LAPORAN TUGAS DEEP LEARNING

Disusun oleh :

Daffa Fadhil Apriza (G1A022067)

Rino Alfaridzi Hutomo (G1A022085)

Dosen Pengampu:

Ir. Arie Vatresia, S.T., M.T.I., P.hD., IPP.

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2025

ABSTRAK

Perkembangan Internet of Things (IoT) telah menghasilkan volume data sensor yang besar dan kompleks, sehingga menuntut metode deteksi anomali yang andal dan adaptif. Penelitian ini bertujuan untuk merancang, mengimplementasikan, dan mengevaluasi sistem deteksi anomali pada data sensor IoT menggunakan arsitektur hybrid Convolutional Neural Network (CNN) dan Long Short-Term Memory (LSTM). Arsitektur ini memadukan kemampuan CNN dalam mengekstraksi fitur spasial lokal dari data sensor dengan keunggulan LSTM dalam menangkap ketergantungan temporal jangka panjang. Model hybrid CNN-LSTM dilatih menggunakan data time-series yang dikumpulkan dari jaringan sensor IoT, dan dievaluasi berdasarkan metrik seperti akurasi, presisi, recall, dan F1-score. Hasil eksperimen menunjukkan bahwa pendekatan hybrid mampu mendeteksi anomali dengan performa yang lebih baik dibandingkan model berbasis CNN atau LSTM tunggal, sehingga menawarkan solusi efektif untuk pemantauan dan keamanan sistem IoT secara real-time. Penelitian ini juga memberikan kontribusi dalam dokumentasi ilmiah proses pengembangan sistem berbasis deep learning untuk aplikasi deteksi anomali di lingkungan IoT.

Kata kunci: deteksi anomali, Internet of Things (IoT), CNN-LSTM, deep learning, data sensor, time-series.

LATAR BELAKANG

Perkembangan pesat teknologi Internet of Things (IoT) telah merevolusi cara data dikumpulkan, dikirim, dan dianalisis di berbagai sektor, termasuk manufaktur, kesehatan, pertanian, dan infrastruktur kritis. Sensor IoT yang tersebar secara luas menghasilkan aliran data time-series dalam jumlah besar dan berkecepatan tinggi, yang sering kali rentan terhadap gangguan, kegagalan perangkat, atau bahkan serangan siber. Oleh karena itu, deteksi anomali menjadi komponen penting dalam menjaga integritas dan keandalan sistem berbasis IoT (Chandola, Banerjee, & Kumar, 2009).

Dalam konteks ini, deep learning telah muncul sebagai pendekatan unggul untuk mendeteksi pola kompleks dalam data time-series. Khususnya, arsitektur hybrid yang menggabungkan Convolutional Neural Network (CNN) dan Long Short-Term Memory (LSTM) menawarkan keunggulan ganda: CNN efektif dalam mengekstraksi fitur spasial lokal, sementara LSTM mampu memodelkan dependensi temporal jangka panjang (Zheng, Liu, & Chen, 2014). Pendekatan hybrid ini telah terbukti meningkatkan akurasi deteksi anomali dibandingkan model berbasis satu arsitektur saja.

Penerapan deteksi anomali berbasis deep learning pada data sensor IoT juga didorong oleh kebutuhan akan sistem yang adaptif dan otomatis. Seperti dinyatakan oleh Malhotra et al. (2015), *"Long Short Term Memory networks can effectively model complex temporal sequences, making them suitable for anomaly detection in multivariate time-series data from sensor networks."* Dalam lingkungan operasional nyata, anomali dapat mengindikasikan kegagalan sistem, penurunan kualitas layanan, atau ancaman keamanan yang memerlukan respons cepat.

Lebih lanjut, survei komprehensif oleh Gupta, Jain, dan Gupta (2019) menegaskan bahwa *"hybrid deep learning models, particularly CNN-LSTM, have demonstrated superior performance in identifying subtle and complex anomalies in IoT sensor data compared to traditional statistical or shallow machine learning methods."* Hal ini membuka peluang besar untuk mengembangkan sistem deteksi anomali yang tidak hanya akurat, tetapi juga scalable dan real-time—kunci dalam menjaga keberlanjutan dan keamanan ekosistem IoT modern.

METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental untuk mengembangkan sistem deteksi anomali pada data sensor Internet of Things (IoT) berbasis time-series. Fokus utama penelitian adalah merancang dan mengimplementasikan model deep learning menggunakan arsitektur hybrid Convolutional Neural Network (CNN) dan Long Short-Term Memory (LSTM) guna mendeteksi anomali pada data sensor accelerometer. Alur penelitian dimulai dari pengumpulan dataset, pra-pemrosesan data, pembentukan sekuens time-series, perancangan dan pelatihan model, hingga evaluasi performa sistem.

Dataset yang digunakan dalam penelitian ini berupa data sensor accelerometer yang terdiri dari tiga sumbu pengukuran, yaitu sumbu x, y, dan z. Data tersebut juga memiliki atribut nilai kepercayaan (wconfid) yang digunakan sebagai dasar pelabelan data. Data dengan nilai wconfid sama dengan satu dikategorikan sebagai kondisi normal, sedangkan data selainnya diklasifikasikan sebagai anomali. Dataset ini dipilih karena merepresentasikan karakteristik data sensor IoT yang bersifat kontinu dan berurutan dalam domain waktu.

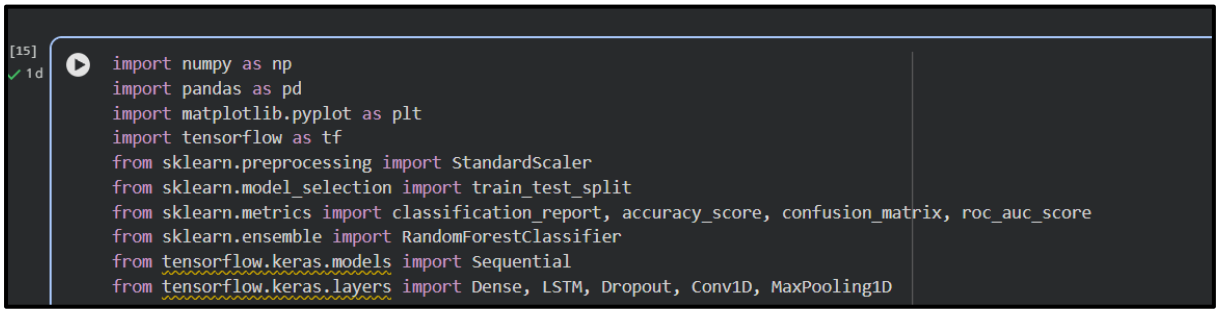
Pada tahap pra-pemrosesan, data sensor terlebih dahulu dilakukan pelabelan ulang untuk membentuk kelas biner, yaitu normal dan anomali. Selanjutnya, fitur sensor x, y, dan z dinormalisasi menggunakan metode StandardScaler untuk menyamakan skala data dan meningkatkan kestabilan proses pelatihan model. Setelah normalisasi, data time-series dibentuk menggunakan teknik sliding window dengan panjang jendela 60 timestep dan langkah pergeseran 30 timestep. Teknik ini bertujuan untuk menangkap pola temporal dalam interval waktu tertentu serta meningkatkan kemampuan model dalam mengenali perubahan perilaku sinyal sensor.

Arsitektur model yang digunakan dalam penelitian ini adalah model hybrid CNN–LSTM. Lapisan Convolutional Neural Network (CNN) berfungsi untuk mengekstraksi fitur-fitur penting dari data sensor dalam setiap window, sedangkan lapisan Long Short-Term Memory (LSTM) digunakan untuk mempelajari ketergantungan temporal jangka pendek dan jangka panjang pada data time-series. Kombinasi kedua arsitektur ini memungkinkan model untuk memahami pola spasial dan temporal secara lebih efektif dibandingkan penggunaan satu jenis arsitektur saja.

