DAFTAR PUSTAKA

Adhikari, P., Oh, Y., & Panthee, D. R. (2017). Current status of early blight resistance in tomato: An update. *International Journal of Molecular Sciences*, 18(10). doi: 10.3390/ijms18102019.

Ali, M. M., Bachik, N. A., Muhadi, N. A., & Yusof, T. N. T. (2019). Non-destructive techniques of detecting plant diseases: A review. *Physiological and Molecular Plant Pathology. Elsevier Ltd*, 108, p. 101426. doi: 10.1016/j.pmpp.2019.101426.

Alim, M. M. F. (2020). Identifikasi Penyakit Tanaman Tomat Menggunakan Algoritma Convolutional Neural Network Dan Pendekatan Transfer Learning. *Fakultas Teknik Universitas Negeri Semarang*.

Albelwi, S., & Mahmood, A. (2017). A Framework for Designing the Architectures of Deep Convolutional Neural Networks. *Entropy*, 19(6).

Abas, M. A. H., Ismail, N., Yassin, A. I. M., & Taib, M. N. (2018). VGG16 for Plant Image Classification with Transfer Learning and Data Augmentation. *International Journal of Engineering & Technology*, 7(4).

Astiningrum, M., Arhandi, P. P., & Ariditya, N. A. (2019) Identifikasi Penyakit pada Daun Tomat Berdasarkan Fitur Warna dan Tekstur. *Jurusan Teknologi Informasi Politeknik Negeri Malang*, pp. 1-4.

BPS. (2020). Produksi dan Luas Panen Tomat. *Perencanaan Pembangunan*, https://www.bps.go.id/

Bahuleyan, H. (2018). Music Genre Classification using Machine Learning Techniques. https://arxiv.org/pdf/1804.01149v1

Chen, W., Sun, Q., Wang, J., Dong, J. J., & Xu, C. (2018). A Novel Model Based on AdaBoost and Deep CNN for Vehicle Classification. *IEEE Access*, *6*, 60445–60455. https://doi.org/10.1109/ACCESS.2018.2875525

Felix, Faisal, S., Butarbutar, T. F. M., & Sirait, P. (2019). Implementasi CNN dan SVM untuk Identifikasi Penyakit Tomat via Daun. *Jurnal SIFO Mikroskil*, 20(2).

Hardiono, D. P., Taha, F. P., & Wahyuni, S. (2020). Learning Tomato-Leaf Dataset using Inception-V3 Architecture with Phyton. *Academia*.

Kankolongo, A. M. (2018). Food Crop Production by Smallholder Farmers in Southern Africa. *Afrika : Academic Press*.

Martinelli, F., dkk. (2015). Advanced methods of plant disease detection. A review. *Agronomy for Sustainable Development*, 35(1), pp. 1–25. doi: 10.1007/s13593-014-0246-1.

Mendes, D. B., & Silva, N. C. (2018). Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images. https://arxiv.org/pdf/1812.02316

Nurmasani, A. & Pristyanto, Y. (2021). Algoritme Stacking Untuk Klasifikasi Penyakit Jantung Pada Dataset Imbalanced Class. *Jurnal Pseudocode*, 8(1).

Osdaghi, E., dkk. (2017). Monitoring the occurrence of tomato bacterial spot and range of the causal agent Xanthomonas perforans in Iran. *Plant Pathology*, 66(6).

Paliwang, A. A. A., Septian, M. R. D., Cahyanti, M., & Swedia, E. R. (2020). Klasifikasi Penyakit Tanaman Apel Dari Citra Daun Dengan Convolutional Neural Network. *SEBATIK*, 24(2).

Pristyanto, Y., Sidauruk, A., & Nurmasani, A. (2022). Klasifikasi Penyakit Diabetes Pada Imbalanced Class Dataset Menggunakan Algoritme Stacking. *Jurnal Media Informatika Budidarma*, 6(1).

Rakhmawati, P. U., Pranoto, Y. M., & Setyati, E. (2018). Klasifikasi Penyakit Daun Kentang Berdasarkan Fitur Tekstur Dan Fitur Warnamenggunakan Support Vector Machine. *Seminar Nasional Teknologi dan Rekayasa (SENTRA)*, pp. 1-8.

Sita, B. R., & Hadi, S. (2016). Produktivitas Dan Faktor-Faktor Yang Berpengaruh Terhadap Produksi Usahatani Tomat (Solanum Lycopersicum Mill) Di Kabupaten Jember. *JSEP*, 9(3).

