# DAFTAR PUSTAKA

Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).

Ajmera, R. (2022, Maret 15). Is Grape Juice Healthy? Sugar Content and More. Healthline Media. https://www.healthline.com/nutrition/is-grape-juice-good-for-you

Al-Yousuf, N., & Kamal-Eldin, A. (2018). The Protective Role of Grape-Derived Polyphenols against Alzheimer's Disease. Journal of Nutrition and Metabolism, 2018.

Amiruddin, A. (2017). Prospek Komoditas Anggur di Indonesia. Pusat Kajian dan Pengembangan Ekonomi Pertanian.

Arthana, R. (2019, April 5). Mengenal Accuracy, Precision, Recall dan Specificity serta yang diprioritaskan dalam Machine Learning. Medium. https://rey1024.medium.com/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8

Chiva-Blanch, G., & Badimon, L. (2017). Effects of Polyphenol Intake on Metabolic Syndrome: Current Evidences from Human Trials. Oxidative Medicine and Cellular Longevity, 2017, 5812401.

Fajri, L.R.H.A. (2022, Januari 3). Artificial Neural Network. Universitas STEKOM (Sains & Teknologi Komputer). http://sistem-informasi-s1.stekom.ac.id/informasi/baca/Artificial-Neural-Network/b1c26e9347ef547ff06845ca38cc443aedc4fa86

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE conference on computer vision and pattern recognition.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Mgbonyebi, O. P., Russo, J., & Russo, I. H. (2006). Antiproliferative effect of synthetic resveratrol on human breast epithelial cells. International Journal of Oncology, 28(4), 837-844.

Petruzzello, M. (2023, Februari 24). Grape. Encyclopedia Britannica. https://www.britannica.com/plant/grape

Puspitarini, I. D. A. R., Sari, R. P., & Wardhani, E. K. (2021). Potensi Anggur sebagai Produk Unggulan Agrowisata Indonesia. Jurnal Agribisnis Indonesia, 9(2), 121-130.

Ridhovan, A., Suharso, A. (2022). Penerapan Metode Residual Network (ResNet) dalam Klasifikasi Penyakit pada Daun Gandum. JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika, 7(1), 58-65.

Shafkat, I. (2018, Juni 2). Intuitively Understanding Convolutions for Deep Learning. Towards Data Science. https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Sholihah, N. (2018, Juni 25). Fully-Connected Layer CNN dan Implementasinya. Machine Learning MIPA Universitas Gadjah Mada. https://machinelearning.mipa.ugm.ac.id/2018/06/25/fully-connected-layer-cnn-dan-implementasinya

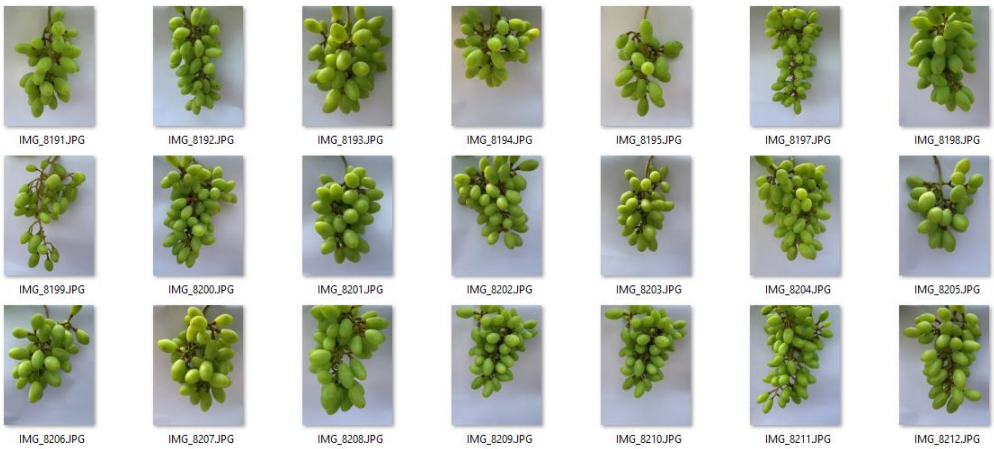
Yulianti, R., Hanifah, N., & Lusiana, N. (2019). Prospek Perdagangan Anggur di Pasar Internasional. Jurnal Bisnis dan Manajemen, 10(2), 112-125.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2014). Learning deep features for discriminative localization. arXiv preprint arXiv:1512.04150.