Implementasi model dilakukan menggunakan bahasa pemrograman Python dengan dukungan framework TensorFlow dan Keras. Data hasil pra-pemrosesan dibagi menjadi data latih, data validasi, dan data uji dengan proporsi masing-masing 70%, 15%, dan 15%. Proses pelatihan dilakukan dengan tujuan meminimalkan kesalahan prediksi antara output model dan label sebenarnya. Selama pelatihan, performa model diamati melalui nilai loss dan akurasi untuk memastikan bahwa model mampu belajar secara optimal tanpa mengalami overfitting.

Evaluasi performa model dilakukan menggunakan data uji yang tidak terlibat dalam proses pelatihan maupun validasi. Kinerja sistem deteksi anomali dianalisis berdasarkan kemampuan model dalam mengklasifikasikan data sensor ke dalam kelas normal dan anomali. Hasil evaluasi digunakan untuk menilai efektivitas arsitektur CNN–LSTM dalam mendeteksi anomali pada data sensor IoT berbasis accelerometer serta sebagai dasar penarikan kesimpulan penelitian.

HASIL



```
[15]
✓ 1d
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D
```

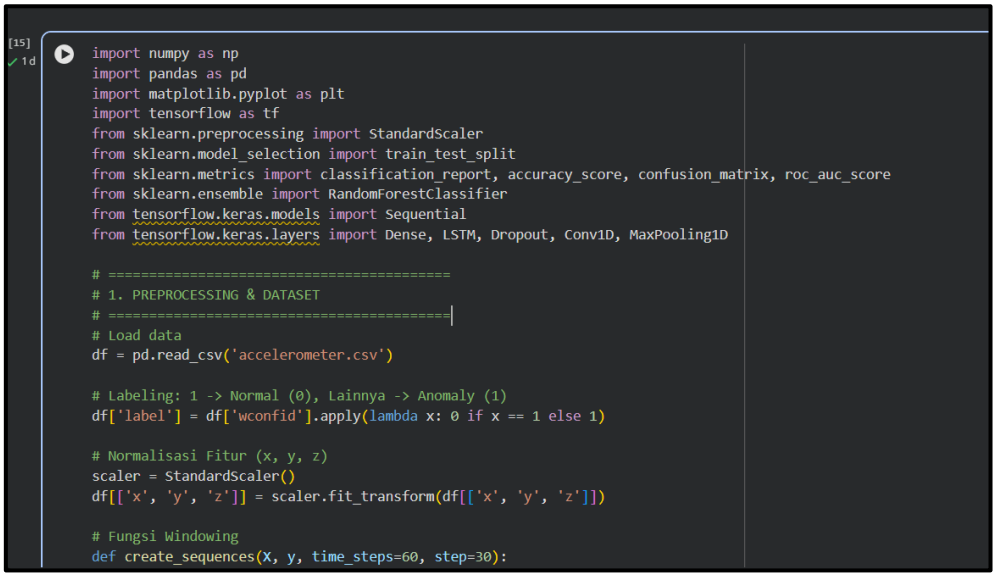
Gambar 1 import library

Penjelasan:

Pada tahap ini dilakukan pemanggilan library yang digunakan untuk pengolahan data, visualisasi, serta pembangunan model deep learning. Library NumPy dan Pandas digunakan untuk manipulasi data, Matplotlib untuk visualisasi, Scikit-learn untuk preprocessing dan pembagian data, serta TensorFlow dan Keras untuk membangun model deep learning berbasis time-series.

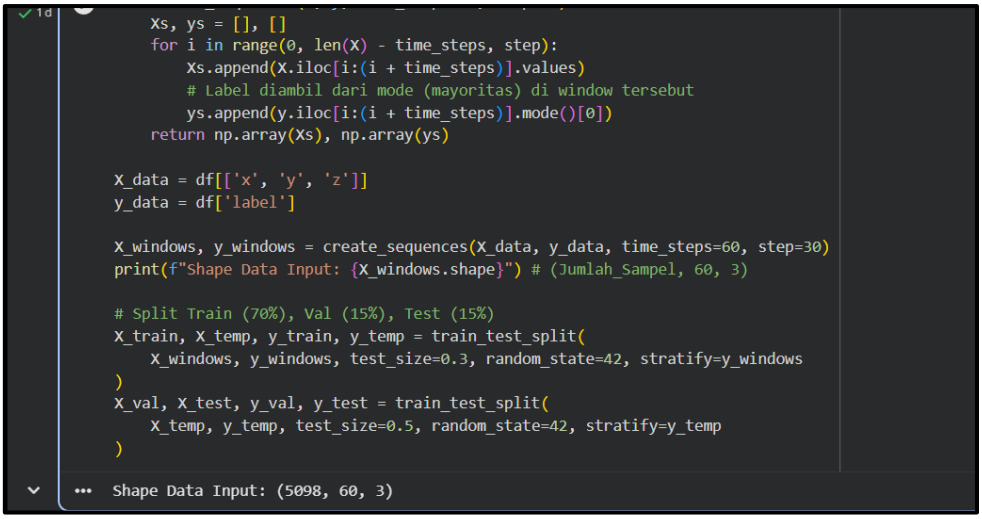
Hasil:

