LAMPIRAN

Kode Program:

Lampiran 1 Source Code import library

```
# train.py
import os
import sys
import tensorflow as tf
from src.data_utils import create_data_generators
from src.model_utils import build_efficientnet_model,
  build_vgg16_model, train_model
print("TensorFlow version:", tf.__version__)
print("Python version:", sys.version)
# Create models directory if it doesn't exist
os.makedirs('models', exist_ok=True)
# Set image size and batch size
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 15
# Create data generators
print("Creating data generators...")
DATA_DIR = 'data'
train_generator, valid_generator = create_data_generators(
DATA_DIR,
img_size=IMG_SIZE,
batch_size=BATCH_SIZE
)
# Train EfficientNet model
print("Building and training EfficientNet model...")
efficientnet_model =
  build_efficientnet_model(img_size=IMG_SIZE)
efficientnet_history, trained_efficientnet = train_model(
efficientnet_model,
train_generator,
valid_generator,
model_name='efficientnet',
epochs=EPOCHS
)
# Train VGG16 model
print("Building and training VGG16 model...")
vgg16_model = build_vgg16_model(img_size=IMG_SIZE)
vgg16_history, trained_vgg16 = train_model(
vgg16_model,
train_generator,
valid_generator,
```

```
40. model_name='vgg16',
41. epochs=EPOCHS
42. )
43. print("Training complete! Models saved in 'models'
    directory.")
```

Lampiran 2 Source Code APP.PY

```
1.  from flask import Flask, request, jsonify
2.  import os
3.  import numpy as np
4.  import tensorflow as tf
5.  from pyngrok import ngrok
6.  import time
7.  from certificate_extractor import CertificateExtractor,
    CertificateVerifier
8.  app = Flask(__name__)
9.  # Path model
10. EFFICIENTNET_MODEL = os.path.join('models',
    'efficientnet_best.h5')
11. VGG16_MODEL = os.path.join('models', 'vgg16_best.h5')
12. # Load model sekali saat startup
13. efficientnet_model = None
14. vgg16_model = None
15. if os.path.exists(EFFICIENTNET_MODEL):
16. efficientnet_model =
    tf.keras.models.load_model(EFFICIENTNET_MODEL)
17. print("Loaded EfficientNet model")
18. if os.path.exists(VGG16_MODEL):
19. vgg16_model = tf.keras.models.load_model(VGG16_MODEL)
20. print("Loaded VGG16 model")
21. def convert_numpy_types(obj):
22. """Convert NumPy types to native Python types for JSON
    serialization"""
23. if isinstance(obj, np.integer):
24. return int(obj)
25. elif isinstance(obj, np.floating):
26. return float(obj)
27. elif isinstance(obj, np.bool_):
28. return bool(obj)
29. elif isinstance(obj, np.ndarray):
30. return obj.tolist()
31. elif isinstance(obj, dict):
32. return {key: convert_numpy_types(value) for key, value in
    obj.items()}
33. elif isinstance(obj, list):
34. return [convert_numpy_types(item) for item in obj]
35. else:
36. return obj
```

```
37. def preprocess_image_for_prediction(image_path,
    target_size=(224,224)):
38. from PIL import Image
39. img = Image.open(image_path).convert('RGB')
40. img = img.resize(target_size)
41. img_array = np.array(img) / 255.0
42. img_array = np.expand_dims(img_array, axis=0)
43. return img_array
44. @app.route('/')
45. def index():
46. return "Certificate Verification API"
47. @app.route('/verify', methods=['POST'])
48. def verify_certificate():
49. if 'image' not in request.files:
50. return jsonify({'error': 'No image uploaded'}), 400
51. file = request.files['image']
52. if file.filename == '':
53. return jsonify({'error': 'Empty file'}), 400
54. temp_path = 'temp_certificate.jpg'
55. file.save(temp_path)
56. try:
```

