DAFTAR PUSTAKA

Alemu, K. (2015) 'Detection of Diseases , Identification and Diversity of Viruses : A Review', 5(1), pp. 204–214.

Arief, M. (2020) *Pengendalian hama dan penyakit pada tanaman bayam*. Available at:http://cybex.pertanian.go.id/mobile/artikel/96328/PENGENDALIAN-HAMA-DAN-PENYAKIT-PADA-TANAMAN-BAYAM/.

Astuti, I., Ariestya, W.W. and Solehudin, B. (2022) 'Deteksi Objek Daun Semanggi Secara Real Time Menggunakan CNN-Single Shot Multibox Detector (SSD)', *Jurnal Ilmiah FIFO*, 14(1), p. 47. Available at: https://doi.org/10.22441/fifo.2022.v14i1.005.

Chen, J. and Ran, X. (2019) 'Deep Learning With Edge Computing: A Review', *Proceedings of the IEEE*, pp. 1–19. Available at: https://doi.org/10.1109/JPROC.2019.2921977.

Codingstudio.id (2023) *Pengertian Tensorflow Dalam Data Science*. Available at: https://codingstudio.id/blog/tensorflow-adalah-alam-data-science/.

Developers (2023) *Mengenal Android studio, developers*. Available at: https://developer.android.com/studio/intro?hl=id.

Dharma, S. (2014) 'Analisa usahatani bayam', *Wahana Inovasi Volume*, 3(1), pp. 73–83.

Fadillah, L. (2017) *Pengertian Android Studio, cara dan fungsinya, Androbuntu*. Available at: https://androbuntu.com/android-studio/.

FAQ, google colab (2020) *Google Collabolatory*. Available at: https://research.google.com/colaboratory/faq.html.

Fitriaty, I. and Mustofa, Y.A. (2019) 'Deteksi Penyakit Tanaman Daun Bayam Menggunakan Metode GLCM dan Artificial Neural Network (ANN)', *Cosphijournal.Unisan.Ac.Id*, 3(1), pp. 2597–9329. Available at: https://www.cosphijournal.unisan.ac.id/index.php/cosphihome/article/view/83.

Francis, M. and Deisy, C. (2019) 'Disease Detection and Classification in Agricultural Plants Using Convolutional Neural Networks - A Visual Understanding', *2019 6th International Conference on Signal Processing and Integrated Networks, SPIN 2019*, pp. 1063–1068. Available at: https://doi.org/10.1109/SPIN.2019.8711701.

Gurrala, K.K. *et al.* (2019) 'A new segmentation method for plant disease diagnosis', *2019 2nd International Conference on Intelligent Communication and Computational Techniques, ICCT 2019*, pp. 137–141. Available at: https://doi.org/10.1109/ICCT46177.2019.8969021.

, S. and Jhon, N. (2014) 'Kumpulan Karya Tulis Ilmiah "Inovasi

Medan Berhias (Bersih, Hijau, Asri dan Sehat)', *Pemanfaatan Running Text Dalam Mengatasi Kemacetan Menuju medan Berhias (Bersih, Hijau, Asri dan Sehat)*, p. 4.

Hermawati, D.T. (2016) 'Kajian Ekonomi antara Pola Tanam Monokultur dan Tumpangsari Tanaman Jagung, Kubis dan Bayam', *Kajian Ekonomi antara Pola Tanam Monokultur dan Tumpangsari Tanaman Jagung, Kubis dan Bayam*, 53(9), pp. 66–71.

Howard, A.G. *et al.* (2017) 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications'. Available at: http://arxiv.org/abs/1704.04861.

Jaisakthi, S.M. *et al.* (2019) 'Grape leaf disease identification using machine learning techniques', *ICCIDS 2019 - 2nd International Conference on Computational Intelligence in Data Science, Proceedings*, pp. 1–6. Available at: https://doi.org/10.1109/ICCIDS.2019.8862084.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) 'ImageNet classification with deep convolutional neural networks', *Communications of the ACM*, 60(6), pp. 84–90. Available at: https://doi.org/10.1145/3065386.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. Available at: https://doi.org/10.1038/nature14539.