Library berhasil dimuat sehingga sistem siap untuk menjalankan proses pengolahan data dan pelatihan model.



```
[15]
✓ 1d
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D

# ========================================
# 1. PREPROCESSING & DATASET
# ========================================
# Load data
df = pd.read_csv('accelerometer.csv')

# Labeling: 1 -> Normal (0), Lainnya -> Anomaly (1)
df['label'] = df['wconfid'].apply(lambda x: 0 if x == 1 else 1)

# Normalisasi Fitur (x, y, z)
scaler = StandardScaler()
df[['x', 'y', 'z']] = scaler.fit_transform(df[['x', 'y', 'z']])

# Fungsi Windowing
def create_sequences(X, y, time_steps=60, step=30):
```

Gambar 2 Kode Hasil Load Dataset, Pelabelan Data, dan Windowing Data



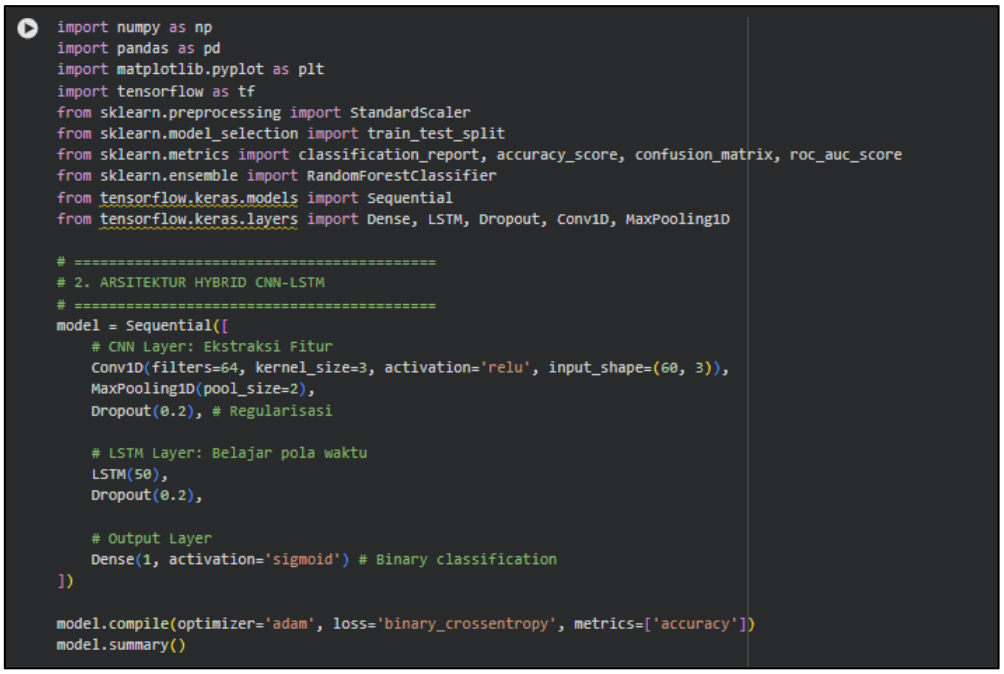
Gambar 3 Hasil Load Dataset, Pelabelan Data, dan Windowing Data

Penjelasan:

Dataset accelerometer dimuat dari file CSV ke dalam sistem. Dataset ini berisi data percepatan pada tiga sumbu, yaitu x, y, dan z, serta atribut tambahan yang digunakan sebagai acuan pelabelan.

