LAMPIRAN

Source Code Models

```
import tensorflow as tf
import numpy as np
import tensorflow_hub as hub
import zipfile
import os
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from google.colab import drive

drive.mount('/content/drive/')

# Direktori dataset
train_dir = '/content/drive/MyDrive/skripsi/dataset/train'

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

ResNet =
hub.KerasLayer("https://www.kaggle.com/models/google/resnet-
v2/frameworks/TensorFlow2/variations/101-
classification/versions/2",
                              trainable=False)

model = tf.keras.Sequential([
    ResNet,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(1,  activation="sigmoid")
])

model.build((None, 224, 224, 3))
model.summary()

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0
.0001),
      loss='binary_crossentropy',
      metrics=['accuracy'])
class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
             if (logs.get('val_accuracy') > 0.92 and
logs.get('accuracy') > 0.92):
                 print("\nReached 92% accuracy so cancelling
training!")
                 self.model.stop_training = True
callback = myCallback()

history= model.fit(train_generator,
      validation_data=validation_generator,
      epochs=100,
      verbose=1,
      callbacks=[callback])

train_loss  =  history.history['loss']
val_loss = history.history['val_loss']
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

print("Training Loss:", train_loss)
print("Validation Loss:", val_loss)
print("Training  Accuracy:",  train_accuracy)
print("Validation  Accuracy:",  val_accuracy)
evaluation_result = model.evaluate(validation_generator)
print(evaluation_result)

model.save('resnet_battery_classification.h5')
```

```
import tensorflow as tf
import numpy as np
import tensorflow_hub as hub
import zipfile
import os
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from google.colab import drive

drive.mount('/content/drive/')

train_dir = '/content/drive/MyDrive/skripsi/dataset/train'

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

Inception = 
hub.KerasLayer("https://www.kaggle.com/models/google/inception-
v3/TensorFlow2/classification/2",
                              trainable=False)

model = tf.keras.Sequential([
    Inception,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(1,  activation="sigmoid")
])

model.build((None, 224, 224, 3))
model.summary()

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0
.0001),
      loss='binary_crossentropy',
      metrics=['accuracy'])
class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
            if (logs.get('val_accuracy') > 0.92 and
logs.get('accuracy') > 0.92):
                print("\nReached 92% accuracy so cancelling
training!")
                self.model.stop_training = True
callback = myCallback()

history= model.fit(train_generator,
      validation_data=validation_generator,
      epochs=100,
      verbose=1,
      callbacks=[callback])

train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_accuracy = history.history['accuracy']
```

```
val_accuracy = history.history['val_accuracy']

print("Training Loss:", train_loss)
print("Validation Loss:", val_loss)
print("Training  Accuracy:",  train_accuracy)
print("Validation  Accuracy:",  val_accuracy)
evaluation_result = model.evaluate(validation_generator)
print(evaluation_result)

model.save('inception_battery_classification.h5')
```

```
import tensorflow as tf
import numpy as np
import tensorflow_hub as hub
import zipfile
import os
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from google.colab import drive

drive.mount('/content/drive/')

train_dir = '/content/drive/MyDrive/skripsi/dataset/train'
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224,  224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224,  224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

MobileNet =
hub.KerasLayer("https://www.kaggle.com/models/google/mobilenet-
v3/TensorFlow2/large-075-224-classification/1",
                              trainable=False)

model = tf.keras.Sequential([
    MobileNet,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.build((None, 224, 224, 3))
model.summary()


model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0
.0001),
      loss='binary_crossentropy',
      metrics=['accuracy'])
class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
            if (logs.get('val_accuracy') > 0.92 and
logs.get('accuracy') > 0.92):
                print("\nReached 92% accuracy so cancelling
training!")
                self.model.stop_training = True
callback = myCallback()

history= model.fit(train_generator,
      validation_data=validation_generator,
      epochs=100,
      verbose=1,
      callbacks=[callback])

train_loss  =  history.history['loss']
val_loss = history.history['val_loss']
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

print("Training Loss:", train_loss)
print("Validation Loss:", val_loss)
print("Training  Accuracy:",  train_accuracy)
print("Validation  Accuracy:",  val_accuracy)
evaluation_result = model.evaluate(validation_generator)
print(evaluation_result)

model.save('mobilenet_battery_classification.h5')
```