Lin, T.Y. *et al.* (2020) 'Focal Loss for Dense Object Detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), pp. 318–327. Available at: https://doi.org/10.1109/TPAMI.2018.2858826.

Lingga, L. (2010) *Cerdas Memilih Sayuran*. Jakarta: PT Agromedia Pustaka. Available at: https://agromedia.net/katalog/cerdas-memilih-sayuran/.

Nugroho, K.S. (2019) *Confusion Matrix untuk Evaluasi Model pada Supervised Learning*, *medium*. Available at: https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f.

Peryanto, A., Yudhana, A. and Umar, R. (2020) 'Klasifikasi Citra Menggunakan Convolutional Neural Network dan K Fold Cross Validation', *Journal of Applied Informatics and Computing*, 4(1), pp. 45–51. Available at: https://doi.org/10.30871/jaic.v4i1.2017.

Reddy, J.N., Vinod, K. and Ajai, A.S.R. (2019) 'Analysis of Classification Algorithms for Plant Leaf Disease Detection', *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*, pp. 1–6. Available at: https://doi.org/10.1109/ICECCT.2019.8869090.

Rianto, D. and Ahmad, N. (2017) 'Optimalisasi Kandungan Serat pada Saus Bayam', *Jurnal Ilmiah Teknologi Pertanian AGROTECHNO*, 2(2), pp. 227–231.

ıita Frankes, S. and Seldev Christopher, C. (2019) 'Disease Identification in Spinach Leaves', *2019 International Conference on Recent Advances in Energy-Efficient Computing and*

Communication, ICRAECC 2019 [Preprint]. Available at: https://doi.org/10.1109/ICRAECC43874.2019.8995163.

Sandler, M. *et al.* (2018) 'Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf ', pp. 4510–4520.

Tamba, K.S., Hasibuan, N.A. and Silalahi, N. (2018) 'Sistem Pakar Mendiagnosa Hama dan Penyakit Pada Tanaman Bayam Dengan Metode Naïve Bayes', *Pelita Informatika*, 17(17), pp. 473–479. Available at: https://ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1088.

Wei Liu, Dragomir Anguelov, Dumitru Erhan Szegedy, Christian Reed, Scott, Fu, Cheng-Yang, Berg, A.C. (2016) 'SSD: Single Shot MultiBox Detector', in. Springer International Publishing, p. 17. Available at: https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2.

Yuda, Kresnamal dan Susmini, I.L. (2018) 'Aplikasi Pendeteksi Penyakit Pada Tanaman Bayam Berbasis Android (Studi Kasus Penyakit Karat)', *Jurnal Teknik Komputer Unikom*, 7(1), pp. 7–13.

Zainudhin, Z. (2016) *Mengenal penyakit tanaman bayam dan pengendaliannya*, *Agrotani.com*. Available at: https://www.agrotani.com/penyakit-tanaman-bayam/amp/.

Zou, Y. *et al.* (2020) 'Ship target detection and identification based on SSD_MobilenetV2', *Proceedings of 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference, ITOEC 2020*, (Itoec), pp. 1676–1680. Available at: https://doi.org/10.1109/ITOEC49072.2020.9141734.

