

**LAPORAN AKHIR
PENAMBANGAN DATA**







Oleh :


Carissa Arivia (2257301027)

Naufal Khairy (2257301101)

**JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI DIV SISTEM INFORMASI
POLITEKNIK CALTEX RIAU
2024**

No	Perintah	Analisa
1.	 <pre>from google.colab import drive drive.mount('/content/drive')</pre>	Kode ini digunakan untuk menghubungkan Google Colab dengan Google Drive. Dengan `drive.mount('/content/drive')`, Google Colab akan diberikan akses untuk membaca dan menulis file di Google Drive. Setelah menjalankan baris ini, kami akan diminta untuk mengautentikasi dan memberikan izin, dan kemudian kami dapat mengakses file di Google Drive menggunakan jalur `/content/drive`.
Output		
	 Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).	

No	Perintah	Analisa
2.	 <pre>import zipfile import os # Define the path to the uploaded file zip_file_path = '/content/drive/MyDrive/BACKEND DM/Dataaaa.zip' extract_path = '/content/drive/MyDrive/BACKEND DM/dm' # Extract the zip file with zipfile.ZipFile(zip_file_path, 'r') as zip_ref: zip_ref.extractall(extract_path) # List the contents of the extracted directory extracted_files = os.listdir(extract_path) extracted_files</pre>	Kode ini mengekstrak file ZIP yang diunggah ke Google Drive dan menampilkan daftar file yang ada di dalam direktori hasil ekstraksi.
Output		
	 ['melanoma', 'nevus', 'pigmented benign keratosis']	

No	Perintah	Analisa
3.	<pre>skin_cancer_path = os.path.join(extract_path, '') skin_cancer_files = os.listdir(skin_cancer_path) skin_cancer_files</pre>	Kode ini digunakan untuk menentukan jalur ke direktori hasil ekstraksi dan menampilkan daftar file yang ada di dalamnya. Dengan `os.path.join(extract_path, "")`, kode menyusun jalur ke direktori yang diinginkan, lalu `os.listdir(skin_cancer_path)` digunakan untuk mendapatkan dan menampilkan daftar file di direktori tersebut.
Output		
	 ['melanoma', 'nevus', 'pigmented benign keratosis', 'r', '.ipynb_checkpoints']	

No	Perintah	Analisa
4.	<pre>import os import random import shutil from sklearn.model_selection import train_test_split import tensorflow as tf from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras import layers, models import matplotlib.pyplot as plt</pre>	<p>Kode ini mengimpor pustaka dan modul penting untuk pemrosesan data, pelatihan model, dan visualisasi. Modul `os`, `random`, dan `shutil` digunakan untuk manipulasi file dan direktori, sementara `train_test_split` membagi dataset menjadi pelatihan dan pengujian. `tensorflow` dan `tensorflow.keras` digunakan untuk membangun dan melatih model pembelajaran mendalam, dan `matplotlib.pyplot` untuk membuat visualisasi grafik dari hasil pelatihan.</p>
Output		

No	Perintah	Analisa
5.	<pre># Define paths for the dataset base_dir = skin_cancer_path classes = ['melanoma', 'nevus', 'pigmented benign keratosis'] # Define temporary directories for train and validation splits train_dir = os.path.join(base_dir, 'train') validation_dir = os.path.join(base_dir, 'validation') if not os.path.exists(train_dir): os.makedirs(train_dir) if not os.path.exists(validation_dir): os.makedirs(validation_dir) # Function to create class subdirectories def create_class_subdirs(base_path, class_names): for class_name in class_names: path = os.path.join(base_path, class_name) if not os.path.exists(path): os.makedirs(path)</pre>	<p>Kode ini mendefinisikan jalur untuk dataset dan menyiapkan direktori untuk membagi data menjadi set pelatihan dan validasi. `base_dir` adalah jalur dasar ke dataset, sedangkan `train_dir` dan `validation_dir` adalah direktori untuk set pelatihan dan validasi. Jika direktori ini belum ada, kode ini akan membuatnya. Fungsi `create_class_subdirs` kemudian membuat subdirektori untuk setiap kelas (melanoma, nevus, dan pigmented benign keratosis) di dalam direktori yang diberikan.</p>
Output		