Source Code Prediction

```
import os
import base64
import matplotlib.pyplot as plt
from datetime import datetime
import streamlit as st
import pandas as pd
from  crud.classification  import  get_all_classification,
delete_classification, delete_all_classification,
get_daily_defect_classification,
get_daily_non_defect_classification
from crud.report import create_report, update_report,
get_month_report, get_year_report,  get_all_report
def classification_menu():
    if st.session_state.role == "Supervisor":
        # report
        st.subheader("Report")
        card_col1, card_col2, card_col3 = st.columns(3)
        date = datetime.today().strftime("%d %b %Y")
        day = datetime.today().strftime("%d")
        month  =  datetime.today().strftime("%b")
        year = datetime.today().strftime("%Y")

        with card_col1.container(border=True):
            total_defect = get_daily_defect_classification(date)
            st.metric("Total Defect Today", total_defect[0])
        with card_col2.container(border=True):
            total_non_defect =
get_daily_non_defect_classification(date)
            st.metric("Total  Non-Defect  Today",
total_non_defect[0])
        with card_col3:
            if st.button("Create Daily Report", type =
"primary"):
                create_report(day,  month,  year,  total_defect[0],
total_non_defect[0])
            if st.button("Update Daily Report", type =
"primary"):
                update_report(day,  month,  year,  total_defect[0],
total_non_defect[0])

        card_graphic1, card_graphic2 = st.columns(2)
        with card_graphic1.container(border=True):
            total_monthly = get_month_report(month, year) #
[total defect, total non]

            labels = ["Defect", "Non-Defect"]
            values  =  [total_monthly[0][0],  total_monthly[0][1]]
            fig, ax = plt.subplots()
            ax.bar(labels, values)

            ax.set_title('Report Bulanan')
            ax.set_xlabel('Klasifikasi')
            ax.set_ylabel('Jumlah')

            st.pyplot(fig)

        with card_graphic2.container(border=True):
            total_yearly = get_year_report(year)
            months = []
            defect_values = []
            non_defect_values =[]
            for month in total_yearly[::-1]:
                months.append(month[0])
                defect_values.append(month[1])
                non_defect_values.append(month[2])

            fig, ax = plt.subplots()
            ax.plot(months,  defect_values,  marker='o',
linestyle='-', color='b', label='Defect')
            ax.plot(months, non_defect_values, marker='s',
linestyle='--', color='r', label='Non-defect')
            ax.set_title('Report Tahunan')
            ax.set_xlabel('Bulan')
            ax.set_ylabel('Jumlah')
            ax.legend()

            st.pyplot(fig)

    report_data = get_all_report()
    if report_data:
        report_list = []
        for i, data in enumerate(report_data, start=1):
            data = {
                "No": i,
                "Date": f"{data[0]} {data[1]} {data[2]}",
                "Total Defect": data[3],
                "Total Non-defect": data[4]
            }
            report_list.append(data)
        st.dataframe(report_list, use_container_width=True)

    st.subheader("Classification History")
    classification_data = get_all_classification()
    if classification_data:
        data_list = []
        for i, data in enumerate(classification_data, start=1):
            data = {
                "No.": i,
                "User ID": data[1],
                "Image": open_image(data[2]),
                "Filename": data[2],
                "Result": data[3],
                "Date": data[4],
                "Model": data[5],
            }
            data_list.append(data)
        data_df  = pd.DataFrame(data_list)

        classification_event= st.dataframe(
            data_df,
            column_config={
                "Image": st.column_config.ImageColumn(
                    "Image"
                ),
            },
            hide_index=True,
            on_select='rerun',
            selection_mode='single-row'
        )
        if st.session_state.role == "Supervisor":
            if st.button("Delete All", type = 'primary'):
                delete_all_classification()
                st.rerun()

        if len(classification_event.selection['rows']):
            selected_row =
classification_event.selection['rows'][0]
            user_id = data_df.iloc[selected_row]['User ID']
            image = data_df.iloc[selected_row]['Image']
            result = data_df.iloc[selected_row]['Result']
            date =  data_df.iloc[selected_row]['Date']
            model = data_df.iloc[selected_row]['Model']
            filename  =  data_df.iloc[selected_row]['Filename']
            classification_modal(user_id, image, result, date,
model, filename)
def open_image(filename):
    try:
        path = os.path.join("images/classification", filename)
        with open(path, "rb") as image_file:
            encoded_string =
base64.b64encode(image_file.read()).decode()
            return f"data:image/png;base64,{encoded_string}"
    except Exception as e:
        st.error(f"Error converting image to base64: {e}")
        return ""

@st.experimental_dialog("History Details", width="small")
def classification_modal(user_id, image, result, date, model,
filename):
    st.image(image, width=400)
    st.subheader(result)
    if "resnet" in model:
        st.write("Model: Resnet")
    elif "mobilenet" in model:
        st.write("Model: Mobilenet")
    elif "inception" in model:
        st.write("Model: Inception")
    st.write("Date: "+date[:11])
    if st.session_state.role=="Supervisor":
        if st.button("Delete", type="primary"):
            delete_classification(filename)
            st.rerun()
```