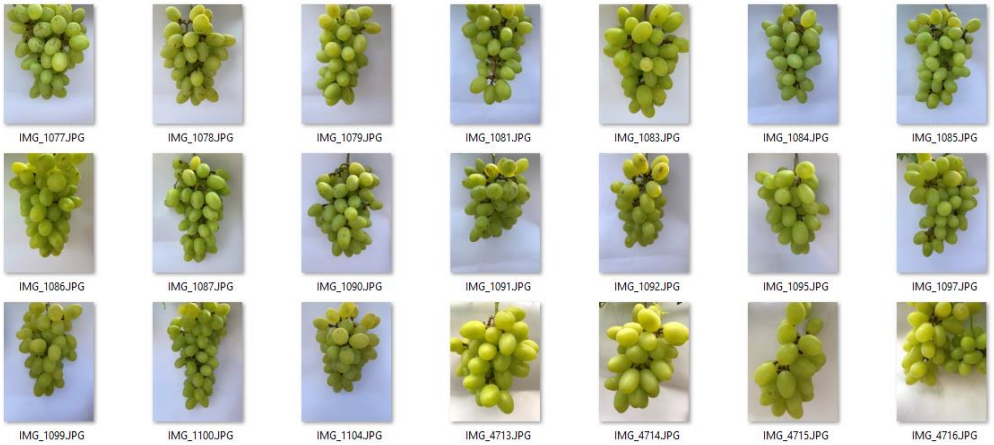
# LAMPIRAN

Lampiran 1 Contoh *Dataset*

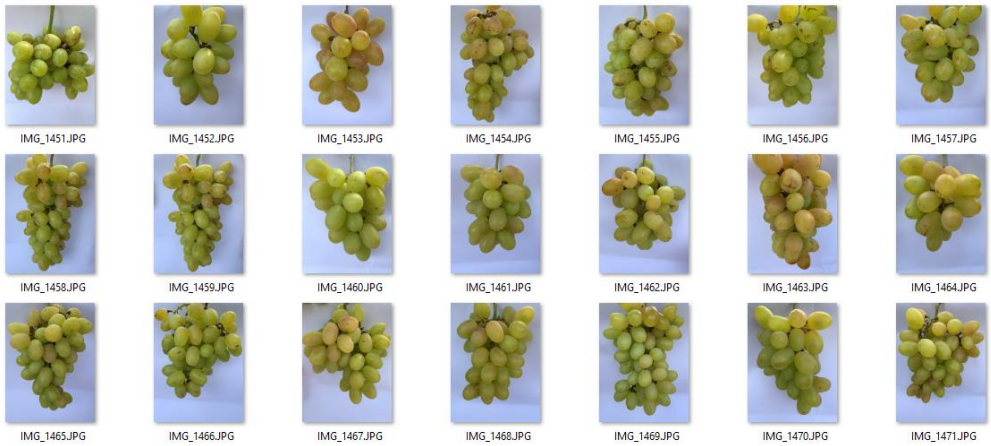
- Kelas Mentah




- Kelas Mengkal




- Kelas Matang



Lampiran 2 *Source Code* Program

a. *Resize* gambar

```
# Library yang dibutuhkan
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
from pathlib import Path
import argparse
import numpy

# Deklarasi variabel ArgumentParser
ap = argparse.ArgumentParser()

ap.add_argument("-i", "--image",
                required=True,
                help="Path to folder")

args = vars(ap.parse_args())

# Membaca image yang ada pada folder image yang disediakan
mypath = args["image"]
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath,
f))]
images = numpy.empty(len(onlyfiles), dtype=object)

# Iterasi pada setiap image
for n in range(0, len(onlyfiles)):

    path = join(mypath, onlyfiles[n])
    images[n] = cv2.imread(join(mypath, onlyfiles[n]),
                           cv2.IMREAD_UNCHANGED)

    # Proses Load image pada variabel img
    img = cv2.imread(path, 1)

    # Menentukan skala resize
    resize_scaling = 30
    resize_width = int(img.shape[1] * resize_scaling/100)
    resize_height = int(img.shape[0] * resize_scaling/100)
    resized_dimensions = (resize_width, resize_height)

    # Membuat image baru menggunakan dimensi yang sudah
ditentukan
    resized_image = cv2.resize(img, resized_dimensions,
                               interpolation=cv2.INTER_AREA)

    # Menyimpan image hasil resize di output folder
```

```
    cv2.imwrite(
    'output/' + str(resize_width) + str(resize_height) + str(n) +
'_resized.jpg', resized_image)

print("Images resized Successfully")
```

b. *Import Library*

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatt
en, BatchNormalization, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentrop
y
from tensorflow.keras.preprocessing.image import ImageDataGe
nerator
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
import os
```

c. *Import Dataset*

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

```
data_dir = '/content/drive/MyDrive/Skripsi/dataset2'
os.listdir(data_dir)
```

```
data = tf.keras.utils.image_dataset_from_directory('/content
/drive/MyDrive/Skripsi/dataset2')
```