No	Perintah	Analisa
6.	<pre> create_class_subdirs(train_dir, classes) create_class_subdirs(validation_dir, classes) def copy_images(src_dir, dest_dir, n_images): all_images = os.listdir(src_dir) selected_images = random.sample(all_images, min(n_images, len(all_images))) for img in selected_images: src = os.path.join(src_dir, img) dest = os.path.join(dest_dir, img) shutil.copyfile(src, dest) n_train_images_per_class = 391 n_validation_images_per_class = 266 # Copy images to train and validation directories for class_name in classes: src_class_dir = os.path.join(base_dir, class_name) dest_train_class_dir = os.path.join(train_dir, class_name) dest_validation_class_dir = os.path.join(validation_dir, class_name) copy_images(src_class_dir, dest_train_class_dir, n_train_images_per_class) copy_images(src_class_dir, dest_validation_class_dir, n_validation_images_per_class) # Parameters batch_size = 32 img_height = 180 img_width = 180 </pre>	<p>Kode ini mengatur pemindahan gambar ke direktori pelatihan dan validasi berdasarkan jumlah gambar yang diinginkan. Fungsi `create_class_subdirs` membuat subdirektori untuk setiap kelas di dalam direktori pelatihan dan validasi. Fungsi `copy_images` memilih gambar secara acak dari direktori sumber dan menyalinnya ke direktori tujuan. Kode ini kemudian menetapkan jumlah gambar untuk pelatihan (`n_train_images_per_class`) dan validasi (`n_validation_images_per_class`) serta menyalin gambar ke direktori yang sesuai. Terakhir, kode menetapkan parameter untuk pemrosesan gambar, seperti ukuran batch dan dimensi gambar.</p>
	Output	

No	Perintah	Analisa
7.	<pre> train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) validation_datagen = ImageDataGenerator(rescale=1./255) </pre>	<p>Kami menggunakan `ImageDataGenerator` untuk melakukan augmentasi data dan membuat generator data untuk pelatihan dan validasi. `train_datagen` mengatur augmentasi pada data pelatihan dengan teknik seperti penskalaan ulang (`rescale`), pemotongan miring (`shear_range`), zoom (`zoom_range`), dan pembalikan horizontal (`horizontal_flip`). Sementara itu, `validation_datagen` hanya melakukan penskalaan ulang pada data validasi. Ini membantu meningkatkan variasi dan ukuran dataset pelatihan serta memastikan konsistensi data saat validasi.</p>
	Output	

No	Perintah	Analisa
8.	<pre> train_generator = train_datagen.flow_from_directory(train_dir, target_size=(img_height, img_width), batch_size=batch_size, class_mode='categorical') validation_generator = validation_datagen.flow_from_directory(validation_dir, target_size=(img_height, img_width), batch_size=batch_size, class_mode='categorical') </pre>	<p>Kami menggunakan `flow_from_directory` untuk membuat generator data pelatihan dan validasi dari direktori yang telah diatur. `train_generator` dan `validation_generator` masing-masing mengonfigurasi generator untuk memuat gambar dari direktori pelatihan dan validasi, mengubah ukuran gambar ke dimensi yang ditentukan (`target_size`), menetapkan ukuran batch (`batch_size`), dan mengatur mode kelas sebagai 'categorical' untuk klasifikasi multi-kelas. Generator ini mempermudah proses pelatihan dan evaluasi model dengan menyediakan data yang sudah diproses dalam batch.</p>
	Output	
	<pre> Found 1071 images belonging to 3 classes. Found 798 images belonging to 3 classes. </pre>	

No	Perintah	Analisa
9.	<pre> model = models.Sequential([layers.InputLayer(input_shape=(img_height, img_width, 3)), layers.Conv2D(32, (3, 3), activation='relu'), layers.MaxPooling2D((2, 2)), layers.Conv2D(64, (3, 3), activation='relu'), layers.MaxPooling2D((2, 2)), layers.Conv2D(128, (3, 3), activation='relu'), layers.MaxPooling2D((2, 2)), layers.Flatten(), layers.Dense(128, activation='relu'), layers.Dense(3, activation='softmax')]) </pre>	<p>Kami mendefinisikan model jaringan saraf konvolusi dengan `models.Sequential` yang dimulai dengan lapisan input untuk gambar berukuran `(img_height, img_width, 3)`. Model ini mencakup tiga lapisan konvolusi dengan 32, 64, dan 128 filter masing-masing, diikuti oleh lapisan pooling untuk mengurangi dimensi spasial. Setelah itu, lapisan `Flatten` mengubah output konvolusi menjadi vektor 1D, yang kemudian diproses oleh lapisan dense dengan 128 neuron dan aktivasi ReLU, sebelum mencapai lapisan output dengan 3 neuron dan aktivasi softmax untuk klasifikasi multi-kelas. Model ini dirancang untuk mengklasifikasikan gambar ke dalam tiga kategori berbeda.</p>
	Output	