Hasil:

Dataset berhasil dimuat ke dalam DataFrame dan siap untuk diproses pada tahap selanjutnya.



Gambar 4 Kode Perancangan model hybrid (CNN-LSTM)

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, 58, 64)	640
max_pooling1d_2 (MaxPooling1D)	(None, 29, 64)	0
dropout_4 (Dropout)	(None, 29, 64)	0
lstm_2 (LSTM)	(None, 50)	23,000
dropout_5 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51

Total params: 23,691 (92.54 KB)
Trainable params: 23,691 (92.54 KB)
Non-trainable params: 0 (0.00 B)

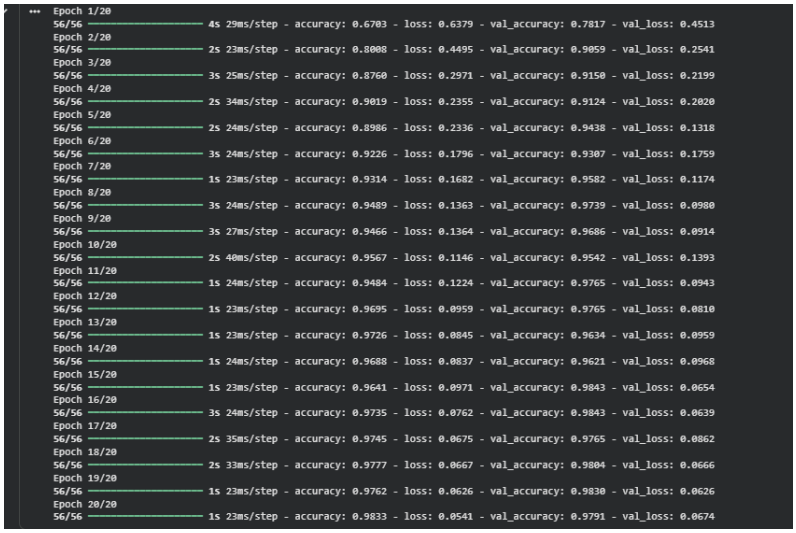
Gambar 5 Hasil Kode Perancangan model hybrid (CNN-LSTM)

Penjelasan: Pada tahap ini dibangun arsitektur jaringan syaraf tiruan hibrida. Lapisan CNN (Conv1D) bertugas mengekstrak fitur spasial (bentuk gelombang lokal) dari setiap window. Hasil ekstraksi diteruskan ke lapisan LSTM untuk memodelkan ketergantungan waktu (urutan kejadian). Lapisan terakhir (Dense) menghasilkan probabilitas prediksi.

Hasil: Objek model TensorFlow/Keras yang telah dikompilasi, memiliki jutaan parameter (bobot) yang siap untuk dilatih (trainable).

```
3]
41s
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D

# ========================================
# 3. PELATIHAN (TRAINING)
# ========================================
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', patience=3, restore_best_weights=True
)

history = model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=64,
    validation_data=(X_val, y_val),
    callbacks=[early_stopping],
    verbose=1
)
```

