format(indent,        os.path.basename(root),
indent, strimages))
        else :
            print('{}{} '.format(indent, os.path.basename(root)))
        level_val += 1
```

```
#show data count
list_files('dataset_raw', 0)
```

```
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

w=256
h=256
fig=plt.figure(figsize=(16, 16))
columns = 5
rows = 4
count = 1
for i in os.listdir('dataset_raw/early'):
    if count >= 21:
        break
    img = Image.open('dataset_raw/early/' + str(i))
    img.thumbnail((h,w))
    fig.tight_layout(pad=3.0)
    fig.add_subplot(rows, columns, count)
    plt.imshow(img)
    count+=1
plt.show()
```

```
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

w=256
h=256
fig=plt.figure(figsize=(16, 16))
columns = 5
rows = 4
count = 1
for i in os.listdir('dataset_raw/healthy'):
    if count >= 21:
        break
    img = Image.open('dataset_raw/healthy/' + str(i))
    img.thumbnail((h,w))
    fig.tight_layout(pad=3.0)
    fig.add_subplot(rows, columns, count)
    plt.imshow(img)
    count+=1
plt.show()
```

```
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

w=256
h=256
fig=plt.figure(figsize=(16, 16))
```

```
columns = 5
rows = 4
count = 1
for i in os.listdir('dataset_raw/late'):
    if count >= 21:
        break
    img = Image.open('dataset_raw/late/' + str(i))
    img.thumbnail((h,w))
    fig.tight_layout(pad=3.0)
    fig.add_subplot(rows, columns, count)
    plt.imshow(img)
    count+=1
plt.show()
```

```
#creating folders
def create_folder(path):
    os.mkdir(path)
    print(path + ' has been created')
dataset = os.listdir('dataset_raw')
create_folder('dataset_split')
create_folder('dataset_split/train')
create_folder('dataset_split/validation')
create_folder('dataset_split/test')
for i in dataset:
    create_folder('dataset_split/train/' + i)
    create_folder('dataset_split/validation/' + i)
    create_folder('dataset_split/test/' + i)
```

```
list_files('dataset_split', 1)
```

```
import Augmentor
import os

def perbanyak_(ini, sebanyak_ini):
  source_dir = ini
  output_dir = "."
  p           =           Augmentor.Pipeline(source_directory=source_dir,
output_directory=output_dir)
  p.rotate(probability=1, max_left_rotation=20, max_right_rotation=20)

  p.sample(sebanyak_ini)

perbanyak_('dataset_aug/train/healthy', 600-92)
perbanyak_('dataset_aug/test/healthy', 200-30)
perbanyak_('dataset_aug/validation/healthy', 200-30)
```

```
list_files('dataset_aug', 1)
```

```
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

w=256
h=256
fig=plt.figure(figsize=(16, 16))
columns = 5
rows = 4
count = 1
for i in range(1,21):
    img = Image.open('dataset_aug/test/healthy/aug (' + str(i) + ').JPG')
    img.thumbnail((h,w))
    fig.tight_layout(pad=3.0)
    fig.add_subplot(rows, columns, count)
    plt.imshow(img)
    count+=1
plt.show()
```

Lampiran 2 Source Code Training CNN

```
# from keras.utils import plot_model
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, MaxPool2D, Flatten, Dense,
Activation, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.metrics import categorical_accuracy
from keras.optimizers import Adam

size_w = 128
size_h = 128

model = Sequential([
    Conv2D(8, kernel_size=7, activation='relu',
           input_shape=(size_w,size_h,3),
           padding = 'same'),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(16, kernel_size=7, activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(32, kernel_size=7, activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(64, kernel_size=7, activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Flatten(),
    Dense(1024, activation='relu'),
    Dense(512, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])
```

```
model.compile(optimizer = 'sgd', loss = 'categorical_crossentropy', metrics
= [categorical_accuracy])
model.summary()
```

```
train = ImageDataGenerator(rescale = 1./255)
validation = ImageDataGenerator(rescale = 1./255)
train_data = train.flow_from_directory('dataset_aug/train', target_size =
(size_w, size_h), batch_size = 32, class_mode = 'categorical')
validation_data = validation.flow_from_directory('dataset_aug/validation',
target_size = (size_w, size_h), batch_size = 32, class_mode = 'categorical')
```

```
epoch = []
acc = []
val_acc = []
loss = []
val_loss = []
```

```
for i in range(1, 51):
    print('Epoch ke-', i)
    history = model.fit_generator(train_data, steps_per_epoch = 100, epochs
= 1, validation_data = validation_data, validation_steps = 100)

    accuracy = history.history['categorical_accuracy']
    val_accuracy = history.history['val_categorical_accuracy']
    losse = history.history['loss']
    val_losse = history.history['val_loss']

    epoch.append(i)
    acc.append(accuracy[0])
    val_acc.append(val_accuracy[0])
    loss.append(losse[0])
    val_loss.append(val_losse[0])
    model.save('saved_model/7x7-4/7x7-4-Epoch-' + str(i) + '.h5')
```