LAMPIRAN

1. Lampiran Listing Program

```
!python --version

!pip install --upgrade pip

!pip install mediapipe-model-maker

from google.colab import files

import os

import json

import tensorflow as tf

assert tf.__version__.startswith('2')

from mediapipe_model_maker import object_detector

#dataset format COCO

# !pip install roboflow

# from roboflow import Roboflow

# rf = Roboflow(api_key="YuVlTDMltEsru3MPnh9U")

# project = rf.workspace("penyakit-bayam").project("penyakit-hama-
bayam")

# version = project.version(3)

# dataset = version.download("coco")

# !pip install roboflow

# from roboflow import Roboflow

# rf = Roboflow(api_key="YuVlTDMltEsru3MPnh9U")

# project = rf.workspace("penyakit-bayam").project("penyakit-bayam")

# version = project.version(2)

# dataset = version.download("coco")

!pip install roboflow

 roboflow import Roboflow

Roboflow(api_key="YuVlTDMltEsru3MPnh9U")
```

```
project = rf.workspace("penyakit-bayam").project("penyakit-bayam-new")
version = project.version(6)
dataset = version.download("coco")
train_dataset_path = "penyakit-bayam-new-6/train"
validation_dataset_path = "penyakit-bayam-new-6/valid"
test_dataset_path = "penyakit-bayam-new-6/test"
import os
import shutil
# Define the data directory
data_dir = train_dataset_path  # Replace with the path to your data directory
# Define the target subdirectory
images_dir = os.path.join(data_dir, 'images')
# Create the target subdirectory if it doesn't exist
if not os.path.exists(images_dir):
    os.makedirs(images_dir)
# Move all jpg files to the images subdirectory
for filename in os.listdir(data_dir):
    if filename.endswith('.jpg'):
        source_path = os.path.join(data_dir, filename)
        target_path = os.path.join(images_dir, filename)
        shutil.move(source_path, target_path)
# Define the original and new filenames
original_filename = '_annotations.coco.json'
new_filename = 'labels.json'
# Construct full paths to the original and new files
    nal_filepath = os.path.join(data_dir, original_filename)
    _filepath = os.path.join(data_dir, new_filename)
    name the file if it exists
```

```
if os.path.exists(original_filepath):
    os.rename(original_filepath, new_filepath)
    print(f"File renamed from {original_filename} to {new_filename}")
else:
    print(f"File {original_filename} does not exist in the directory {data_dir}"
# Path to the labels.json file
labels_filepath = os.path.join(data_dir, 'labels.json')

# Check if the file exists
if not os.path.exists(labels_filepath):
    print(f"File {labels_filepath} does not exist.")
else:
    # Read the existing labels.json file
    with open(labels_filepath, 'r') as file:
        data = json.load(file)
    # Change the category with id 0 to "background"
    for category in data.get('categories', []):
        if category.get('id') == 0:
            category['name'] = 'background'
            break
    # Write the updated data back to labels.json
    with open(labels_filepath, 'w') as file:
        json.dump(data, file, indent=4)
    print("Label index 0 changed to 'background' successfully!")
# Define the data directory
    _dir = validation_dataset_path  # Replace with the path to your data
    tory
    fine the target subdirectory
```

```
images_dir = os.path.join(data_dir, 'images')

# Create the target subdirectory if it doesn't exist
if not os.path.exists(images_dir):
    os.makedirs(images_dir)

# Move all jpg files to the images subdirectory
for filename in os.listdir(data_dir):
    if filename.endswith('.jpg'):
        source_path = os.path.join(data_dir, filename)
        target_path = os.path.join(images_dir, filename)
        shutil.move(source_path, target_path)

# Define the original and new filenames
original_filename = '_annotations.coco.json'
new_filename = 'labels.json'

# Construct full paths to the original and new files
original_filepath = os.path.join(data_dir, original_filename)
new_filepath = os.path.join(data_dir, new_filename)

# Rename the file if it exists
if os.path.exists(original_filepath):
    os.rename(original_filepath, new_filepath)
    print(f"File renamed from {original_filename} to {new_filename}")
```

```
    int(f"File {original_filename} does not exist in the directory
    ı_dir}")
```

