Lampiran 1. Daftar Riwayat Hidup Peneliti

DAFTAR RIWAYAT HIDUP

Nama Lengkap	: Lusindah Erdian Nova
NIM	: 14240010
Program Studi	: Informatika
Fakultas	: Teknologi Informasi
Jenjang	: Strata Dua (S2)
Peminatan	: Artificial Intelligence dan *Blockchain*
Tempat Tinggal	: DKI Jakarta, Indonesia
Email	: 14240010@nusamandiri.ac.id

RIWAYAT PENDIDIKAN

1. D3 Universitas Bina Sarana Informatika, Program Studi System Informasi, 2003-2006
2. S1 Universitas Nusa Mandiri, Program Studi System Informasi, 2012-2013
3. S2 Universitas Persada Indonesia Y.A.I, Program Studi Magister Manajemen, 2014–2016.
4. S2 Universitas Nusa Mandiri, Program Studi Informatika, 2024–2026.

PENGALAMAN KERJA

1. Operation Area Manager, PT. Tunas Berkat Semesta, 2017–2025
2. Dosen International Bussines, STIE Swadaya, 2016-2017
3. Shift Leader, PT. Sari Melati Kencana, 2011–2015.
4. Store Manager, PT. Perintis Pelayanan Paripurna, 2008–2011.



Jakarta, Agustus 2026

Lusindah Erdian Nova

Lampiran 2. Hasil Cek Plagiarisme



GRAPH ATTENTIVE CONVOLUTIONAL NETWORKS DALAM DETEKSI TRANSAKSI ILEGAL PADA JARINGAN BITCOIN UNTUK KEBUTUHAN DIGITAL FORENSIK

ORIGINALITY REPORT

25%	19%	17%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	123dok.com Internet Source	2%
2	repository.bsi.ac.id Internet Source	1%
3	Submitted to UPN Veteran Yogyakarta Student Paper	<1%
4	Submitted to Universitat Oberta de Catalunya Student Paper	<1%
5	repository.unissula.ac.id Internet Source	<1%
6	Submitted to Berlin School of Business and Innovation Student Paper	<1%
7	Submitted to Universitas Putera Batam Student Paper	<1%
8	repository.ubpkarawang.ac.id Internet Source	<1%
9	kc.umn.ac.id Internet Source	<1%
10	dokumen.tips Internet Source	<1%
11	repository.ub.ac.id Internet Source	<1%
12	arxiv.org Internet Source	<1%
13	Submitted to Universitas Islam Riau Student Paper	<1%
14	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1%
	Submitted to Universitas Sangga Buana YPKP	

Lampiran 3. Daftar Publikasi Yang Telah Dihasilkan

Jurnal Terakreditasi Sinta 4

1. "Real-Time Road Damage Detection on Mobile Devices using TensorF*Low* Lite and Teachable Machine",2025, MALCOM (Indonesian Journal of *Machine learning* and Computer Science), Juli 2025, Pekanbaru, DOI: https://doi.org/10.57152/malcom.v5i3.1933, ISSN 2775-8575 (Online), 2797-2313 (Print), Publisher Institut Riset dan Publikasi Indonesia (IRPI), url: https://journal.irpi.or.id/index.php/malcom

IEEE Xplore (ICITRI)

1. "Automatic CRUD Code Generation from UML Models for *Low*-Code Platforms: A Model-Driven Engineering Approach", 2025, ICITRI (International Conference on Information Technology Research and Innovation), September 2025, Jakarta, DOI: 10.1109/ICITRI67507.2025.11232843, Publisher IEEE, url: https://ieeexplore.ieee.org/document/11232843

