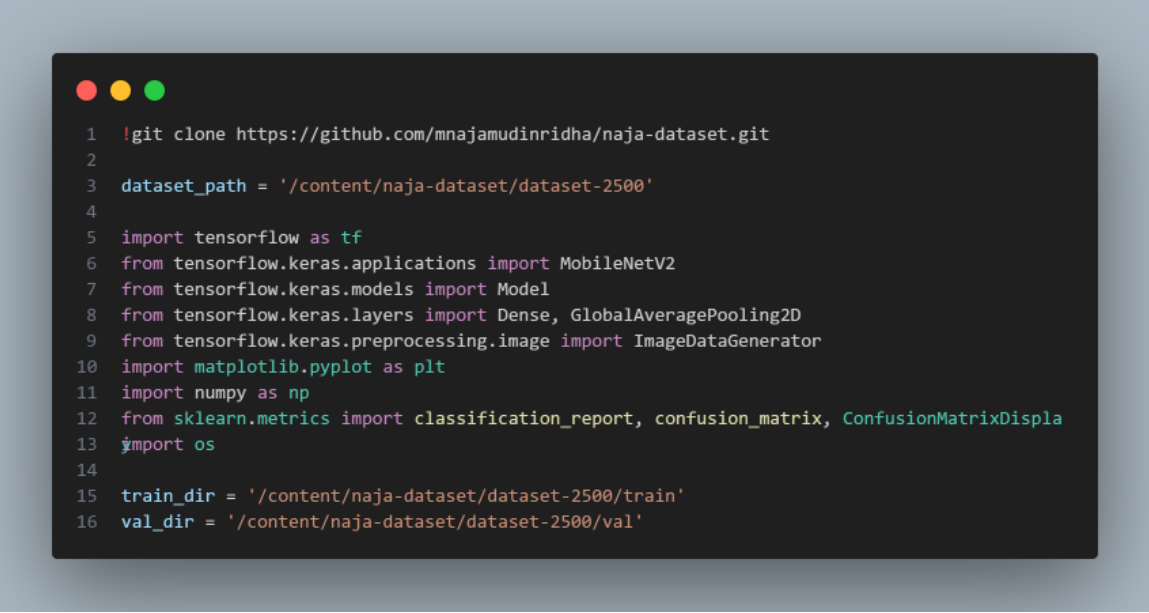


Laporan Proyek Klasifikasi Hijab vs NonHijab

1. Persiapan Dataset

- Dataset dikloning dari GitHub menggunakan perintah `git clone`.
- Direktori dataset disiapkan dalam format yang sesuai untuk `ImageDataGenerator`.
- Dataset dipisahkan menjadi folder `train/` dan `val/` dengan struktur subfolder per kelas.



```
1 !git clone https://github.com/mnajamudinridha/naja-dataset.git
2
3 dataset_path = '/content/naja-dataset/dataset-2500'
4
5 import tensorflow as tf
6 from tensorflow.keras.applications import MobileNetV2
7 from tensorflow.keras.models import Model
8 from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
9 from tensorflow.keras.preprocessing.image import ImageDataGenerator
10 import matplotlib.pyplot as plt
11 import numpy as np
12 from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDispla
13 import os
14
15 train_dir = '/content/naja-dataset/dataset-2500/train'
16 val_dir = '/content/naja-dataset/dataset-2500/val'
```

2. Preprocessing dan Augmentasi

- Gambar di-rescale ke $[0,1]$ menggunakan `rescale=1./255`.
- Augmentasi dilakukan dengan `rotation_range` dan `horizontal_flip` untuk data training.
- Data validation hanya dilakukan rescaling.

```

1  IMG_SIZE = (224, 224)
2  BATCH_SIZE = 32
3
4  train_datagen = ImageDataGenerator(rescale=1./255,
5                                     rotation_range=20,
6                                     horizontal_flip=True)
7
8  val_datagen = ImageDataGenerator(rescale=1./255)
9
10 train_generator = train_datagen.flow_from_directory(
11     train_dir,
12     target_size=IMG_SIZE,
13     batch_size=BATCH_SIZE,
14     class_mode='binary'
15 )
16
17 val_generator = val_datagen.flow_from_directory(
18     val_dir,
19     target_size=IMG_SIZE,
20     batch_size=BATCH_SIZE,
21     class_mode='binary'
22 )

```

3. Arsitektur Model (TensorFlow)

- Model dasar menggunakan MobileNetV2 tanpa top layer.
- Layer tambahan: GlobalAveragePooling2D → Dense(128) → Dense(1, sigmoid).
- Model dikompilasi menggunakan `adam`, loss `binary_crossentropy`, dan metrik `accuracy`.

```

1  base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224,
2  base_model.trainable = False
3
4  x = base_model.output
5  x = GlobalAveragePooling2D()(x)
6  x = Dense(128, activation='relu')(x)
7  output = Dense(1, activation='sigmoid')(x)
8
9  model = Model(inputs=base_model.input, outputs=output)
10 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