```
import pandas as pd
logs = pd.DataFrame(
    {'epoch': epoch,
     'train_cc': acc,
     'val_acc': val_acc,
     'loss': loss,
     'val_loss' : val_loss
    })
```

```
logs.to_excel('saved_model/7x7-4/7x7-4-logs.xlsx')
print("Max Val Acc"  , str(max(val_acc)))
```

```
import matplotlib.pyplot as plt
%matplotlib inline
categorical_accuracy = acc
val_categorical_accuracy = val_acc
loss = loss
val_loss = val_loss
epochs = range(1, len(categorical_accuracy) + 1)
#Train and validation accuracy
plt.plot(epochs, categorical_accuracy, 'b', label='Training accurarcy')
plt.plot(epochs,    val_categorical_accuracy,    'r',    label='Validation
accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

```
i = val_acc.index(max(val_acc))
epoch[i]
print("Best Val Acc")
print("Epoch\tAcc\tVal_Acc\t\t\tLoss\t\t\tVal_Loss")
print(str(epoch[i]) + "\t" + str(acc[i]) + "\t" + str(val_acc[i]) + "\t" +
str(loss[i]) + "\t" + str(val_loss[i]) + "\t")
```

```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model
import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
import os
from sklearn.metrics import classification_report

nama_train_data = {0: 'early', 1: 'healthy', 2: 'late'}
def confusion_mat(model_loc):
    model = load_model(model_loc)


    benar = 0
    actual = []
    predicted = []
    for suatu in os.listdir('dataset_aug/test'):
        for a in os.listdir('dataset_aug/test/' + suatu + '/'):
            img = image.load_img('dataset_aug/test/' + suatu + '/' + a,
target_size = (128, 128))
            img = image.img_to_array(img)
            img = np.expand_dims(img, axis = 0)

            hasil = model.predict_classes(img)

            for num, kelas in nama_train_data.items():
                if num == hasil[0]:
                    actual.append(suatu)
                    predicted.append(kelas)

    for i in range(0,len(predicted)):
        if actual[i] == predicted[i]:
            benar+=1
    confusion = confusion_matrix(actual, predicted)
    akurasi = accuracy_score(actual, predicted)
    f1Score = f1_score(actual, predicted, average='macro')
    print('Accuracy: {:.4f}'.format(akurasi))
    print('F1Score: {:.4f}'.format(f1Score))
```

```
    plot_confusion_matrix(cm           = confusion,
                          normalize    = False,
                          target_names = nama_train_data.values(),
                          title        = "Confusion Matrix")
    print('\nClassification Report\n')
    print(classification_report(actual,                          predicted,
target_names=nama_train_data.values()))
```

```
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    """
    given a sklearn confusion matrix (cm), make a nice plot

    Arguments
    ---------
    cm:           confusion matrix from sklearn.metrics.confusion_matrix

    target_names: given classification classes such as [0, 1, 2]
                  the class names, for example: ['high', 'medium', 'low']

    title:        the text to display at the top of the matrix

    cmap:                 the  gradient  of  the  values  displayed  from
matplotlib.pyplot.cm
                  see
http://matplotlib.org/examples/color/colormaps_reference.html
                  plt.get_cmap('jet') or plt.cm.Blues

    normalize:    If False, plot the raw numbers
                  If True, plot the proportions

    Usage
    -----
    plot_confusion_matrix(cm           = cm,                  # confusion
matrix created by
                                                             #
sklearn.metrics.confusion_matrix
                          normalize     = True,                    # show
proportions
                          target_names = y_labels_vals,         # list of
names of the classes
                          title         = best_estimator_name) # title of
graph

    Citiation
    ---------
    http://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

    """
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
```

```
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

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


    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted                          label\naccuracy={:0.4f};
misclass={:0.4f}'.format(accuracy, misclass))
    plt.show()
```

```
print('7x7-4')
confusion_mat('saved_model/7x7-4/7x7-4-Epoch-' + str('50') + '.h5')
```

Lampiran 3 Source Code Plotting Grafik Hasil Training

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
arch = "7x7-4"
data = pd.read_excel('saved_model/'+ arch + '/' + arch + '-logs.xlsx')

data.head()
```

```
epoch = data['epoch'].values
train_acc = data['train_cc'].values
val_acc = data['val_acc'].values
loss = data['loss'].values
val_loss = data['val_loss'].values
```

```
fig, acc_graph = plt.subplots()

acc_graph.set_title('Training and Validation Accuracy - ' + arch)
acc_graph.plot(epoch, train_acc, 'b', label = 'Training Accuracy')
acc_graph.plot(epoch, val_acc, 'r', label = 'Validation Accuracy')
acc_graph.set_xlabel('Epoch')
acc_graph.set_ylabel('Accuracy')
acc_graph.legend()
```

```
fig, acc_graph = plt.subplots()

acc_graph.set_title('Loss and Validation Loss - ' + arch)
acc_graph.plot(epoch, loss, 'b', label = 'Loss')
acc_graph.plot(epoch, val_loss, 'r', label = 'Validation Loss')
acc_graph.set_xlabel('Epoch')
acc_graph.set_ylabel('Loss')
acc_graph.legend()
```