Lampiran 4. Code Hasil Penelitian

A. Persiapan Library dan Upload Dataset

```
# Instalasi library utama
!pip install torch-geometric -q

import os
import zipfile
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
import torch.nn.functional as F

from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import (
    confusion_matrix, accuracy_score, precision_score,
    recall_score, f1_score, roc_auc_score, roc_curve, auc
)
from torch_geometric.data import Data
from torch_geometric.nn import GCNConv, GATConv
```

```
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(SEED)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Device:', device)

# Upload tiga file Elliptic Bitcoin Dataset:
# 1) elliptic_txs_features.zip
# 2) elliptic_txs_classes.csv
# 3) elliptic_txs_edgelist.csv
uploaded = files.upload()

# Ekstrak file features jika masih berbentuk ZIP
zip_files = [f for f in os.listdir('/content') if f.lower().endswith('.zip')]
for z in zip_files:
    with zipfile.ZipFile(os.path.join('/content', z), 'r') as zip_ref:
        zip_ref.extractall('/content')

print('File pada /content:')
print(os.listdir('/content'))
```

B. Membaca Dataset dan Memberi Nama Kolom

```
# Sesuaikan nama file dengan file yang terunggah di Google Colab
all_files = os.listdir('/content')
```

```
FEATURE_PATH = next(os.path.join('/content', f) for f in all_files
                    if 'features' in f.lower() and f.lower().endswith('.csv'))
CLASS_PATH = next(os.path.join('/content', f) for f in all_files
                  if 'classes' in f.lower() and f.lower().endswith('.csv'))
EDGE_PATH = next(os.path.join('/content', f) for f in all_files
                 if 'edgelist' in f.lower() and f.lower().endswith('.csv'))
features = pd.read_csv(FEATURE_PATH, header=None)
classes = pd.read_csv(CLASS_PATH)
edges_all = pd.read_csv(EDGE_PATH)

# 167 kolom = txId + Time_step + 165 fitur
# 165 fitur terdiri atas 93 local features dan 72 aggregate features
column_names = (
    ['txId', 'Time_step']
    + [f'Local_feature_{i}' for i in range(1, 94)]
    + [f'Aggregate_feature_{i}' for i in range(1, 73)]
)
features.columns = column_names

# Standarisasi nama kolom edge
edges_all.columns = ['txId1', 'txId2']

# Normalisasi label agar konsisten
classes['class'] = classes['class'].astype(str).str.strip().str.lower()
classes['class'] = classes['class'].replace({'unknown':'unknown', '1':'1', '2':'2'})

df_all = features.merge(classes[['txId','class']], on='txId', how='left')
df_all['label_name'] = df_all['class'].map({
```

```
   '1':'Illicit', '2':'Licit', 'unknown':'Unknown'
})

print('Features :', features.shape)
print('Classes  :', classes.shape)
print('Edges    :', edges_all.shape)
print('Merged   :', df_all.shape)
print(df_all['label_name'].value_counts())
```

Output utama dataset

Jumlah transaksi : 203.769

Licit : 42.019

Illicit : 4.545

Unknown : 157.205

Jumlah edge : 234.355

Jumlah fitur : 165

C. Preprocessing Data Berlabel (Licit dan Illicit)

```
feature_cols = (
   [f'Local_feature_{i}' for i in range(1, 94)]
   + [f'Aggregate_feature_{i}' for i in range(1, 73)]
)

# Unknown tidak digunakan sebagai target training/evaluasi
# karena tidak memiliki ground truth.
df_known = df_all[df_all['class'] != 'unknown'].copy()
df_known['target'] = df_known['class'].map({'2':0, '1':1})
df_known.reset_index(drop=True, inplace=True)

indices = np.arange(len(df_known))
```

```
train_idx, test_idx = train_test_split(
    indices,
    test_size=0.20,
    random_state=SEED,
    stratify=df_known['target']
)

X_known = df_known[feature_cols].to_numpy(dtype=np.float32)
y_known = df_known['target'].to_numpy(dtype=np.int64)

# Scaling hanya di-fit pada data training untuk mencegah data leakage
scaler = StandardScaler()
scaler.fit(X_known[train_idx])
X_known_scaled = scaler.transform(X_known)

print('Total labeled :', len(df_known))
print('Training      :', len(train_idx))
print('Testing       :', len(test_idx))
print('Shape fitur   :', X_known_scaled.shape)
print(df_known['target'].value_counts())
```