```

4. Pelatihan Model

- Model dilatih selama 10 epoch menggunakan `model.fit()`.
- History akurasi dan loss disimpan untuk visualisasi.
- Model disimpan dalam format `.keras`.

```
1 history = model.fit(  
2     train_generator,  
3     validation_data=val_generator,  
4     epochs=10  
5 )  
6  
7 model.save('naja-dataset.keras')  
8  
9 print("Model saved as naja-dataset.keras")
```

5. Evaluasi Model

- Menggunakan confusion matrix dan classification report.
- Visualisasi akurasi dan loss per epoch menggunakan matplotlib.

```
1 val_generator.reset()  
2 preds = model.predict(val_generator)  
3 y_pred = (preds > 0.5).astype(int).flatten()  
4 y_true = val_generator.classes  
5 labels = list(val_generator.class_indices.keys())  
6  
7 cm = confusion_matrix(y_true, y_pred)  
8 ConfusionMatrixDisplay(cm, display_labels=labels).plot(cmap='Blue')  
9 plt.title('Confusion Matrix')  
10 plt.show()  
11  
12 print(classification_report(y_true, y_pred, target_names=labels))
```

```
1 # Ambil loss dari history  
2 train_losses = history.history['loss']  
3 test_losses = history.history['val_loss']  
4 epochs = len(train_losses)  
5 epochs_range = range(1, epochs + 1)  
6  
7 # Visualisasi akurasi  
8 plt.plot(history.history['accuracy'], label='Training Accuracy')  
9 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')  
10 plt.title('Accuracy per Epoch')  
11 plt.xlabel('Epoch')  
12 plt.ylabel('Accuracy')  
13 plt.legend()  
14 plt.grid(True)  
15 plt.show()  
16  
17  
18 plt.figure(figsize=(10, 5))  
19 plt.plot(epochs_range, train_losses, label='Train Loss', marker='o')  
20 plt.plot(epochs_range, test_losses, label='Validation Loss', marker='o')  
21 plt.xlabel('Epoch')  
22 plt.ylabel('Loss')  
23 plt.title('Train vs Validation Loss per Epoch')  
24 plt.legend()  
25 plt.grid(True)  
26 plt.tight_layout()  
27 plt.show()
```

6. Prediksi Gambar

- Fungsi prediksi gambar disediakan menggunakan `predict_single_image()`.
- Visualisasi hasil prediksi pada beberapa gambar.
- Perbandingan antara prediksi dan label asli juga divisualisasikan.

```
1 import tensorflow as tf
2 from PIL import Image
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 # Load the saved Keras model
7 model = tf.keras.models.load_model('naja-dataset.keras')
8
9 class_names = ['Hijab', 'NonHijab']
10
11 img_path = [
12     '/content/5.jpg',
13     '/content/1.jpg',
14     '/content/2.jpg',
15     '/content/3.jpg',
16 ]
17
18
19
20
21 # Use the predict_single_image function for Keras models
22 def predict_single_image(img_path, model, class_names):
23     img = tf.keras.preprocessing.image.load_img(img_path, target_size=(224, 224))
24     img_array = tf.keras.preprocessing.image.img_to_array(img)
25     img_array = np.expand_dims(img_array, axis=0)
26     img_array /= 255.0
27
28     prediction = model.predict(img_array)[0][0] # float: 0.0 ~ 1.0
29     predicted_class_index = int(prediction > 0.5)
30     predicted_label = class_names[predicted_class_index]
31
32     print(f"Predicted: {predicted_label} | Confidence: {prediction:.2f}")
33     return predicted_label, img
34
35
36     plt.imshow(image_to_show)
37     plt.axis('off')
38     plt.title(f"Prediction: {predicted_label}", fontsize=16)
39     plt.tight_layout()
40     plt.show()
41
42
43     print(f"Predicted: {predicted_label} | Confidence: {prediction:.2f}")
44     return predicted_label, img
45
46 plt.figure(figsize=(16, 4)) # Atur ukuran keseluruhan figure
47
48 for i, path in enumerate(img_path):
49     predicted_label, image_to_show = predict_single_image(path, model, class_name
50 s)
51     plt.subplot(1, len(img_path), i + 1) # Buat grid horizontal
52     plt.imshow(image_to_show)
53     plt.axis('off')
54     plt.title(f"{predicted_label}", fontsize=10)
55
56 plt.tight_layout()
57 plt.show()
58
```

```
1/1 ----- 1s 1s/step
Predicted: NonHijab | Confidence: 1.00
1/1 ----- 0s 79ms/step
Predicted: Hijab | Confidence: 0.00
1/1 ----- 0s 94ms/step
Predicted: Hijab | Confidence: 0.00
1/1 ----- 0s 92ms/step
Predicted: Hijab | Confidence: 0.00
```