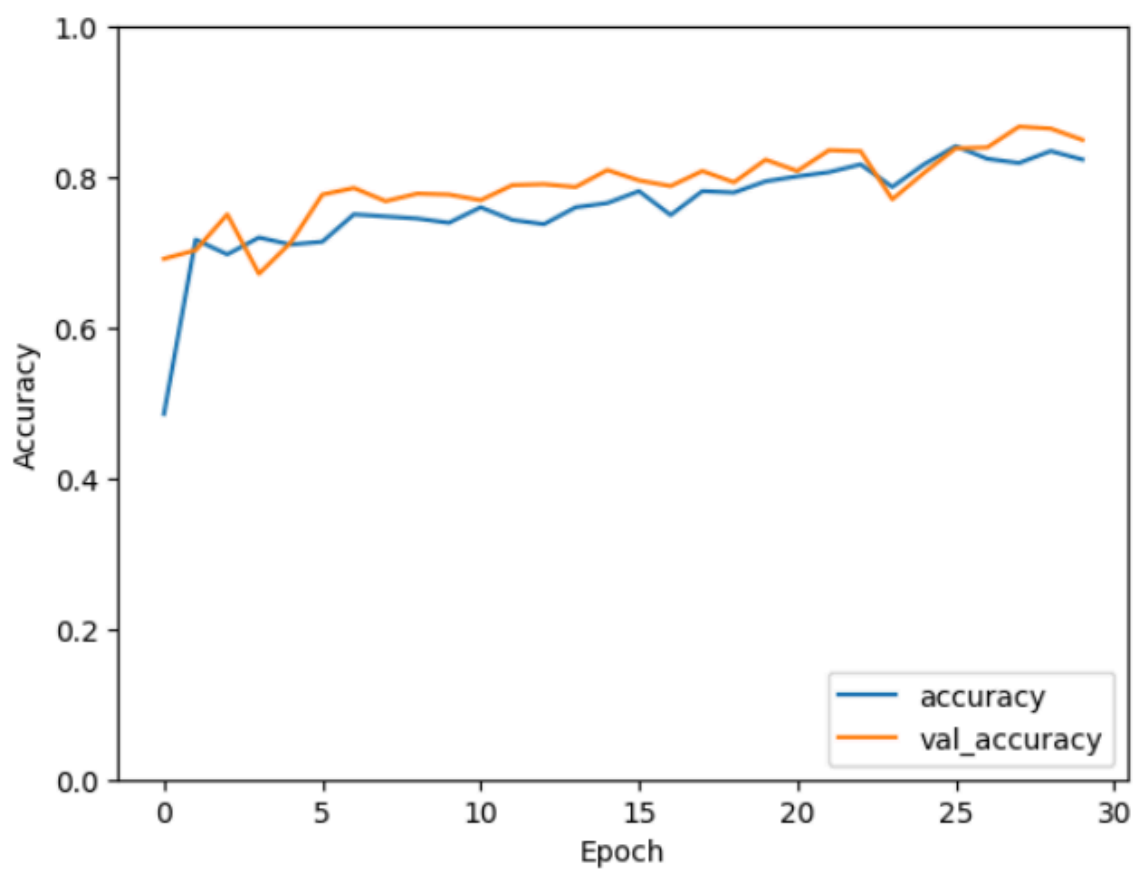
No	Perintah	Analisa
10	<pre> model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) </pre>	<p>Kami mengonfigurasi model dengan metode `compile` menggunakan optimizer 'adam', fungsi loss 'categorical_crossentropy', dan metrik 'accuracy'. Optimizer 'adam' mengoptimalkan pembelajaran, 'categorical_crossentropy' digunakan sebagai fungsi loss untuk klasifikasi multi-kelas, dan 'accuracy' diukur untuk mengevaluasi kinerja model selama pelatihan dan evaluasi.</p>
	Output	

