DAFTAR PUSTAKA

Alidrus, S. A., Aziz, M., & Putra, O. V. (2021). Deteksi Penyakit Pada Daun Tanaman Padi Menggunakan Metode Convolutional Neural Network. *Creative Information Technology Journal*, *8*(1), 22. https://doi.org/10.24076/citec.2021v8i1.263

AWS. (2019). Amazon Machine Learning. *Machine Learning in the AWS Cloud*, 317–351. https://doi.org/10.1002/9781119556749.ch15

Barman, R., Deshpande, S., Agarwal, S., & Inamdar, U. (2019). Transfer Learning for Small Dataset. *Proceedings of National Conference on Machine Learning*. https://www.researchgate.net/publication/333080572

BPS. (2020). Statistik Luas Panen dan Produksi Padi. *Berita Resmi Statistik*, 1–12. https://www.bps.go.id/pressrelease/2020/02/04/1752/

BPS. (2022). Luas Panen dan Produksi Padi di Indonesia 2021. *BPS*, *1999*(December), 1–6. https://www.bps.go.id/publication/2022/07/12/c52d5cebe530c363d0ea4198/luas-panen-dan-produksi-padi-di-indonesia-2021.html

El-Amir, H., & Hamdy, M. (2020). Deep Learning Pipeline. Dalam *Deep Learning Pipeline*. https://doi.org/10.1007/978-1-4842-5349-6

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2019). Fundamental concepts of convolutional neural network. *Intelligent Systems Reference Library*, *172*, 519–567. https://doi.org/10.1007/978-3-030-32644-9_36

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. Dalam *MIT Press*. MIT Press.

Groth, D., & Hollier, C. (2020). Leaf Smut of Rice. *The LSU AgCenter*, *3115*, 3115.

Hossain, S. M. M., Tanjil, M. M. M., Ali, M. A. bin, Islam, M. Z., Islam, M. S., Mobassirin, S., Sarker, I. H., & Islam, S. M. R. (2020). Rice Leaf Diseases Recognition Using Convolutional Neural Networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12447 LNAI*, 299–314. https://doi.org/10.1007/978-3-030-65390-3_23

IRRI. (2019). *Brown Spot*. IRRI; Knowledge Bank. http://www.knowledgebank.irri.org/training/fact-sheets/pest-management/diseases/item/brown-spot

Liu, J., & Wang, X. (2020). Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. *Plant Methods*, *16*(1), 1–16. https://doi.org/10.1186/s13007-020-00624-2

Moroney, L. (2021). AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence. Dalam *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9). O'Reilly.

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *Cornell University*, 1–11. https://doi.org/10.48550/arxiv.1511.08458

Premi, M. S. G., Narmadha, R., & Bernatin, T. (2019). A Brief Survey on Diseases of Paddy Plant. *Journal Pharmaceutical Sciences and Research*, *11*(7), 2739–2743.

Rahman, C. R., Arko, P. S., Ali, M. E., Khan, M. A. I., Apon, S. H., Nowrin, F., & Wasif, A. (2018). Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks. *Biosystems Engineering*, *194*, 112–120. https://doi.org/10.1016/j.biosystemseng.2020.03.020

Rajayogi, J. R., Manjunath, G., & Shobha, G. (2019). Indian Food Image Classification with Transfer Learning. *CSITSS 2019 - 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution, Proceedings*. https://doi.org/10.1109/CSITSS47250.2019.9031051

Sandler, M., & Andrew Howard. (2018). *MobileNetV2: The Next Generation of On-Device Computer Vision Networks*. Google AI Blog. https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.48550/arxiv.1801.04381

Singh, H. (2019). *Practical Machine Learning and Image Processing For Facial Recognition, Object Detection, and Pattern Recognition Using Python*. Apress Berkeley, CA. https://doi.org/https://doi.org/10.1007/978-1-4842-4149-3

Suyanto. (2018). *Machine Learning Tingkat Dasar dan Lanjut*. Informatika Bandung.