Sari, D. F., & Swanjaya, D. (2020). Implementasi Convolutional Neural Network Untuk Identifikasi Penyakit Daun Gambas. *Seminar Nasional Inovasi Teknologi*, 4(3).

Satria, D. A. (2011). Biologi Tungau Laba-laba Merah Tetranychus urticae Koch (Acari: Tetranychidae) pada Kacang Merah dan Kacang Hijau. *Fakultas Pertanian Universitas Brawijaya.*

Sanoubar, R., & Barbanti, L. (2017). Fungal diseases on tomato plant under greenhouse condition. *European Journal of Biological Research*, 7(4).

Setiawan, W. (2019). Perbandingan Arsitektur Convolutional Neural Network Untuk Klasifikasi Fundus. *Jurnal SimanteC*, 7(2).

Srinivasan, K., dkk. (2021). Performance Comparison of Deep CNN Models for Detecting Driver's Distraction. *Computers, Materials & Continua*, 68(3).

Sidik, D. D. (2019). Penggunaan Stacking Classifier Untuk Prediksi Curah Hujan. *IT FOR SOCIETY*, 4(1).

Thiodorus, G., Prasetia, A., Ardhani, L. A., & Yudistira, N. (2021). Klasifikasi Citra Makanan/Nonmakanan Menggunakan Metode Transfer Learning Dengan Model Residual Network. *Jurnal Ilmiah Sistem Informasi*, 11(2).

LAMPIRAN

Lampiran 1 Source Code

A. Model Transfer Learning Arsitektur InceptionV3, Xception, dan VGG16

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
from os import listdir

import keras
from keras.preprocessing import image
from keras import backend as K
from keras.layers import Input

from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split

import tensorflow as tf
import numpy as np
import cv2
import os
from tensorflow.keras.applications import InceptionV3, DenseN
et121, ResNet50, Xception, VGG16, MobileNet, InceptionResNetV
2
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGen
erator

dir_root = "/content/drive/My Drive/Dataset/tomato/"

# Show the image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
img = mpimg.imread('/content/drive/My Drive/Dataset/tomato/train/Tomato___healthy/84cbb98b-5c2f-4367-8d78-52be491e66bd___GH_HL Leaf 336.JPG')
imgplot = plt.imshow(img)
plt.show()

def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

default_image_size = tuple((299, 299))
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(dir_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{dir_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)

        for plant_disease_folder in plant_disease_folder_list :
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{dir_root}/{plant_folder}/{plant_disease_folder}/")
```

```
            for single_plant_disease_image in plant_disease_i
mage_list :
                if single_plant_disease_image == ".DS_Store"
:
                    plant_disease_image_list.remove(single_pl
ant_disease_image)

            for image in plant_disease_image_list[:100]:
                image_directory = f"{dir_root}/{plant_folder}
/{plant_disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True o
r image_directory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(
image_directory))
                    label_list.append(plant_disease_folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

len(image_list)

label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
n_classes = len(label_binarizer.classes_)

np_image_list = np.array(image_list, dtype=np.float16) / 255.
0

x_train, x_test, y_train, y_test = train_test_split(
    np_image_list,
    image_labels,
    test_size=0.2,
    random_state = 42)

len(x_train), len(x_test), len(y_train), len(y_test)

aug = ImageDataGenerator(
    rotation_range=40,
    horizontal_flip=True,
    shear_range = 0.2,
    fill_mode = 'nearest',
    zoom_range = 0.2)
```

```
inceptionv3_model = InceptionV3(input_tensor=Input(shape = (2
99, 299, 3)), include_top = False, weights = 'imagenet')
x = inceptionv3_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation= 'relu')(x)
x = layers.Dense(n_classes, activation= 'softmax')(x)
modelInceptionV3 = Model(inceptionv3_model.input, x)

xception_model = Xception(input_tensor=Input(shape = (299, 29
9, 3)), include_top = False, weights = 'imagenet')
x = xception_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation= 'relu')(x)
x = layers.Dense(n_classes, activation= 'softmax')(x)
modelXception = Model(xception_model.input, x)

vgg16_model = VGG16(input_tensor=Input(shape = (299, 299, 3))
, include_top = False, weights = 'imagenet')
x = vgg16_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation= 'relu')(x)
x = layers.Dense(n_classes, activation= 'softmax')(x)
modelVGG16 = Model(vgg16_model.input, x)

modelInceptionV3.summary()
modelXception.summary()
modelVGG16.summary()

modelInceptionV3.compile (optimizer = Adam(learning_rate=0.00
01),
                loss = tf.keras.losses.CategoricalCrossentropy
(),
                metrics = ['accuracy'])

modelXception.compile (optimizer = Adam(learning_rate=0.0001)
,
                loss = tf.keras.losses.CategoricalCrossentropy
(),
                metrics = ['accuracy'])

modelVGG16.compile (optimizer = Adam(learning_rate=0.0001),
                loss = tf.keras.losses.CategoricalCrossentropy
(),
                metrics = ['accuracy'])
```