No	Perintah	Analisa
11	<pre> # Train the model history = model.fit(train_generator, validation_data=validation_generator, epochs=30) # Plot training and validation accuracy plt.plot(history.history['accuracy'], label='accuracy') plt.plot(history.history['val_accuracy'], label = 'val_accuracy') plt.xlabel('Epoch') plt.ylabel(['Accuracy']) plt.ylim([0, 1]) plt.legend(loc='lower right') plt.show() # Evaluate the model on validation set val_loss, val_accuracy = model.evaluate(validation_generator) print(f'Validation accuracy: {val_accuracy}') </pre>	<p>Kami melatih model menggunakan `model.fit` dengan generator pelatihan dan validasi selama 30 epoch. Setelah pelatihan, kami memvisualisasikan akurasi pelatihan dan validasi dengan grafik menggunakan `matplotlib`, menampilkan perubahan akurasi selama epoch. Terakhir, kami mengevaluasi model pada set validasi dan mencetak akurasi validasi untuk menilai kinerja model.</p>
Output		
	<pre> Epoch 1/30 34/34 [=====] - 136s 4s/step - loss: 1.0357 - accuracy: 0.4855 - val_loss: 0.7298 - val_accuracy: 0.6917 Epoch 2/30 34/34 [=====] - 125s 4s/step - loss: 0.6882 - accuracy: 0.7171 - val_loss: 0.6910 - val_accuracy: 0.7030 Epoch 3/30 34/34 [=====] - 124s 4s/step - loss: 0.7070 - accuracy: 0.6975 - val_loss: 0.6030 - val_accuracy: 0.7506 Epoch 4/30 34/34 [=====] - 124s 4s/step - loss: 0.6557 - accuracy: 0.7199 - val_loss: 0.7416 - val_accuracy: 0.6717 Epoch 5/30 34/34 [=====] - 130s 4s/step - loss: 0.6918 - accuracy: 0.7106 - val_loss: 0.6940 - val_accuracy: 0.7130 Epoch 6/30 34/34 [=====] - 124s 4s/step - loss: 0.6715 - accuracy: 0.7143 - val_loss: 0.5475 - val_accuracy: 0.7769 Epoch 7/30 34/34 [=====] - 123s 4s/step - loss: 0.6040 - accuracy: 0.7507 - val_loss: 0.5177 - val_accuracy: 0.7857 Epoch 8/30 34/34 [=====] - 125s 4s/step - loss: 0.6057 - accuracy: 0.7479 - val_loss: 0.5680 - val_accuracy: 0.7682 Epoch 9/30 34/34 [=====] - 127s 4s/step - loss: 0.6282 - accuracy: 0.7451 - val_loss: 0.5876 - val_accuracy: 0.7782 Epoch 10/30 34/34 [=====] - 131s 4s/step - loss: 0.6151 - accuracy: 0.7395 - val_loss: 0.5230 - val_accuracy: 0.7769 Epoch 11/30 34/34 [=====] - 126s 4s/step - loss: 0.5857 - accuracy: 0.7600 - val_loss: 0.5540 - val_accuracy: 0.7694 Epoch 