Output preprocessing

Total data berlabel : 46.564

Training : 37.251

Testing : 9.313

Licit (0) : 42.019

Illicit (1) : 4.545

D. Baseline Support Vector Machine (SVM)

```
# SVM menggunakan fitur transaksi secara individual (tanpa edge graph)
```

```
X_train = X_known_scaled[train_idx]
X_test  = X_known_scaled[test_idx]
y_train = y_known[train_idx]
y_test  = y_known[test_idx]

svm = SVC(kernel='rbf', random_state=SEED, probability=True)
svm.fit(X_train, y_train)

svm_pred = svm.predict(X_test)
svm_prob = svm.predict_proba(X_test)[:, 1]

svm_cm = confusion_matrix(y_test, svm_pred, labels=[0,1])
print('Confusion Matrix SVM:')
print(svm_cm)
print('Accuracy :', accuracy_score(y_test, svm_pred))
print('Precision:', precision_score(y_test, svm_pred))
print('Recall   :', recall_score(y_test, svm_pred))
print('F1-score :', f1_score(y_test, svm_pred))
print('ROC-AUC  :', roc_auc_score(y_test, svm_prob))
```

Output SVM yang digunakan pada Bab IV

Accuracy : 97,36%

Recall : 78,22%

FN : 198

TP : 711

E. Pembentukan Graph untuk Model GCN dan GAT

```
known_mapping = {txid: idx for idx, txid in enumerate(df_known['txId'])}

edges_known = edges_all[
```

```
    edges_all['txId1'].isin(known_mapping)
    & edges_all['txId2'].isin(known_mapping)
].copy()

edges_known['source'] = edges_known['txId1'].map(known_mapping)
edges_known['target'] = edges_known['txId2'].map(known_mapping)

edge_index_known = torch.tensor(
    edges_known[['source','target']].values.T,
    dtype=torch.long
)

x_known_tensor = torch.tensor(X_known_scaled, dtype=torch.float)
y_known_tensor = torch.tensor(y_known, dtype=torch.long)

train_mask = torch.zeros(len(df_known), dtype=torch.bool)
test_mask = torch.zeros(len(df_known), dtype=torch.bool)
train_mask[train_idx] = True
test_mask[test_idx] = True

data_known = Data(
    x=x_known_tensor,
    edge_index=edge_index_known,
    y=y_known_tensor
)
data_known.train_mask = train_mask
data_known.test_mask = test_mask
data_known = data_known.to(device)
```

```
print('Known nodes:', len(df_known))
print('Known edges:', len(edges_known))
print(data_known)
```

Output graph data berlabel

Known nodes : 46.564

Known edges : 36.624

Data(x=[46564, 165], edge_index=[2, 36624], y=[46564])

F. Definisi Model GCN dan GAT

```
class GCNModel(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim=64, output_dim=2):
        super().__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, output_dim)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return x
class GATModel(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim=64, output_dim=2,
                 heads=4, dropout=0.2):
        super().__init__()
        self.gat1 = GATConv(
            input_dim, hidden_dim, heads=heads, dropout=dropout
        )
        self.gat2 = GATConv(
```

```
            hidden_dim * heads, output_dim,
            heads=1, concat=False, dropout=dropout
        )
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.gat1(x, edge_index)
        x = F.elu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.gat2(x, edge_index)
        return x
```