Zhou, G., Zhang, W., Chen, A., He, M., & Ma, X. (2019). Rapid Detection of Rice Disease Based on FCM-KM and Faster R-CNN Fusion. *IEEE Access*, *7*, 143190–143206. https://doi.org/10.1109/ACCESS.2019.2943454

LAMPIRAN

Berikut ini merupakan *Source Code* yang digunakan untuk mengimplementasikan arsitektur *MobileNetV2* dalam mendeteksi penyakit daun padi.

A. Visualisasi Data

```
# Mengakses Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Menginstall package tensorflow-addons
!pip install tensorflow-addons

# Import package
import os

# Visualisasi citra asli
directory = "/content/drive/MyDrive/rice_leaf_disease/Bacterial Leaf
Blight/DSC_0365.JPG"
img = mpimg.imread(directory)
imgplot = plt.imshow(img)
plt.show()

#Visualisasi citra yang diresize
img_resize = load_img(directory, grayscale=False, color_mode='rgb',
target_size=(224,224))
plt.figure(figsize=(3,3))
imgplot = plt.imshow(img_resize)

# Mengubah gambar menjadi array
img_arr = img_to_array(img_resize)

# Menampilkan array
img_arr

# Melakukan normalisasi
img_norm = img_arr/255.0

# Menampilkan array hasil normalisasi
img_norm
```

```
import pathlib

import numpy as np
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow_hub as hub
import tensorflow as tf
import keras
import tensorflow_addons as tfa

from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
from scipy import interp
from itertools import cycle

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc, roc_auc_score

# Direktori dataset
dir = '/content/drive/MyDrive/rice_leaf_disease'

# Nama-nama kelas pada dataset
File=[]
for file in os.listdir(dir):
    File+=[file]
print(File)

# Visualisasi citra asli
directory = "/content/drive/MyDrive/rice_leaf_disease/Bacterial Leaf
Blight/DSC_0365.JPG"
img = mpimg.imread(directory)
imgplot = plt.imshow(img)
plt.show()
```

```
#Visualisasi citra yang diresize
img_resize = load_img(directory, grayscale=False, color_mode='rgb',
target_size=(224,224))
plt.figure(figsize=(3,3))
imgplot = plt.imshow(img_resize)

# Mengubah gambar menjadi array
img_arr = img_to_array(img_resize)

# Menampilkan array
img_arr

# Melakukan normalisasi
img_norm = img_arr/255.0

# Menampilkan array hasil normalisasi
img_norm

# Direktori masing-masing kelas
# Directory with our Bacterial Leaf Blight pictures
train_bacterial_dir =
os.path.join('/content/drive/MyDrive/rice_leaf_disease/Bacterial
Leaf Blight')

# Directory with our Brown Spot pictures
train_brown_dir =
os.path.join('/content/drive/MyDrive/rice_leaf_disease/Brown Spot')

# Directory with our Leaf Smut pictures
train_leaf_dir =
os.path.join('/content/drive/MyDrive/rice_leaf_disease/Leaf Smut')

# Mengambil nama file masing-masing kelas
train_bacterial_names = os.listdir(train_bacterial_dir)
print("File names in Bacterial Leaf Blight directory:")
print(train_bacterial_names[:10])

train_brown_names = os.listdir(train_brown_dir)
print("\nFile names in Brown Spot directory:")
print(train_brown_names[:10])
```

```
train_leaf_names = os.listdir(train_leaf_dir)
print("\nFile names in Leaf Smut directory:")
print(train_leaf_names[:10])

# Menampilkan jumlah citra pada masing-masing kelas
print('total Bacterial Leaf Blight images:',
len(os.listdir(train_bacterial_dir)))
print('total Brown Spot images:', len(os.listdir(train_brown_dir)))
print('total Leaf Smut images:', len(os.listdir(train_leaf_dir)))

%matplotlib inline

# Parameters for our graph; we'll output images in a 3x3
configuration
nrows = 3
ncols = 3

# Index for iterating over images
pic_index = 0

# Menampilkan 9 sampel citra kelas Bacterial Leaf Blight
# Set up matplotlib fig, and size it to fit 3x3 pics
fig = plt.gcf()
fig.set_size_inches(16, 8)

pic_index += 9
next_bacterial_pix = [os.path.join(train_bacterial_dir, fname)
                for fname in train_bacterial_names[pic_index-
9:pic_index]]

for i, img_path in enumerate(next_bacterial_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)
  if (i + 1 == 2):
    plt.title("Bacterial Leaf Blight disease", pad = 40, fontsize =
20)
```

```
plt.show()

pic_index = 0

# Menampilkan 9 sampel citra kelas Brown Spot
# Set up matplotlib fig, and size it to fit 3x3 pics
fig = plt.gcf()
fig.set_size_inches(16, 8)

pic_index += 9
next_brown_pix = [os.path.join(train_brown_dir, fname)
                for fname in train_brown_names[pic_index-
9:pic_index]]

for i, img_path in enumerate(next_brown_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i+1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)
  if (i + 1 == 2):
    plt.title("Brown Spot disease", pad = 40, fontsize = 20)

plt.show()

pic_index = 0

# Menampilkan 9 sampel citra kelas Leaf Smut
# Set up matplotlib fig, and size it to fit 3x3 pics
fig = plt.gcf()
fig.set_size_inches(16, 8)

pic_index += 9
next_leaf_pix = [os.path.join(train_leaf_dir, fname)
                for fname in train_leaf_names[pic_index-
9:pic_index]]

for i, img_path in enumerate(next_leaf_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i+1)
```

```
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)
  if (i + 1 == 2):
    plt.title("Leaf Smut disease", pad = 40, fontsize = 20)

plt.show()
```