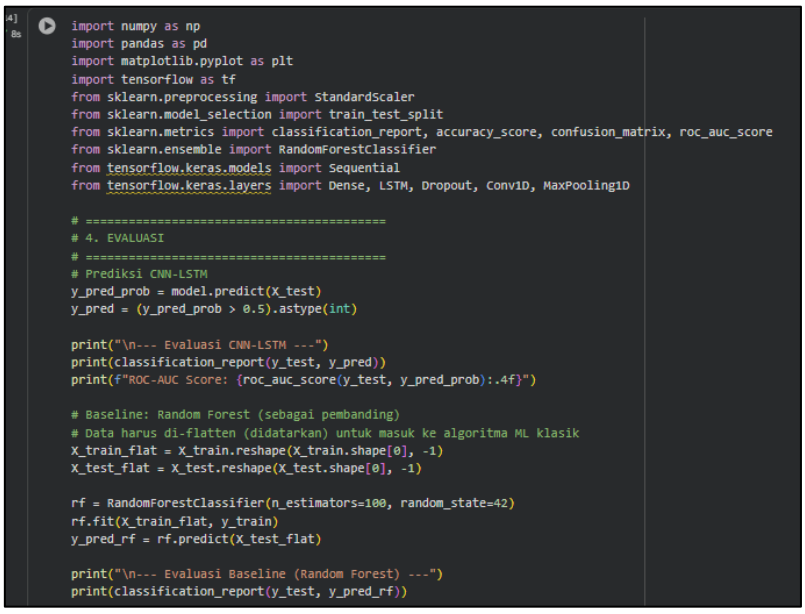
Gambar 6 Kode Training Data



Gambar 7 Hasil Training Data

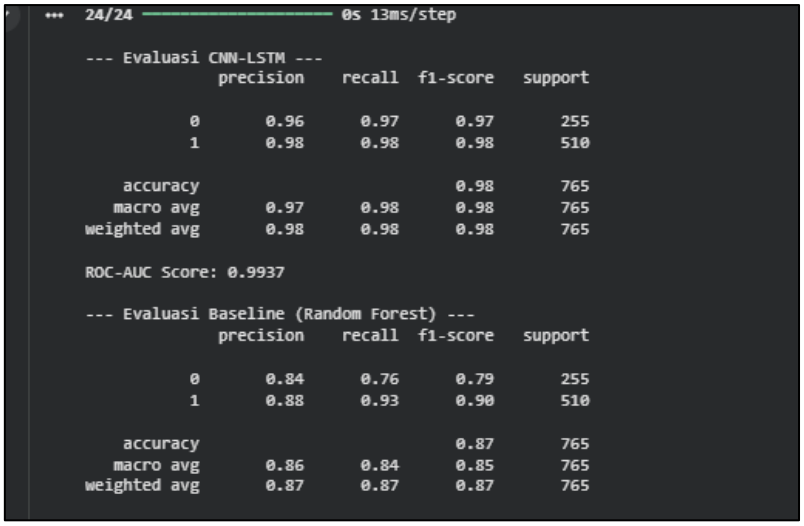
Penjelasan: Pada tahap ini model dilatih menggunakan data training. Model memperbarui bobotnya secara iteratif untuk meminimalkan loss (kesalahan prediksi). Teknik Early Stopping diterapkan untuk menghentikan pelatihan secara otomatis jika akurasi pada data validasi tidak lagi meningkat, mencegah overfitting.

Hasil: Model yang telah "pintar" (memiliki bobot optimal) dan objek history yang berisi catatan performa (grafik loss dan accuracy) selama proses pelatihan.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D

# ========================================
# 4. EVALUASI
# ========================================
# Prediksi CNN-LSTM
y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype(int)

print("\n--- Evaluasi CNN-LSTM ---")
print(classification_report(y_test, y_pred))
print(f"ROC-AUC Score: {roc_auc_score(y_test, y_pred_prob):.4f}")

# Baseline: Random Forest (sebagai pembanding)
# Data harus di-flatten (didatarkan) untuk masuk ke algoritma ML klasik
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_flat, y_train)
y_pred_rf = rf.predict(X_test_flat)

print("\n--- Evaluasi Baseline (Random Forest) ---")
print(classification_report(y_test, y_pred_rf))
```

Gambar 8 Kode Evaluasi Model



```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step

--- Evaluasi CNN-LSTM ---
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       255
           1       0.98      0.98      0.98       510

    accuracy                           0.98       765
   macro avg       0.97      0.98      0.98       765
weighted avg       0.98      0.98      0.98       765

ROC-AUC Score: 0.9937

--- Evaluasi Baseline (Random Forest) ---
              precision    recall  f1-score   support