G. Fungsi Training dan Evaluasi GCN/GAT

```
def train_graph_model(model, criterion, epochs=200, lr=0.01):
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)
    loss_history = []

    for epoch in range(1, epochs + 1):
        model.train()
        optimizer.zero_grad()
        output = model(data_known.x, data_known.edge_index)
        loss = criterion(
            output[data_known.train_mask],
            data_known.y[data_known.train_mask]
        )
        loss.backward()
        optimizer.step()
        loss_history.append(loss.item())
```

```
        if epoch == 1 or epoch % 20 == 0:
            print(f'Epoch {epoch:03d} | Loss: {loss.item():.6f}')
    model.eval()
    with torch.no_grad():
        logits = model(data_known.x, data_known.edge_index)
        prob = F.softmax(logits, dim=1)[:,1]
        pred = logits.argmax(dim=1)
    yt = data_known.y[data_known.test_mask].cpu().numpy()
    yp = pred[data_known.test_mask].cpu().numpy()
    ys = prob[data_known.test_mask].cpu().numpy()
    return model, loss_history, yt, yp, ys
def metric_row(name, yt, yp, ys):
    tn, fp, fn, tp = confusion_matrix(yt, yp, labels=[0,1]).ravel()
    return {
        'Model': name,
        'TN': tn, 'FP': fp, 'FN': fn, 'TP': tp,
        'Accuracy': accuracy_score(yt, yp),
        'Precision': precision_score(yt, yp),
        'Recall': recall_score(yt, yp),
        'F1-score': f1_score(yt, yp),
        'ROC-AUC': roc_auc_score(yt, ys)
    }
```

H. Training GCN dan GAT Tanpa Class Weight

```
criterion_plain = torch.nn.CrossEntropyLoss()
gcn_model = GCNModel(input_dim=len(feature_cols),
hidden_dim=64).to(device)
gcn_model, gcn_loss, y_gcn, p_gcn, s_gcn = train_graph_model(
    gcn_model, criterion_plain
```

```
)


gat_model = GATModel(
    input_dim=len(feature_cols), hidden_dim=64,
    output_dim=2, heads=4, dropout=0.2
).to(device)
gat_model, gat_loss, y_gat, p_gat, s_gat = train_graph_model(
    gat_model, criterion_plain
)
print(metric_row('GCN', y_gcn, p_gcn, s_gcn))
print(metric_row('GAT', y_gat, p_gat, s_gat))
```

Ringkasan hasil model graph pada Bab IV

GCN: kemampuan deteksi Illicit lebih baik daripada baseline SVM pada Recall.

GAT: mekanisme attention meningkatkan kemampuan model dalam mengenali kelas Illicit.

I. Perhitungan Class Weight

```
train_labels = y_known[train_idx]
weights = compute_class_weight(
    class_weight='balanced',
    classes=np.array([0,1]),
    y=train_labels
)
class_weights = torch.tensor(weights, dtype=torch.float).to(device)


print('Weight Licit   :', class_weights[0].item())
print('Weight Illicit :', class_weights[1].item())
```

Output Class Weight

Weight Licit   : 0,554083

Weight Illicit : 5,122525

J. Model Utama: GAT + Class Weight

```
gat_cw_model = GATModel(
    input_dim=len(feature_cols),
    hidden_dim=64,
    output_dim=2,
    heads=4,
    dropout=0.2
).to(device)
criterion_weighted = torch.nn.CrossEntropyLoss(weight=class_weights)


gat_cw_model, gat_cw_loss, y_gat_cw, p_gat_cw, s_gat_cw =
train_graph_model(
    gat_cw_model,
    criterion_weighted,
    epochs=200,
    lr=0.01
)


final_gat_cw = metric_row(
    'GAT + Class Weight',
    y_gat_cw,
    p_gat_cw,
    s_gat_cw
)
print(final_gat_cw)
```