```
# InceptionV3 model training
procces = modelInceptionV3.fit(
    aug.flow(x_train, y_train, batch_size=32),
    validation_data=(x_test,y_test),
    steps_per_epoch=len(x_train) // 32,
    epochs=20,
    verbose=1)

scores = modelInceptionV3.evaluate(x_test, y_test, batch_size
=32)
print(f"Test Accuracy: {scores[1]*100}")

import matplotlib.pyplot as plt

accuracy = procces.history['accuracy']
val_accuracy = procces.history['val_accuracy']
loss = procces.history['loss']
val_loss = procces.history['val_loss']

plt.plot(accuracy)
plt.plot(val_accuracy)
plt.title('Accuracy Function')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train', 'val'])
plt.show()

plt.plot(loss)
plt.plot(val_loss)
plt.title('Loss Function')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.show()

pred = modelInceptionV3.predict(x_test, batch_size=32, verbos
e=2)
model_predicted = np.argmax(pred, axis = 1)

labels = os.listdir("/content/drive/My Drive/Dataset/tomato/t
rain")
labels
```

```
# Xception
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), model_pred
icted, target_names=labels))

# Xception
from sklearn.metrics import confusion_matrix
import seaborn as sn
import pandas as pd

model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_
predicted)
#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) #
 font size
plt.show()

from sklearn.metrics import roc_curve, auc

plt.figure(1, figsize=(10, 10))
plt.plot([0, 1], [0, 1], 'k--')
for i in range(len(label_binarizer.classes_)):
   fpr, tpr, thresholds = roc_curve(y_test[:, i], pred[:, i])
   individual_auc = auc(fpr, tpr)
   plt.plot(fpr, tpr, label= (label_binarizer.classes_[i] + '
(area = {})'.format(individual_auc)))

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()


# Xception
procces = modelXception.fit(
    aug.flow(x_train, y_train, batch_size=32),
    validation_data=(x_test,y_test),
    steps_per_epoch=len(x_train) // 32,
    epochs=20,
    verbose=1)
```

```
scores = modelXception.evaluate(x_test, y_test, batch_size=32
)
print(f"Test Accuracy: {scores[1]*100}")

import matplotlib.pyplot as plt

accuracy = procces.history['accuracy']
val_accuracy = procces.history['val_accuracy']
loss = procces.history['loss']
val_loss = procces.history['val_loss']

plt.plot(accuracy)
plt.plot(val_accuracy)
plt.title('Accuracy Function')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train', 'val'])
plt.show()

plt.plot(loss)
plt.plot(val_loss)
plt.title('Loss Function')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.show()

pred = modelXception.predict(x_test, batch_size=32, verbose=2
)
model_predicted = np.argmax(pred, axis = 1)

from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), model_pred
icted, target_names=labels))

model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_
predicted)
#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) #
 font size
```

```
plt.show()

from sklearn.metrics import roc_curve, auc

plt.figure(1, figsize=(10, 10))
plt.plot([0, 1], [0, 1], 'k--')
for i in range(len(label_binarizer.classes_)):
   fpr, tpr, thresholds = roc_curve(y_test[:, i], pred[:, i])
   individual_auc = auc(fpr, tpr)
   plt.plot(fpr, tpr, label= (label_binarizer.classes_[i] + '
(area = {})'.format(individual_auc)))

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()


# VGG16
procces = model.fit(
    aug.flow(x_train, y_train, batch_size=32),
    validation_data=(x_test,y_test),
    steps_per_epoch=len(x_train) // 32,
    epochs=20,
    verbose=1)

scores = modelVGG16.evaluate(x_test, y_test, batch_size=32)
print(f"Test Accuracy: {scores[1]*100}")

import matplotlib.pyplot as plt

accuracy = procces.history['accuracy']
val_accuracy = procces.history['val_accuracy']
loss = procces.history['loss']
val_loss = procces.history['val_loss']

plt.plot(accuracy)
plt.plot(val_accuracy)
plt.title('Accuracy Function')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train', 'val'])
plt.show()
```