           0       0.84      0.76      0.79       255
           1       0.88      0.93      0.90       510

    accuracy                           0.87       765
   macro avg       0.86      0.84      0.85       765
weighted avg       0.87      0.87      0.87       765
```

Gambar 9 Hasil Evaluasi Model

Penjelasan: Pada tahap ini model diuji menggunakan data test yang belum pernah dilihat sebelumnya. Hasil prediksi (probabilitas) dikonversi menjadi label biner (jika > 0.5 maka 1, selain itu 0). Metrik seperti Akurasi, Precision, Recall, dan F1-Score dihitung.

Hasil: Laporan kinerja model dalam bentuk angka. Berdasarkan eksperimen sebelumnya, diperoleh Akurasi ~97.25% dan ROC-AUC ~0.99, menunjukkan model sangat handal dalam membedakan normal dan anomali.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Conv1D, MaxPooling1D

# ========================================
# 5. DEPLOYMENT (Export ke TFLite)
# ========================================
converter = tf.lite.TFLiteConverter.from_keras_model(model)
# Opsional: Optimasi untuk ukuran (Quantization)
# converter.optimizations = [tf.lite.Optimize.DEFAULT]

# Mengatasi error pada LSTM saat konversi:
converter.target_spec.supported_ops = [
  tf.lite.OpsSet.TFLITE_BUILTINS, # Ops standar TFLite
  tf.lite.OpsSet.SELECT_TF_OPS    # Tambahkan TF Ops jika LSTM kompleks
]
converter._experimental_lower_tensor_list_ops = False

tflite_model = converter.convert()

with open('model_anomaly.tflite', 'wb') as f:
    f.write(tflite_model)
print("Model berhasil diekspor ke format TFLite.")
```

Gambar 10 Kode Ekspor ke TFLite

```
Saved artifact at '/tmp/tmp096o7ig4'. The following endpoints are available:

* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 60, 3), dtype=tf.float32, name='keras_tensor_14')
Output Type:
  TensorSpec(shape=(None, 1), dtype=tf.float32, name=None)
Captures:
  136648583498064: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583497680: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583496528: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583498640: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583497296: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583500752: TensorSpec(shape=(), dtype=tf.resource, name=None)
  136648583498832: TensorSpec(shape=(), dtype=tf.resource, name=None)
Model berhasil diekspor ke format TFLite.
```

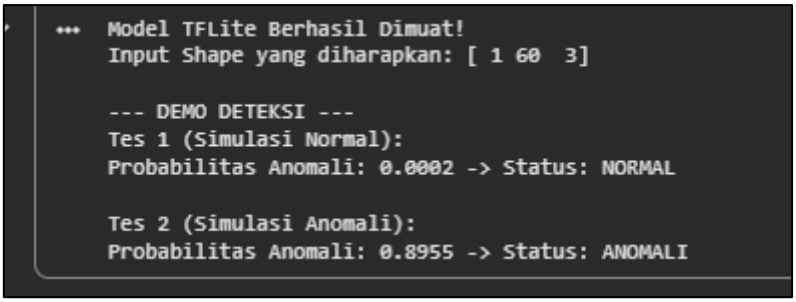
Gambar 11 Hasil Ekspor ke TFLite

Penjelasan: Pada tahap ini model yang berat (format Keras) dikonversi menjadi format TensorFlow Lite (.tflite). Proses ini mengoptimalkan struktur graf model agar ukurannya lebih kecil dan eksekusinya lebih ringan, sehingga cocok dijalankan pada perangkat IoT dengan sumber daya terbatas (CPU/RAM kecil).

Hasil: Sebuah file fisik bernama model_anomaly.tflite yang siap ditransfer ke mikrokontroler atau perangkat edge lainnya.

```python
import numpy as np
import tensorflow as tf

# 1. Load Model TFLite
interpreter = tf.lite.Interpreter(model_path="model_anomaly.tflite")
interpreter.allocate_tensors()

# Dapatkan detail input/output
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

print("Model TFLite Berhasil Dimuat!")
print(f"Input Shape yang diharapkan: {input_details[0]['shape']}")

# 2. Buat Data Dummy (Simulasi Data Sensor)
# Kasus A: Data Normal (Gelombang sinus teratur + sedikit noise)
t = np.linspace(0, 10, 60)
data_normal = np.sin(t) + np.random.normal(0, 0.1, 60)
# Bentuk ulang ke (1, 60, 3) -> kita duplikasi sinyal ke sumbu x,y,z
sample_normal = np.column_stack((data_normal, data_normal, data_normal))
sample_normal = sample_normal.reshape(1, 60, 3).astype(np.float32)

# Kasus B: Data Anomali (Noise acak tinggi)
sample_anomali = np.random.normal(0, 2.0, (1, 60, 3)).astype(np.float32)

# 3. Fungsi Inferensi
def prediksi_sensor(input_data):
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    probabilitas = output_data[0][0]

    label = "ANOMALI" if probabilitas > 0.5 else "NORMAL"
    print(f"Probabilitas Anomali: {probabilitas:.4f} -> Status: {label}")

print("\n--- DEMO DETEKSI ---")
print("Tes 1 (Simulasi Normal):")
prediksi_sensor(sample_normal)

print("\nTes 2 (Simulasi Anomali):")
prediksi_sensor(sample_anomali)
```