```
print('Total anggur Matang :', len(os.listdir('/content/driv
e/MyDrive/Skripsi/dataset2/matang')))
print('Total anggur Mengkal :', len(os.listdir('/content/dri
ve/MyDrive/Skripsi/dataset2/mengkal')))
print('Total anggur Mentah :', len(os.listdir('/content/driv
e/MyDrive/Skripsi/dataset2/mentah')))
```

d. *Preview* sampel *dataset*

```
fig, ax = plt.subplots(ncols=5, figsize=(20,20))
for idx, img in enumerate(batch[0][:5]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```

e. *Build model* menggunakan *ResNet152v2*

```
import tensorflow as tf
from tensorflow.keras.layers import Input
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import ResNet152V2
model = tf.keras.models.Sequential([
    ResNet152V2(weights="imagenet", include_top=False, input
_tensor=Input(shape=(256, 256, 3))),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
model.layers[0].trainable = False
```

```
model.compile(optimizer=Adam(learning_rate=0.01), loss='spar
se_categorical_crossentropy', metrics=['accuracy'])
```

f. Proses *training*

```
logdir='logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_di
r=logdir)
hist = model.fit(train, validation_data=val, epochs=20, call
backs=[tensorboard_callback])
```

g. *Plotting accuracy* dan *loss*

```
fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='va
l_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

```
fig = plt.figure()
plt.plot(hist.history['accuracy'], color='teal', label='accu
racy')
plt.plot(hist.history['val_accuracy'], color='orange', label
='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

h. *Testing* model

```
img = cv2.imread('/content/drive/MyDrive/Skripsi/tes2/matang
/matang20.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()

resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()

yhat = model.predict(np.expand_dims(resize/255, 0))
yhat
```

```
img = cv2.imread('/content/drive/MyDrive/Skripsi/tes2/mengka
l/mengkal20.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()

resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()

yhat = model.predict(np.expand_dims(resize/255, 0))
yhat
```

```
img = cv2.imread('/content/drive/MyDrive/Skripsi/tes2/mentah
/mentah20.jpg')

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()

resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()

yhat = model.predict(np.expand_dims(resize/255, 0))
yhat
```

i. Implementasi model ke Website

```
from flask import Flask, render_template, request, jsonify
from keras.models import load_model

from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras_applications.mobilenet_v2 import preprocess_input
from keras_applications.mobilenet_v2 import decode_predictions
from keras_applications.mobilenet_v2 import MobileNetV2
```

```
import tensorflow as tf
from tensorflow import keras
from skimage import transform, io
import numpy as np
import os
from PIL import Image
from datetime import datetime
from flask_cors import CORS


app = Flask(__name__)

model = load_model("model.h5")


UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'tiff',
'webp', 'jfif'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower()
in ALLOWED_EXTENSIONS

# routes
@app.route("/", methods=['GET', 'POST'])
def main():
    return render_template("cnn.html")

@app.route("/classification", methods = ['GET', 'POST'])
def classification():
    return render_template("classifications.html")

@app.route('/submit', methods=['POST'])
def predict():
    if 'file' not in request.files:
        resp = jsonify({'message': 'No image in the request'})
        resp.status_code = 400
        return resp
    files = request.files.getlist('file')
    filename = "temp_image.png"
    errors = {}
    success = False
    for file in files:
        if file and allowed_file(file.filename):
            file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename))
            success = True
        else:
```

```
            errors["message"] = 'File type of {} is not
allowed'.format(file.filename)

    if not success:
        resp = jsonify(errors)
        resp.status_code = 400
        return resp
    img_url = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    # convert image to RGB
    img = Image.open(img_url).convert('RGB')
    now = datetime.now()
    predict_image_path = 'static/uploads/' +
now.strftime("%d%m%y-%H%M%S") + ".png"
    image_predict = predict_image_path
    img.convert('RGB').save(image_predict, format="png")
    img.close()

    # prepare image for prediction
    img = image.load_img(predict_image_path, target_size=(256,
256, 3))
    x = image.img_to_array(img)
    x = x/127.5-1
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])

    # predict
    prediction_array_model = model.predict(images)

    # prepare api response
    class_names = ['Matang', 'Mengkal', 'Mentah']
    # def argmedian(x):
    #      return np.argpartition(x, len(x) // 2)[len(x) // 2]
    sorted_indexes = np.argsort(prediction_array_model[0])
    median_index = sorted_indexes[len(sorted_indexes) // 2]
    # result = {
    #     "filename" : predict_image_path,
    #     "prediction": class_names[np.argmax(prediction_array)],
    #     "confidence": '{:2.0f}%'.format(100 *
np.max(prediction_array))
    # }

    return render_template("classifications.html", img_path =
predict_image_path,
                           predictionmodelmax =
class_names[np.argmax(prediction_array_model)],
                           predictionmodelmin =
class_names[np.argmin(prediction_array_model)],
```

```
                        predictionmodelmid =
class_names[median_index],
                        confidencemax = '{:2.0f}%'.format(100 *
np.max(prediction_array_model)),
                        confidencemedian = '{:2.0f}%'.format(100
* np.median(prediction_array_model)),
                        confidencemin = '{:2.0f}%'.format(100 *
np.min(prediction_array_model))
                        )

if __name__ =='__main__':
    #app.debug = True
    app.run(debug = True)
```