12/30 34/34 [=====] - 124s 4s/step - loss: 0.5804 - accuracy: 0.7432 - val_loss: 0.4922 - val_accuracy: 0.7895 Epoch 13/30 34/34 [=====] - 125s 4s/step - loss: 0.6124 - accuracy: 0.7376 - val_loss: 0.4855 - val_accuracy: 0.7907 Epoch 14/30 34/34 [=====] - 127s 4s/step - loss: 0.5614 - accuracy: 0.7600 - val_loss: 0.5178 - val_accuracy: 0.7870 Epoch 15/30 34/34 [=====] - 130s 4s/step - loss: 0.5340 - accuracy: 0.7656 - val_loss: 0.4701 - val_accuracy: 0.8095 Epoch 16/30 34/34 [=====] - 124s 4s/step - loss: 0.5342 - accuracy: 0.7815 - val_loss: 0.4769 - val_accuracy: 0.7957 </pre>	

```

Epoch 17/30
34/34 [=====] - 137s 4s/step - loss: 0.5914 - accuracy: 0.7498 - val_loss: 0.4919 - val_accuracy: 0.7882
Epoch 18/30
34/34 [=====] - 137s 4s/step - loss: 0.5261 - accuracy: 0.7815 - val_loss: 0.4493 - val_accuracy: 0.8083
Epoch 19/30
34/34 [=====] - 120s 4s/step - loss: 0.4981 - accuracy: 0.7796 - val_loss: 0.4971 - val_accuracy: 0.7932
Epoch 20/30
34/34 [=====] - 124s 4s/step - loss: 0.5091 - accuracy: 0.7946 - val_loss: 0.4244 - val_accuracy: 0.8233
Epoch 21/30
34/34 [=====] - 124s 4s/step - loss: 0.4539 - accuracy: 0.8011 - val_loss: 0.4717 - val_accuracy: 0.8083
Epoch 22/30
34/34 [=====] - 131s 4s/step - loss: 0.4909 - accuracy: 0.8067 - val_loss: 0.4146 - val_accuracy: 0.8358
Epoch 23/30
34/34 [=====] - 122s 4s/step - loss: 0.4540 - accuracy: 0.8170 - val_loss: 0.3955 - val_accuracy: 0.8346
Epoch 24/30
34/34 [=====] - 124s 4s/step - loss: 0.5012 - accuracy: 0.7871 - val_loss: 0.5305 - val_accuracy: 0.7707
Epoch 25/30
34/34 [=====] - 124s 4s/step - loss: 0.4457 - accuracy: 0.8170 - val_loss: 0.4532 - val_accuracy: 0.8058
Epoch 26/30
34/34 [=====] - 137s 4s/step - loss: 0.3984 - accuracy: 0.8413 - val_loss: 0.3646 - val_accuracy: 0.8383
Epoch 27/30
34/34 [=====] - 138s 4s/step - loss: 0.4382 - accuracy: 0.8245 - val_loss: 0.3958 - val_accuracy: 0.8396
Epoch 28/30
34/34 [=====] - 137s 4s/step - loss: 0.4370 - accuracy: 0.8189 - val_loss: 0.3783 - val_accuracy: 0.8672
Epoch 29/30
34/34 [=====] - 124s 4s/step - loss: 0.3936 - accuracy: 0.8347 - val_loss: 0.3154 - val_accuracy: 0.8647
Epoch 30/30
34/34 [=====] - 136s 4s/step - loss: 0.4018 - accuracy: 0.8235 - val_loss: 0.3605 - val_accuracy: 0.8496