B. *Preprocessing* Data

```
# Preprocessing data citra
dataset = []
mapping = {'Bacterial_Leaf_Blight':0, 'Brown_Spot':1, 'Leaf_Smut':2}
count = 0

for file in os.listdir(dir):
    path = os.path.join(dir,file)
    for im in os.listdir(path):
        image = load_img(os.path.join(path,im), grayscale=False,
color_mode='rgb', target_size=(224,224))
        image = img_to_array(image)
        image = image/255.0
        dataset.append([image,count])
    count=count+1

# Mengubah data menjadi array
data,labels0=zip(*dataset)
labels1=to_categorical(labels0)
data=np.array(data)
labels=np.array(labels1)
print(data.shape)
print(labels.shape)

# Pembagian data training dan testing
train_x,test_x,train_y,test_y=train_test_split(data,labels,test_size
=0.2,random_state=13)

# Menampilkan bentuk dimensi data training dan testing
print(train_x.shape)
print(test_x.shape)
```

```
print(train_y.shape)
print(test_y.shape)

# Augmentasi data
datagen = ImageDataGenerator(
        horizontal_flip=True,
        vertical_flip=True,
        rotation_range=20,
        zoom_range=0.2,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.1,
        fill_mode="nearest")
```

C. Membangun dan melatih arsitektur *MobileNetV2*

```
# Mengambil model MobileNetV2 dari TensorFlow Hub
module_selection = ("mobilenet_v2", 224, 1280) #@param
["(\"mobilenet_v2\", 224, 1280)", "(\"inception_v3\", 299, 2048)"]
{type:"raw", allow-input: true}
handle_base, pixels, FV_SIZE = module_selection
MODULE_HANDLE ="https://tfhub.dev/google/tf2-
preview/{}/feature_vector/4".format(handle_base)
IMAGE_SIZE = (pixels, pixels)
print("Using {} with input size {} and output dimension
{}".format(MODULE_HANDLE, IMAGE_SIZE, FV_SIZE))

do_fine_tuning = False #@param {type:"boolean"}

# Membuat lapisan untuk MobileNetV2
feature_extractor = hub.KerasLayer(MODULE_HANDLE,
                                   input_shape=IMAGE_SIZE + (3,),
                                   output_shape=[FV_SIZE],
                                   trainable=do_fine_tuning,
                                   name="MobileNetV2",)

# Membangun model akhir
print("Building model with", MODULE_HANDLE)

model = tf.keras.Sequential([
        feature_extractor,
```

```
        tf.keras.layers.Dense(256, activation='relu',
name='Fully_Connected_Layer'),
        tf.keras.layers.Dropout(0.5, name='Dropout_Layer'),
        tf.keras.layers.Dense(3, activation='softmax',
name='Output_Layer')
])

model.summary()

# Membekukan lapisan agar bobot tidak berubah
NUM_LAYERS = 10 #@param {type:"slider", min:1, max:50, step:1}

if do_fine_tuning:
    feature_extractor.trainable = True

    for layer in model.layers[-NUM_LAYERS:]:
        layer.trainable = True

else:
    feature_extractor.trainable = False

# Compile model
METRICS = [
      'accuracy',
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      tfa.metrics.F1Score(num_classes=3)
]

if do_fine_tuning:
    model.compile(optimizer=tf.keras.optimizers.SGD(lr=0.002,
momentum=0.9),

loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                  metrics=METRICS)
else:
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=METRICS)

# plot the model including the sizes of the model
```

```
tf.keras.utils.plot_model(model, show_shapes=True)

# Train the model.
history = model.fit(
            datagen.flow(train_x, train_y, batch_size=8),
            validation_data = (test_x, test_y),
            epochs = 50)
```