Connection to Database

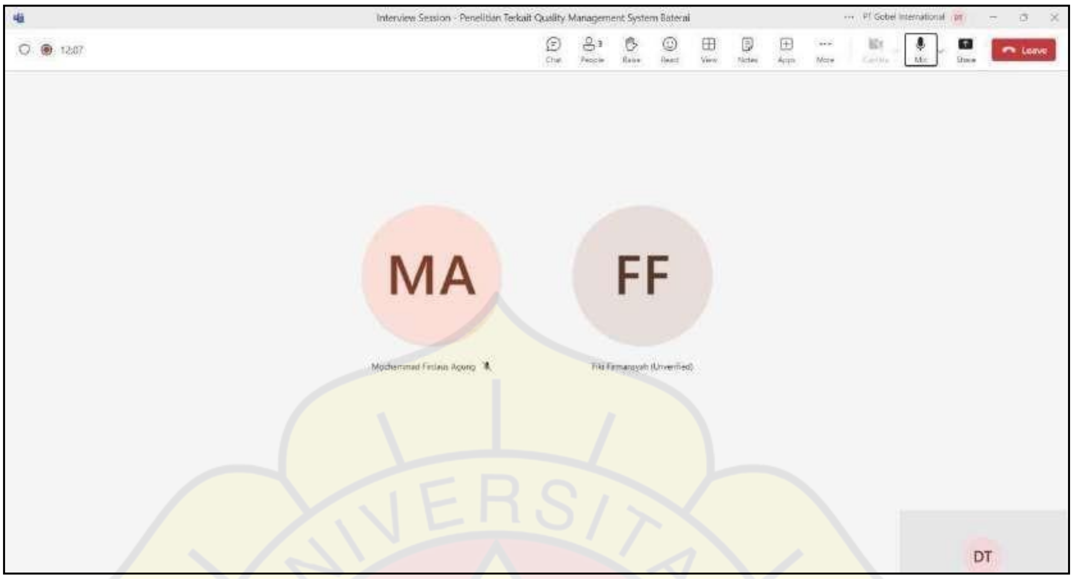
```
import sqlite3

def db_connection():
    conn = sqlite3.connect('history.db')
    return conn
```