```



```

25/25 [=====] - 26s 1s/step - loss: 0.3605 - accuracy: 0.8496
Validation accuracy: 0.8496240377426147

```

No	Perintah	Analisa
12	<pre>import numpy as np from google.colab import files from keras.preprocessing import image import matplotlib.pyplot as plt import matplotlib.image as mpimg import matplotlib.pyplot as plt import matplotlib.image as mpimg %matplotlib inline</pre>	Kami mengimpor pustaka penting untuk pemrosesan gambar dan visualisasi. Pustaka 'numpy' digunakan untuk manipulasi array, 'files' dari Google Colab untuk mengunggah file, dan 'image' dari 'keras.preprocessing' untuk memuat gambar. 'matplotlib.pyplot' dan 'matplotlib.image' digunakan untuk visualisasi dan menampilkan gambar. Dengan '%matplotlib inline', kami mengaktifkan plotting inline sehingga grafik dan gambar dapat ditampilkan langsung dalam notebook.
	Output	

No	Perintah	Analisa
13	<pre>from google.colab import files from keras.preprocessing import image import numpy as np import matplotlib.pyplot as plt # Mengunggah file gambar menggunakan widget unggah dari Jupyter Notebook uploaded = files.upload() # Iterasi melalui setiap file yang diunggah for fn in uploaded.keys(): # Mendapatkan path file yang diunggah path = fn # Memuat gambar dari path dan mengubah ukurannya sesuai target size yang diharapkan oleh model img = image.load_img(path, target_size=(180, 180)) # Menampilkan gambar imgplot = plt.imshow(img) plt.show() # Mengonversi gambar ke array x = image.img_to_array(img) # Menambahkan dimensi tambahan untuk batch size (menjadi bentuk (1, 180, 180, 3)) x = np.expand_dims(x, axis=0) # Menggabungkan array gambar ke dalam batch images = np.vstack([x]) # Melakukan prediksi menggunakan model yang telah dilatih classes = model.predict(images, batch_size=10) # Menampilkan nama file gambar print(fn) # Menentukan kelas berdasarkan hasil prediksi if classes[0][0] == 1: print('Melanoma') elif classes[0][1] == 1: print('Pigmen') elif classes[0][2] == 1: print('Nevus') else: print('unknown')</pre>	Kami membuat kode agar bisa mengunggah file gambar menggunakan widget unggah dari Google Colab dan memproses setiap file yang diunggah. Gambar dimuat dan diubah ukurannya menjadi '(180, 180)', kemudian ditampilkan menggunakan 'matplotlib'. Gambar dikonversi menjadi array dan diproses dengan menambahkan dimensi batch. Kami menggabungkan gambar menjadi batch dan melakukan prediksi menggunakan model yang telah dilatih. Berdasarkan hasil prediksi, kami menentukan kelas gambar (Melanoma, Pigmen, Nevus) dan menampilkan nama file bersama dengan kelas yang terdeteksi.
	Output	

Choose Files

No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving ISIC_0001134.jpg to ISIC_0001134.jpg

1/1 [=====] - 0s 56ms/step

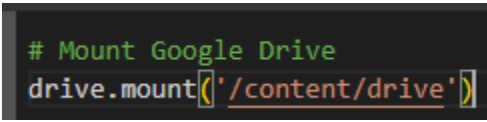
ISIC_0001134.jpg

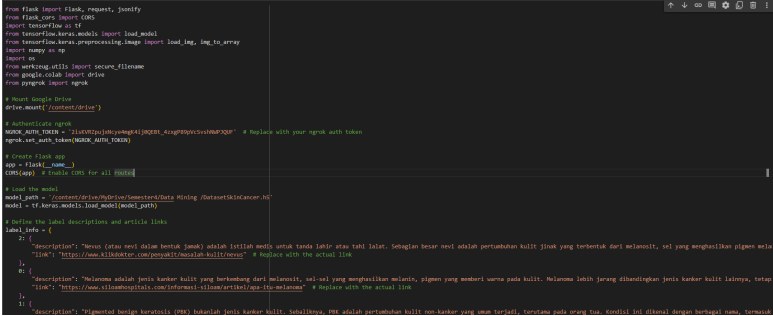
Melonema

No	Perintah	Analisa
14	<pre>model.save("DatasetSkinCancer.h5")</pre>	Kami menyimpan model yang telah dilatih ke dalam file dengan nama "DatasetSkinCancer.h5" menggunakan metode 'save'.
Output		
	<pre>/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered l saving_api.save_model(</pre>	

No	Perintah	Analisa
15	<pre># Install necessary packages !pip install flask-ngrok !pip install flask !pip install tensorflow !pip install pyngrok !pip install flask-cors # Import necessary libraries from flask import Flask, request, jsonify from werkzeug.utils import secure_filename from tensorflow.keras.preprocessing.image import load_img, img_to_array import tensorflow as tf import os from google.colab import drive from pyngrok import ngrok</pre>	Kami menginstal paket-paket yang diperlukan untuk mengembangkan aplikasi web menggunakan Flask dan TensorFlow, termasuk 'flask-ngrok', 'flask', 'tensorflow', 'pyngrok', dan 'flask-cors'. Setelah itu, kami mengimpor pustaka yang diperlukan untuk membangun aplikasi web, termasuk Flask untuk pengembangan web, 'secure_filename' dari 'werkzeug' untuk menangani nama file yang aman, serta TensorFlow untuk pemrosesan gambar dan model. Kami juga mengimpor 'ngrok' untuk membuat terowongan ke server lokal dan 'drive' dari Google Colab untuk akses ke Google Drive jika diperlukan.
Output		