D. Evaluasi kinerja model

```
# Menampilkan nilai presisi, recall, dan F1-score
pred = model.predict(test_x, verbose=2)
model_predicted = np.argmax(pred, axis = 1)

print(classification_report(test_y.argmax(axis=1), model_predicted,
target_names=File))

colors = plt.rcParams['axes.prop_cycle'].by_key()['color']

# Menampilkan kurva akurasi
acc      = history.history['accuracy']
val_acc  = history.history['val_accuracy']

epochs   = range(len(acc))

plt.plot(epochs, acc, color=colors[0], label='Train')
plt.plot(epochs, val_acc, color=colors[0], linestyle="--",
label='Val')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.show()

# Menampilkan kurva F1-score untuk masing-masing kelas
f1_score = history.history['f1_score']
val_f1_score = history.history['val_f1_score']
f1_numpy = np.array(f1_score)
val_f1_numpy = np.array(val_f1_score)
```

```
epochs   = range(len(f1_score))

n_classes = 3
colors = cycle(['orange', 'green', 'blue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(epochs, f1_numpy[:, i], color=color,
             label='Train F1-score of class {0}'.format(i))
for i, color in zip(range(n_classes), colors):
    plt.plot(epochs, val_f1_numpy[:, i], color=color,
             linestyle="--",
             label='Val F1-score of class {0}'.format(i))

plt.xlabel('Epoch')
plt.ylabel('F1-score')
plt.title('Training and validation f1-score')
plt.legend(loc=0)
plt.figure(figsize=(12,12))
plt.show()

# Menampilkan Confusion Matrix
disease_types = ["Bacterial Leaf Blight", "Brown Spot", "Leaf Smut"]
Y_pred = model.predict(test_x)

Y_pred = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(test_y, axis=1)

fig, ax = plt.subplots(figsize=(7,7))
cm = confusion_matrix(Y_true, Y_pred)
#plt.figure(figsize=(12, 12))
ax = sns.heatmap(cm, cmap=plt.cm.Blues, annot=True,
                 square=True, xticklabels=disease_types,
                 yticklabels=disease_types, ax=ax)
ax.set_ylabel('Actual', fontsize=15)
ax.set_xlabel('Predicted', fontsize=15)
ax.set_title('Confusion Matrix', fontsize=20, pad=20)

# Menampilkan kurva AUC-ROC
Y_pred = model.predict(test_x)

n_classes = 3
```

```
lw = 2
Y_val = test_y
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(Y_val[:, i], Y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(Y_val.ravel(),
Y_pred.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# Pertama-tama, gabungkan semua False Positive Rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in
range(n_classes)]))

# Kemudian interpolasi semua kurva ROC pada titik ini
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Terakhir, rata-rata dan hitung AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot semua kurva ROC
plt.figure(figsize=(8, 8))
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)
```

```
colors = cycle(['orange', 'green', 'blue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to
multi-class')
plt.legend(loc="lower right")
plt.show()
```

E. *Convert* model *for deployment*

```
# Save model
export_dir = 'content/saved_model/1'
tf.saved_model.save(model, export_dir)

# Convert the model.
converter = tf.lite.TFLiteConverter.from_saved_model(export_dir)
tflite_model = converter.convert()

# Membuat model berformat .tflite
tflite_model_file = '/content/model.tflite'

with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```

***Source code* android application**

```
https://github.com/maximilian179/PadiKu
```