```
print("Files moved successfully!")


# Path to the labels.json file
labels_filepath = os.path.join(data_dir, 'labels.json')
# Check if the file exists
if not os.path.exists(labels_filepath):
    print(f"File {labels_filepath} does not exist.")
else:
    # Read the existing labels.json file
    with open(labels_filepath, 'r') as file:
        data = json.load(file)
    # Change the category with id 0 to "background"
    for category in data.get('categories', []):
        if category.get('id') == 0:
            category['name'] = 'background'
            break
    # Write the updated data back to labels.json
    with open(labels_filepath, 'w') as file:
        json.dump(data, file, indent=4)
    print("Label index 0 changed to 'background' successfully!")
# Define the data directory
data_dir = test_dataset_path  # Replace with the path to your data directory


# Define the target subdirectory
images_dir = os.path.join(data_dir, 'images')
# Create the target subdirectory if it doesn't exist
    t os.path.exists(images_dir):
    .makedirs(images_dir)
```

```
# Move all jpg files to the images subdirectory
for filename in os.listdir(data_dir):
    if filename.endswith('.jpg'):
        source_path = os.path.join(data_dir, filename)
        target_path = os.path.join(images_dir, filename)
        shutil.move(source_path, target_path)
# Define the original and new filenames
original_filename = '_annotations.coco.json'
new_filename = 'labels.json'
# Construct full paths to the original and new files
original_filepath = os.path.join(data_dir, original_filename)
new_filepath = os.path.join(data_dir, new_filename)
# Rename the file if it exists
if os.path.exists(original_filepath):
    os.rename(original_filepath, new_filepath)
    print(f"File renamed from {original_filename} to {new_filename}")
else:
    print(f"File {original_filename} does not exist in the directory
{data_dir}")
print("Files moved successfully!")
# Path to the labels.json file
labels_filepath = os.path.join(data_dir, 'labels.json')
# Check if the file exists
if not os.path.exists(labels_filepath):
    print(f"File {labels_filepath} does not exist.")
else:
        Read the existing labels.json file
        th open(labels_filepath, 'r') as file:
        data = json.load(file)
```

```
    # Change the category with id 0 to "background"
    for category in data.get('categories', []):
        if category.get('id') == 0:
            category['name'] = 'background'
            break
    # Write the updated data back to labels.json
    with open(labels_filepath, 'w') as file:
        json.dump(data, file, indent=4)
    print("Label index 0 changed to 'background' successfully!")
with open(os.path.join(train_dataset_path, "labels.json"), "r") as f:
  labels_json = json.load(f)
for category_item in labels_json["categories"]:
  print(f"{category_item['id']}: {category_item['name']}")
train_data = object_detector.Dataset.from_coco_folder(train_dataset_path,
cache_dir="/tmp/od_data/train")
validation_data =
object_detector.Dataset.from_coco_folder(validation_dataset_path,
cache_dir="/tmp/od_data/validation")
test_data = object_detector.Dataset.from_coco_folder(test_dataset_path,
cache_dir="/tmp/od_data/test")
print("train_data size: ", train_data.size)
print("validation_data size: ", validation_data.size)
print("test_data size: ", test_data.size)
epoch_count = 130
spec = object_detector.SupportedModels.MOBILENET_MULTI_AVG
hparams = object_detector.HParams(export_dir='exported_model',
learning_rate=0.01,
      ochs=epoch_count)
ons = object_detector.ObjectDetectorOptions(
      pported_model=spec,
```

```
    hparams=hparams
)
model = object_detector.ObjectDetector.create(
    train_data=train_data,
    validation_data=validation_data,
    options=options)
loss, coco_metrics = model.evaluate(validation_data, batch_size=4)
print(f"Validation loss: {loss}")
print(f"Validation coco metrics: {coco_metrics}")
model.export_model()
# !ls exported_model
# files.download('exported_model/model.tflite')
from mediapipe_model_maker import quantization
model.restore_float_ckpt()
model.export_model(model_name="model_fp16.tflite",
quantization_config=quantization_config)
!ls -lh exported_model
files.download('exported_model/model_fp16.tflite')
```