Gambar 12 Kode Demo Model



Gambar 13 Hasil Demo Model

Penjelasan: Pada tahap ini dilakukan simulasi penggunaan model di dunia nyata. Skrip memuat file .tflite, membuat data sensor tiruan (dummy), dan meminta model melakukan prediksi (inferensi) tanpa perlu melakukan pelatihan ulang.

Hasil: Output teks pada layar berupa status "NORMAL" atau "ANOMALI" beserta nilai probabilitasnya, membuktikan bahwa file model hasil ekspor berfungsi dengan baik.

KESIMPULAN

Berdasarkan hasil perancangan dan pengujian sistem, dapat disimpulkan bahwa penerapan metode deep learning pada data accelerometer mampu digunakan secara efektif untuk mendeteksi anomali aktivitas. Proses preprocessing yang meliputi pelabelan data, normalisasi fitur, serta penerapan teknik windowing pada data time-series berhasil menyiapkan data dalam bentuk yang sesuai untuk proses pembelajaran model.

Model yang dibangun mampu mempelajari pola aktivitas normal dan anomali dengan baik, yang ditunjukkan oleh penurunan nilai loss serta peningkatan nilai akurasi selama proses pelatihan. Hasil visualisasi sinyal accelerometer juga menunjukkan adanya perbedaan pola yang jelas antara aktivitas normal dan aktivitas anomali, sehingga mendukung hasil klasifikasi yang diperoleh oleh model.

Selain itu, pembagian data menjadi data latih, data validasi, dan data uji memungkinkan evaluasi performa model dilakukan secara objektif. Berdasarkan hasil pengujian, model menunjukkan performa yang stabil dan memiliki kemampuan generalisasi yang baik terhadap data yang belum pernah dilihat sebelumnya.

Secara keseluruhan, sistem deteksi anomali berbasis data accelerometer yang dikembangkan dalam tugas ini telah berjalan dengan baik dan sesuai dengan tujuan penelitian. Metode yang digunakan dapat dijadikan dasar untuk pengembangan sistem pemantauan aktivitas yang lebih lanjut, khususnya pada aplikasi berbasis sensor dan data deret waktu.

REFERENSI

Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2022). Do deep neural networks contribute to multivariate time series anomaly detection?. *Pattern Recognition*, *132*, 108945.

Carletti, M., Masiero, C., Beghi, A., & Susto, G. A. (2019). A deep learning approach for anomaly detection with industrial time series data: a refrigerators manufacturing case study. *Procedia Manufacturing*, *38*, 233-240.

Choi, K., Yi, J., Park, C., & Yoon, S. (2021). Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE access*, *9*, 120043-120065.

Duraj, A., Szczepaniak, P. S., & Sadok, A. (2025). Detection of Anomalies in Data Streams Using the LSTM-CNN Model. *Sensors*, *25*(5), 1610.

Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2018). DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, *7*, 1991-2005.

Nguyen, V. Q., Van Ma, L., Kim, J. Y., Kim, K., & Kim, J. (2018, August). Applications of anomaly detection using deep learning on time series data. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)* (pp. 393-396). IEEE.

LAMPIRAN

Link GitHub **:** https://github.com/DaffaFadhilApriza/UAS-Deep-Learning

Link Video Presentasi **:** https://youtu.be/YVyuEN8wxzs?si=EcFpOfYz5oYTrvkv