Hasil final GAT + Class Weight yang digunakan pada Bab IV–V

Recall : 90,43%

ROC-AUC : 0,9809

False Negative (FN) : 87

True Positive (TP) : 822

K. Evaluasi Final dan Perbandingan Model

```
# Gabungkan seluruh hasil model ke satu tabel evaluasi
rows = []
rows.append(metric_row('SVM', y_test, svm_pred, svm_prob))
rows.append(metric_row('GCN', y_gcn, p_gcn, s_gcn))
rows.append(metric_row('GAT', y_gat, p_gat, s_gat))
rows.append(final_gat_cw)

result_final = pd.DataFrame(rows)
for c in ['Accuracy','Precision','Recall','F1-score']:
    result_final[c] = (result_final[c] * 100).round(2)
result_final['ROC-AUC'] = result_final['ROC-AUC'].round(4)

print(result_final)

# ROC Curve seluruh model
plt.figure(figsize=(8,6))
for name, yt, ys in [
    ('SVM', y_test, svm_prob),
    ('GCN', y_gcn, s_gcn),
    ('GAT', y_gat, s_gat),
    ('GAT + Class Weight', y_gat_cw, s_gat_cw)
]:
    fpr, tpr, _ = roc_curve(yt, ys)
    model_auc = auc(fpr, tpr)
```

```
    plt.plot(fpr, tpr, label=f'{name} (AUC={model_auc:.4f})')

plt.plot([0,1],[0,1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Perbandingan Kurva ROC')
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

L. Analisis Struktur Graph: Unknown yang Terhubung dengan Illicit

```
illicit_txids = set(
    df_all.loc[df_all['label_name'] == 'Illicit', 'txId']
)
unknown_txids = set(
    df_all.loc[df_all['label_name'] == 'Unknown', 'txId']
)

# Arah Illicit -> Unknown
illicit_to_unknown = edges_all[
    edges_all['txId1'].isin(illicit_txids)
    & edges_all['txId2'].isin(unknown_txids)
].copy()

# Arah Unknown -> Illicit
unknown_to_illicit = edges_all[
    edges_all['txId1'].isin(unknown_txids)
    & edges_all['txId2'].isin(illicit_txids)
```

```
].copy()

unknown_neighbor_ids = set(illicit_to_unknown['txId2']).union(
    set(unknown_to_illicit['txId1'])
)

unknown_neighbor_df = df_all[
    df_all['txId'].isin(unknown_neighbor_ids)
][['txId','Time_step','label_name']].copy()

outgoing_count = unknown_to_illicit.groupby('txId').size()
incoming_count = illicit_to_unknown.groupby('txId2').size()

unknown_neighbor_df['Unknown_to_Illicit'] = (
    unknown_neighbor_df['txId'].map(outgoing_count).fillna(0).astype(int)
)
unknown_neighbor_df['Illicit_to_Unknown'] = (
    unknown_neighbor_df['txId'].map(incoming_count).fillna(0).astype(int)
)
unknown_neighbor_df['Total_Illicit_Connections'] = (
    unknown_neighbor_df['Unknown_to_Illicit']
    + unknown_neighbor_df['Illicit_to_Unknown']
)

print('Unknown yang terhubung langsung dengan Illicit:',
len(unknown_neighbor_df))
```

Output Neighbor Risk Analysis

Unknown terhubung langsung dengan Illicit : 4.027 transaksi

M. Inferensi GAT + Class Weight terhadap Seluruh Transaksi Unknown

```
# Bentuk graph penuh: 203.769 node dan 234.355 edge
all_mapping = {txid: idx for idx, txid in enumerate(df_all['txId'])}
edges_full = edges_all.copy()
edges_full['source'] = edges_full['txId1'].map(all_mapping)
edges_full['target'] = edges_full['txId2'].map(all_mapping)

edge_index_full = torch.tensor(
    edges_full[['source','target']].values.T,
    dtype=torch.long
).to(device)

X_all = df_all[feature_cols].to_numpy(dtype=np.float32)
X_all_scaled = scaler.transform(X_all)
x_all = torch.tensor(X_all_scaled, dtype=torch.float).to(device)

# Inferensi menggunakan model utama GAT + Class Weight
gat_cw_model.eval()
with torch.no_grad():
    logits_all = gat_cw_model(x_all, edge_index_full)
    probabilities_all = F.softmax(logits_all, dim=1)
    predictions_all = probabilities_all.argmax(dim=1)

unknown_mask = (df_all['label_name'] == 'Unknown').to_numpy()
unknown_indices = np.where(unknown_mask)[0]

unknown_probabilities = probabilities_all[unknown_indices].cpu().numpy()
unknown_predictions = predictions_all[unknown_indices].cpu().numpy()
```