Dokumentasi Saat Meminta Izin Untuk Penelitian

Dokumentasi Sesi Wawancara *Online* dengan salah satu Tim *Quality Control*

PT. Panasonic Gobel Energy Indonesia



Dokumentasi Pengambilan Dataset Gambar

HASIL WAWANCARA

Narasumber	:	Bapak Fiki Firmansyah
Jabatan	:	Tim *Quality Control*
Tempat Wawancara	:	*Microsoft Teams*
Tanggal Wawancara	:	28 Mei 2024

No.	Pertanyaan	Jawaban
1	Apa yang menjadi indikator utama kerusakan baterai yang sering terjadi selama proses produksi ?	Terdapat beberapa indikator utama yang menyebabkan kerusakan baterai yang umum terjadi, seperti gores (*scratch*), penyok dan kotor (seperti bahan kimia yang menempel saat proses produksi).
2	Bagaimana cara mengevaluasi kerusakan baterai yang ditemukan selama proses produksi ?	Mengkategorikannya secara terpisah, jika baterai termasuk dalam kategori *non-defect*, maka baterai tersebut akan lanjut untuk masuk ke tahap selanjutnya, sedangkan jika baterai termasuk dalam kategori *defect*, maka baterai tersebut akan melalui beberapa proses untuk mengolah baterai secara lebih lanjut.
3	Bagaimana prosedur atau langkah-langkah yang diambil	Setelah mendapat informasi dari tim produksi, kemudian tim *quality control*

	saat baterai dinyatakan rusak selama proses produksi ?	akan melakukan pengecekkan ulang untuk menyatakan apakah baterai termasuk dalam kategori *defect*, setelah melalui pengecekkan, tim *quality control* akan membuat laporan kepada tim *supervisor* dan pimpinan terlebih dahulu untuk melakukan diskusi, kemudian biasanya akan dilakukan proses lebih lanjut untuk mengolah kembali seluruh baterai yang teridentifikasi *defect*.
4	Selama proses produksi di perusahaan ini, apakah terdapat hal yang umum terjadi terkait dengan kerusakan baterai?	Kerusakan yang sering terjadi pada baterai biasanya berupa goresan dan penyok yang seringkali disebabkan oleh mesin saat proses produksi.
5	Bagaimana sistem pemantauan dan pengendalian kualitas yang diterapkan untuk mencegah kerusakan baterai selama proses produksi ?	Biasanya pemantauan dilakukan menggunakan alat sensor untuk mendeteksi baterai *defect* dan *non-defect*, namun alat sensor tersebut juga tidak selalu akurat, karena beberapa faktor seperti pencahayaan yang kurang memadai, sistem yang belum maksimal saat mendeteksi tingkat *defect* seperti

		goresan ataupun penyok pada baterai yang sangat kecil yang mengharuskan untuk dilakukan *sampling* dan pengecekkan ulang oleh tim *quality control*.
6	Apakah ada perubahan atau peningkatan yang telah dilakukan dalam upaya mengurangi jumlah kerusakan baterai selama proses produksi ?	Pembuatan sistem untuk dapat mengklasifikasikan gambar baterai apakah termasuk dalam kategori *defect* atau *non-defect* yang saat ini tengah dikembangkan, agar tidak lagi mengklasifikasikan baterai secara manual hanya dengan indera peraba.
7	Bagaimana proses identifikasi penyebab akar kerusakanbaterai yang dilakukan di perusahaan ini ?	Berdasarkan perbandingan gambar baterai yang *defect* dan *non-defect*, namun sistem yang ingin digunakan masih dalam tahap pembuatan.
8	Apakah ada langkah-langkah spesifik yang diambil untuk mengatasi kerusakan baterai yang terjadi selama proses produksi ?	Pembuatan sistem alat ukur untuk mendeteksi kedalaman dan *defect* baterai secara spesifik yang saat ini tengah dikembangkan.