```
plt.plot(loss)
plt.plot(val_loss)
plt.title('Loss Function')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.show()

pred = modelVGG16.predict(x_test, batch_size=32, verbose=2)
model_predicted = np.argmax(pred, axis = 1)

from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), model_pred
icted, target_names=labels))

model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_
predicted)
#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) #
 font size
plt.show()

from sklearn.metrics import roc_curve, auc

plt.figure(1, figsize=(10, 10))
plt.plot([0, 1], [0, 1], 'k--')
for i in range(len(label_binarizer.classes_)):
   fpr, tpr, thresholds = roc_curve(y_test[:, i], pred[:, i])
   individual_auc = auc(fpr, tpr)
   plt.plot(fpr, tpr, label= (label_binarizer.classes_[i] + '
(area = {})'.format(individual_auc)))

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()

# Save model
```

```
modelInceptionV3.save('/content/drive/My Drive/Model/inceptionv3.h5')
modelXception.save('/content/drive/My Drive/Model/xception.h5')
modelVGG16.save('/content/drive/My Drive/Model/vgg16.h5')
```

B. Ensembled Stacking Model Transfer Learning

```
from keras.models import load_model
from sklearn.metrics import accuracy_score

model1 = load_model('/content/drive/My Drive/Model/xception_tomato.h5')
model2 = load_model('/content/drive/My Drive/Model/inceptionv3_tomato.h5')
model3 = load_model('/content/drive/My Drive/Model/vgg16_tomato.h5')

models = [model1, model2, model3]

model1._name = "inception"
model2._name = "xception"
model3._name = "vgg16"

for layer in model1.layers:
  layer.trainable = False
  layer._name = layer._name + str("_inception")

for layer in model2.layers:
  layer.trainable = False
  layer._name = layer._name + str("_xception")

for layer in model3.layers:
  layer.trainable = False
  layer._name = layer._name + str("_vgg16")

model_input = tf.keras.Input(shape=(299, 299, 3))
model_outputs = [model(model_input) for model in models]
combinedOutput = layers.concatenate(model_outputs)
x = layers.Dense(512, activation="relu")(combinedOutput)
x = layers.Dense(n_classes, activation="softmax")(x)
ensemble_model = tf.keras.Model(inputs=model_input, outputs=x)
```

```
print(ensemble_model.summary())

ensemble_model.compile (optimizer = Adam(learning_rate=0.0001
),
               loss = tf.keras.losses.CategoricalCrossentropy
(),
               metrics = ['accuracy'])

procces = ensemble_model.fit(
    x_train, y_train,
    validation_data=(x_test,y_test),
    steps_per_epoch=len(x_train) // 32,
    epochs=20,
    verbose=1)

scores = ensemble_model.evaluate(x_test, y_test, batch_size=3
2)
print(f"Test Accuracy: {scores[1]*100}")

acc = procces.history['accuracy']
val_acc = procces.history['val_accuracy']
loss = procces.history['loss']
val_loss = procces.history['val_loss']
epochs = range(1, len(acc) + 1)

#Train and validation accuracy
plt.plot(epochs, acc, 'coral', label='Training accurarcy')
plt.plot(epochs, val_acc, 'gold', label='Validation accurarcy
')
plt.title('Training and Validation accurarcy')
plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'coral', label='Training loss')
plt.plot(epochs, val_loss, 'gold', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

pred = ensemble_model.predict(x_test, batch_size=32, verbose=
2)
model_predicted = np.argmax(pred, axis = 1)

labels = label_binarizer.classes_
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), model_pred
icted, target_names=labels))

from sklearn.metrics import confusion_matrix
import seaborn as sn
import pandas as pd

model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_
predicted)
#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) #
 font size
plt.show()

from sklearn.metrics import roc_curve, auc

plt.figure(1, figsize=(10, 10))
plt.plot([0, 1], [0, 1], 'k--')
for i in range(len(label_binarizer.classes_)):
   fpr, tpr, thresholds = roc_curve(y_test[:, i], pred[:, i])
   individual_auc = auc(fpr, tpr)
   plt.plot(fpr, tpr, label= (label_binarizer.classes_[i] + '
(area = {})'.format(individual_auc)))

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()

from keras.utils.vis_utils import plot_model
plot_model(ensemble_model, to_file='model.png', show_shapes=T
rue, show_layer_names=True)

# Save model .h5
ensemble_model.save('/content/drive/My Drive/Model/ensemble_m
odel.h5')

# convert model to tflite
```

```
from tensorflow import lite
converter = lite.TFLiteConverter.from_keras_model(ensemble_mo
del)
tfmodel = converter.convert()
open('model_fix.tflite', 'wb').write(tfmodel)
```