```
unknown_result = pd.DataFrame({
   'txId': df_all.loc[unknown_mask, 'txId'].values,
   'Time_step': df_all.loc[unknown_mask, 'Time_step'].values,
   'Prob_Licit': unknown_probabilities[:,0],
   'Prob_Illicit': unknown_probabilities[:,1],
   'Prediction_Code': unknown_predictions
})
unknown_result['Prediction'] = unknown_result['Prediction_Code'].map({
   0:'Licit', 1:'Illicit'
})

unknown_summary = (
   unknown_result.groupby('Prediction').size().reset_index(name='Jumlah')
)
unknown_summary['Persentase (%)'] = (
   unknown_summary['Jumlah'] / len(unknown_result) * 100
).round(2)

print('Total Unknown:', len(unknown_result))
print(unknown_summary)
```

Output prediksi Unknown

Total Unknown : 157.205

Diprediksi Licit : 119.100 (75,76%)

Diprediksi Illicit : 38.105 (24,24%)

N. Kombinasi Prediksi Model dan Hubungan Graph

```
unknown_forensic = unknown_result.merge(
   unknown_neighbor_df[[
```

```
        'txId',
        'Unknown_to_Illicit',
        'Illicit_to_Unknown',
        'Total_Illicit_Connections'
    ]],
    on='txId',
    how='left'
)

cols_conn = [
    'Unknown_to_Illicit',
    'Illicit_to_Unknown',
    'Total_Illicit_Connections'
]
unknown_forensic[cols_conn] = (
    unknown_forensic[cols_conn].fillna(0).astype(int)
)

# Kandidat prioritas: diprediksi Illicit DAN memiliki hubungan langsung
# dengan transaksi yang telah berlabel Illicit.
forensic_candidates = unknown_forensic[
    (unknown_forensic['Prediction'] == 'Illicit')
    & (unknown_forensic['Total_Illicit_Connections'] > 0)
].sort_values(
    ['Prob_Illicit','Total_Illicit_Connections'],
    ascending=[False,False]
)
```

```
print('=' * 70)
print('RINGKASAN ANALISIS UNKNOWN')
print('=' * 70)
print('Total Unknown:', len(unknown_result))
print('Diprediksi Illicit:', (unknown_result['Prediction']=='Illicit').sum())
print('Terhubung langsung dengan Illicit:', len(unknown_neighbor_df))
print('Diprediksi Illicit DAN terhubung dengan Illicit:', len(forensic_candidates))
```

Output final analisis Unknown untuk digital forensik

Total Unknown : 157.205

Diprediksi Illicit : 38.105

Terhubung langsung dengan Illicit : 4.027

Diprediksi Illicit + terhubung Illicit : 1.942

O. Menampilkan Kandidat Prioritas dan Menyimpan Hasil

```
# Tampilkan 20 kandidat prioritas tertinggi
print(forensic_candidates[[
    'txId','Time_step','Prob_Licit','Prob_Illicit','Prediction',
    'Unknown_to_Illicit','Illicit_to_Unknown',
    'Total_Illicit_Connections'
]].head(20))

# Simpan seluruh hasil analisis
unknown_result.to_csv(
    '/content/Hasil_Prediksi_Unknown.csv', index=False
)
unknown_neighbor_df.to_csv(
    '/content/Unknown_Neighbor_Illicit.csv', index=False
)
forensic_candidates.to_csv(
```

```
    '/content/Kandidat_Unknown_Digital_Forensik.csv', index=False
)
result_final.to_csv(
    '/content/Hasil_Evaluasi_Final_Model.csv', index=False
)

print('Semua file hasil penelitian berhasil disimpan.')

# Download bila diperlukan
files.download('/content/Hasil_Evaluasi_Final_Model.csv')
files.download('/content/Hasil_Prediksi_Unknown.csv')
files.download('/content/Unknown_Neighbor_Illicit.csv')
files.download('/content/Kandidat_Unknown_Digital_Forensik.csv')
```