Lampiran 3 Hasil proses *Training* dan *Validasi*

```
Epoch 1/20
15/15 [==============================] - 22s 830ms/step -
loss: 11.1560 - accuracy: 0.6958 - val_loss: 0.9435 -
val_accuracy: 0.8854

Epoch 2/20
15/15 [==============================] - 11s 665ms/step -
loss: 1.0316 - accuracy: 0.8292 - val_loss: 0.8684 -
val_accuracy: 0.8333

Epoch 3/20
15/15 [==============================] - 9s 542ms/step - loss:
0.6483 - accuracy: 0.8021 - val_loss: 0.3383 - val_accuracy:
0.8542

Epoch 4/20
15/15 [==============================] - 10s 633ms/step -
loss: 0.3497 - accuracy: 0.8313 - val_loss: 0.4236 -
val_accuracy: 0.8854

Epoch 5/20
15/15 [==============================] - 9s 526ms/step - loss:
0.3689 - accuracy: 0.8750 - val_loss: 0.2676 - val_accuracy:
0.8646

Epoch 6/20
15/15 [==============================] - 9s 533ms/step - loss:
0.3579 - accuracy: 0.8625 - val_loss: 0.3265 - val_accuracy:
0.8854

Epoch 7/20
15/15 [==============================] - 10s 610ms/step -
loss: 0.5303 - accuracy: 0.8771 - val_loss: 0.3703 -
val_accuracy: 0.8750

Epoch 8/20
15/15 [==============================] - 9s 527ms/step - loss:
0.6304 - accuracy: 0.8833 - val_loss: 0.3854 - val_accuracy:
0.8854

Epoch 9/20
15/15 [==============================] - 10s 533ms/step -
loss: 0.9273 - accuracy: 0.9187 - val_loss: 0.2383 -
val_accuracy: 0.9375

Epoch 10/20
15/15 [==============================] - 11s 673ms/step -
loss: 0.8028 - accuracy: 0.9104 - val_loss: 0.3480 -
val_accuracy: 0.8333

Epoch 11/20
15/15 [==============================] - 11s 668ms/step -
loss: 0.3977 - accuracy: 0.8646 - val_loss: 0.4058 -
val_accuracy: 0.8750
```

```
Epoch 12/20
15/15 [==============================] - 12s 674ms/step -
loss: 0.2032 - accuracy: 0.9146 - val_loss: 0.1733 -
val_accuracy: 0.9271

Epoch 13/20
15/15 [==============================] - 11s 668ms/step -
loss: 0.2847 - accuracy: 0.9000 - val_loss: 0.1790 -
val_accuracy: 0.9167

Epoch 14/20
15/15 [==============================] - 9s 526ms/step - loss:
0.2768 - accuracy: 0.9146 - val_loss: 0.2564 - val_accuracy:
0.8958

Epoch 15/20
15/15 [==============================] - 10s 605ms/step -
loss: 0.1788 - accuracy: 0.9187 - val_loss: 0.1535 -
val_accuracy: 0.9271

Epoch 16/20
15/15 [==============================] - 11s 669ms/step -
loss: 0.1547 - accuracy: 0.9312 - val_loss: 0.1044 -
val_accuracy: 0.9271

Epoch 17/20
15/15 [==============================] - 11s 667ms/step -
loss: 0.1215 - accuracy: 0.9333 - val_loss: 0.1189 -
val_accuracy: 0.9271

Epoch 18/20
15/15 [==============================] - 11s 676ms/step -
loss: 0.1314 - accuracy: 0.9333 - val_loss: 0.1886 -
val_accuracy: 0.8750

Epoch 19/20
15/15 [==============================] - 10s 587ms/step -
loss: 0.1230 - accuracy: 0.9292 - val_loss: 0.0762 -
val_accuracy: 0.9479

Epoch 20/20
15/15 [==============================] - 11s 667ms/step -
loss: 0.1156 - accuracy: 0.9417 - val_loss: 0.1102 -
val_accuracy: 0.9479
```