	<pre>Collecting flask-ngrok Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB) Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.10/dist-packages (from flask-ngrok) (2.2.5) Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from flask-ngrok) (2.31.0) Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages (from Flask>=0.8->flask-ngrok) (3.0.3) Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from Flask>=0.8->flask-ngrok) (3.1.4) Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from Flask>=0.8->flask-ngrok) (2.2.0) Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from Flask>=0.8->flask-ngrok) (8.1.7) Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (3.3.2) Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (3.7) Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (2.0.7) Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (2024.7.4) Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=3.0->Flask>=0.8->flask-ngrok) (2.1.5) Installing collected packages: flask-ngrok Successfully installed flask-ngrok-0.0.25 Requirement already satisfied: flask in /usr/local/lib/python3.10/dist-packages (2.2.5) Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages (from flask) (3.0.3) Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from flask) (3.1.4) Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from flask) (2.2.0) Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from flask) (8.1.7) Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=3.0->flask) (2.1.5) Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0) Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0) Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3) Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25) Requirement already satisfied: gast>=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0) Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0) Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0) Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1) Requirement already satisfied: ml-dtypes==0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0) Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2) Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0) Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1) Requirement already satisfied: protobuf<4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3) Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2) Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0) Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0) Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2) Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1) Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1) Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1) Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)</pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

No	Perintah	Analisa
16		Agar bisa terhubung ke google drive
	Output	

No	Perintah	Analisa
17		Setelah menginstal paket yang diperlukan dan mem-mount Google Drive untuk mengakses model, kami membuat aplikasi Flask dengan CORS diaktifkan. Kami memuat model yang disimpan di Google Drive dan mendefinisikan informasi label untuk klasifikasi gambar. Aplikasi ini memiliki dua route: satu untuk menampilkan pesan selamat datang dan satu lagi untuk melakukan prediksi berdasarkan gambar yang diunggah. Gambar yang diunggah diproses, diprediksi, dan hasilnya disajikan dengan deskripsi dan tautan terkait. Ngrok digunakan untuk membuat terowongan dan menyediakan URL publik untuk mengakses aplikasi Flask.

```

}

# define root route
app.route('/', methods=['GET'])
def home():
    return "Welcome world!"

# define prediction route
app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify(success=False, error='No file part')
    file = request.files['file']
    if file.filename == '':
        return jsonify(success=False, error='No selected file')
    if file:
        # Save the filename and save it
        filename = secure_filename(file.filename)
        filepath = os.path.join('/tmp', filename)
        file.save(filepath)

        try:
            # Load and preprocess the image
            img = load_img(filepath, target_size=(100, 100)) # Adjust target size to 100x100
            img_array = img.to_array()
            img_array = np.expand_dims(img_array, axis=0)
            img_array = img_array / 255.0

            # Predict using the model
            prediction = model.predict(img_array)
            predicted_class = np.argmax(prediction)
            confidence = np.max(prediction) * 100

            # Map the predicted class to the descriptive label
            labels = {'classroom': 'classroom', 'house': 'house'} # Adjust according to your model's classes
            result = {
                'label': labels[predicted_class],
                'confidence': confidence,
                'description': labels[predicted_class]['description'],
                'link': labels[predicted_class]['link']
            }

            return jsonify(success=True, 'label': result['label'], 'confidence': result['confidence'], 'description': result['description'], 'link': result['link'])
        except Exception as e:
            return jsonify(success=False, error=str(e))

if __name__ == '__main__':
    # Ensure the /tmp directory exists
    if not os.path.exists('/tmp'):
        os.makedirs('/tmp')

    # Start ngrok tunnel
    public_url = ngrok.connect()
    print(f'Public URL: {public_url}')

    # Run the flask app
    app.run(port=5000)
```

Output

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Public URL: NgrokTunnel: "https://fd8a-34-143-169-220.ngrok-free.app" -> "http://localhost:5000"
* Serving Flask app '__main__'
* Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
1/1 [=====] - 0s 70ms/step
INFO:werkzeug:127.0.0.1 - - [21/Jul/2024 07:24:46] "POST /predict HTTP/1.1" 200 -
1/1 [=====] - 0s 18ms/step
INFO:werkzeug:127.0.0.1 - - [21/Jul/2024 07:24:56] "POST /predict HTTP/1.1" 200 -
```

Logbook kegiatan

Pekerjaan	Bulan Juni - Juli				PIC
	Minggu 2 Juni	Minggu 3 Juni	Minggu 4 Juni	Minggu 3 Juli	
Membuat Machine Learning menggunakan python	100%	-	-	-	Carissa, Naufal
Membuat halaman web menggunakan react.js	-	100%	-	-	Carissa, Naufal
Menghubungkan Machine Learning dan halaman web	-	-	100%	-	Carissa, Naufal
Membuat laporan project	-	-	-	100%	Carissa, Naufal