Lampiran 3 Source Code Prepare_For_Deployment

```
1.  # prepare_for_deployment.py
2.  import os
3.  import shutil
4.  # Create deployment directory
5.  DEPLOYMENT_DIR = 'deployment'
6.  os.makedirs(DEPLOYMENT_DIR, exist_ok=True)
7.  os.makedirs(os.path.join(DEPLOYMENT_DIR, 'models'),
    exist_ok=True)
8.  # Source models
9.  EFFICIENTNET_MODEL = 'models/efficientnet_final.h5'
10. VGG16_MODEL = 'models/vgg16_final.h5'
11. # Copy models to deployment directory
12. if os.path.exists(EFFICIENTNET_MODEL):
13. shutil.copy(EFFICIENTNET_MODEL, os.path.join(DEPLOYMENT_DIR,
    'models', 'efficientnet.h5'))
14. print(f"Copied {EFFICIENTNET_MODEL} to deployment directory")
15. else:
16. print(f"Warning: {EFFICIENTNET_MODEL} not found")
17. if os.path.exists(VGG16_MODEL):
18. shutil.copy(VGG16_MODEL, os.path.join(DEPLOYMENT_DIR,
    'models', 'vgg16.h5'))
19. print(f"Copied {VGG16_MODEL} to deployment directory")
20. else:
21. print(f"Warning: {VGG16_MODEL} not found")
22. # Copy server requirements file
```

```
23. shutil.copy('server_requirements.txt',
    os.path.join(DEPLOYMENT_DIR, 'requirements.txt'))
24. # Create Flask API file content
25. flask_api = '''
26. # app.py
27. from flask import Flask, request, jsonify
28. import os
29. import numpy as np
30. import tensorflow as tf
31. app = Flask(__name__)
32. # Load models
33. EFFICIENTNET_MODEL = os.path.join('models', 'efficientnet.h5')
34. VGG16_MODEL = os.path.join('models', 'vgg16.h5')
35. # Check if models exist
36. efficientnet_model = None
37. vgg16_model = None
38. if os.path.exists(EFFICIENTNET_MODEL):
39. efficientnet_model =
    tf.keras.models.load_model(EFFICIENTNET_MODEL)
40. print("Loaded EfficientNet model")
41. if os.path.exists(VGG16_MODEL):
42. vgg16_model = tf.keras.models.load_model(VGG16_MODEL)
43. print("Loaded VGG16 model")
44. @app.route('/')
45. def index():
46. return "Certificate Verification API"
47. @app.route('/verify', methods=['POST'])
48. def verify_certificate():
49. # Check if image was uploaded
50. if 'image' not in request.files:
51. return jsonify({'error': 'No image uploaded'})
52. file = request.files['image']
53. # Check if file is empty
54. if file.filename == '':
55. return jsonify({'error': 'Empty file'})
56. # Save the uploaded file temporarily
57. temp_path = 'temp_certificate.jpg'
58. file.save(temp_path)
59. try:
60. # Preprocess the image
61. img = tf.keras.preprocessing.image.load_img(temp_path,
    target_size=(224, 224))
62. img_array = tf.keras.preprocessing.image.img_to_array(img)
63. img_array = img_array / 255.0
64. img_array = np.expand_dims(img_array, axis=0)
65. # Get predictions
66. results = {}
67. if efficientnet_model is not None:
68. efficientnet_pred =
    float(efficientnet_model.predict(img_array)[0][0])
69. results['efficientnet_probability'] = efficientnet_pred
```

```
if vgg16_model is not None:
vgg16_pred = float(vgg16_model.predict(img_array)[0][0])
results['vgg16_probability'] = vgg16_pred
# Combine predictions if both models are available
if efficientnet_model is not None and vgg16_model is not None:
combined_pred = (results['efficientnet_probability'] +
results['vgg16_probability']) / 2.0
results['combined_probability'] = combined_pred
results['is_authentic'] = combined_pred < 0.5  # Assuming 0 is
authentic, 1 is fake
elif efficientnet_model is not None:
results['combined_probability'] =
results['efficientnet_probability']
results['is_authentic'] = results['efficientnet_probability']
< 0.5
elif vgg16_model is not None:
results['combined_probability'] = results['vgg16_probability']
results['is_authentic'] = results['vgg16_probability'] < 0.5
else:
return jsonify({'error': 'No models available'})
# Add a textual interpretation
if results['is_authentic']:
results['result'] = "Certificate appears to be authentic"
else:
results['result'] = "Certificate appears to be fake"
# Clean up
if os.path.exists(temp_path):
os.remove(temp_path)
return jsonify(results)
except Exception as e:
# Clean up on error
if os.path.exists(temp_path):
os.remove(temp_path)
return jsonify({'error': str(e)})
if __name__ == '__main__':
app.run(debug=False, host='0.0.0.0', port=5000)
'''
# Write Flask API to file
with open(os.path.join(DEPLOYMENT_DIR, 'app.py'), 'w') as f:
f.write(flask_api)

print(f"Deployment files prepared in '{DEPLOYMENT_DIR}'
directory.")
```

Lampiran 4 Source Code Train.py

```
# train.py
import os
import sys
import tensorflow as tf
from src.data_utils import create_data_generators
```

```
from src.model_utils import build_efficientnet_model,
build_vgg16_model, train_model
print("TensorFlow version:", tf.__version__)
print("Python version:", sys.version)
# Create models directory if it doesn't exist
os.makedirs('models', exist_ok=True)
# Set image size and batch size
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 15
# Create data generators
print("Creating data generators...")
DATA_DIR = 'data'
train_generator, valid_generator = create_data_generators(
DATA_DIR,
img_size=IMG_SIZE,
batch_size=BATCH_SIZE
)
# Train EfficientNet model
print("Building and training EfficientNet model...")
efficientnet_model =
build_efficientnet_model(img_size=IMG_SIZE)
efficientnet_history, trained_efficientnet = train_model(
efficientnet_model,
train_generator,
valid_generator,
model_name='efficientnet',
epochs=EPOCHS
)
# Train VGG16 model
print("Building and training VGG16 model...")
vgg16_model = build_vgg16_model(img_size=IMG_SIZE)
vgg16_history, trained_vgg16 = train_model(
vgg16_model,
train_generator,
valid_generator,
model_name='vgg16',
epochs=EPOCHS
)
print("Training complete! Models saved in 'models'
directory.")
```