ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

каф. ЕОМ

Система виявлення аномалій у трафіку

Ст. групи КІУКІ-21-6
Слабунов К.А.

Керівник
ас.Крюкова І.В.

2025

Мета та задачі кваліфікаційної роботи

Мета кваліфікаційної роботи розробка системи виявлення аномалій у трафіку

Задачі:

- Провести аналіз теоретичних досліджень та виявити актуальний метод
- Запропонувати модель побудовану на використанні методів ШІ
- Обрати набори даних та показники оцінювання
- Провести тестування та виявити аномалії в трафіку

Актуальність роботи

- Сучасні методи виявлення аномалій не враховують різну важливість ознак на різних каналах, а безперервні операції згортки та об'єднання можуть призвести до втрати інформації. Для вирішення цих проблем в роботі пропонуємо, метод виявлення аномалій мережевого трафіку, заснований на розширеній згортці та увазі каналу.
- Модель розроблена для ефективної та точної класифікації інформації про трафік.
- Запропонована архітектура системи та метод розширюють можливості виявлення аномалій . Основні нововведення полягають у наступному.
- Для запобігання втраті інформації, спричиненій зниженням частоти дискретизації під час процесу згортки, було побудовано модель вилучення ознак на основі розширеної згортки, зокрема розширену згортку-1D (DC-1D). Крім того, було розроблено структуру, що містить розширені згортки зі швидкостями розширення 1, 2 та 3, що забезпечує безперервність ознак під час процесу вилучення інформації.
- Для моделі було розроблено метод оцінки важливості ознакових каналів, використовуючи увагу до каналу. Це пояснюється тим, що використання лише згортки не може виділити різні проблеми на основі важливості ознак, що впливає на ефективність виявлення.

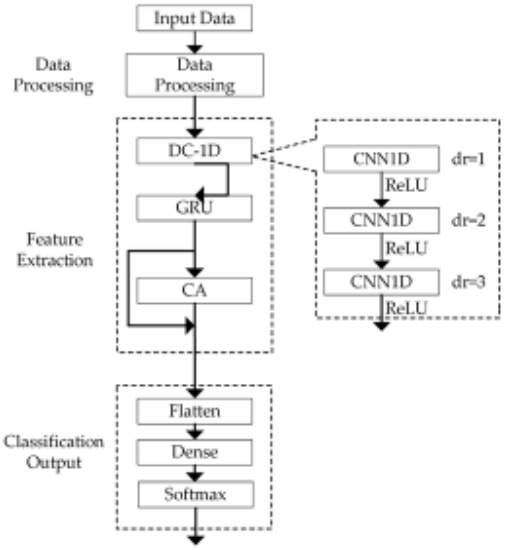
3

Загальна структура моделі

- Модель складається з трьох частин: обробка даних, вилучення ознак та класифікація результатів, як показано на рисунку
- Попередня обробка даних проводиться, оскільки різні інтервали ознак у векторі ознак трафіку мають різні розмірності. Щоб пом'якшити вплив цих відмінностей у розмірності на результати класифікації, ознаки спочатку слід нормалізувати. Нормалізація середнього значення та дисперсії використовується для перетворення даних у нормальний розподіл із середнім значенням 0 та дисперсією 1. Формула для нормалізації така:

$$x' = \frac{x - x_m}{s}$$

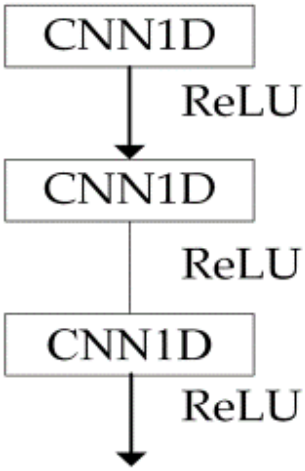
де x' – оброблені дані, x – характеристичне значення кожної групи вхідних ознак, x_m– середнє значення вхідних ознак, а s – стандартне відхилення кожної групи вхідних ознак.



4

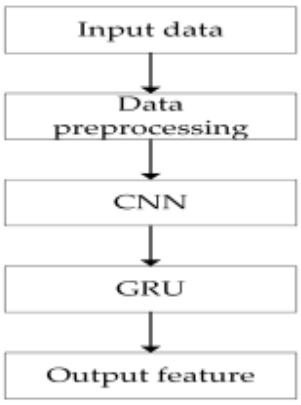
Структура DC-1D

- Щоб уникнути втрати інформації, спричиненої операціями об'єднання в традиційних згортках, у цій роботі замість операцій об'єднання використовуються розширені згортки. Розширені згортки можуть розширювати рецептивне поле без зміни роздільної здатності ознак, тим самим покращуючи швидкість обчислень.
- Оскільки розширені згортки можуть призвести до втрати безперервності між ознаками, щоб уникнути цього та досягти багатомасштабного збору інформації, розширені згортки зі швидкостями розширення 1, 2 та 3 об'єднуються для формування структури DC-1D, як показано на рисунку.

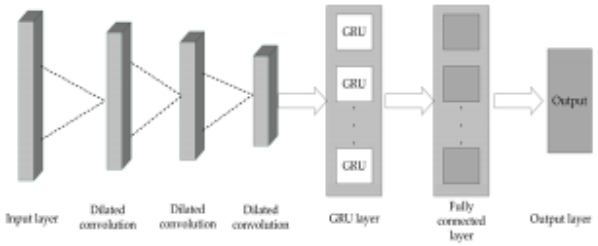


5

DC-GRU



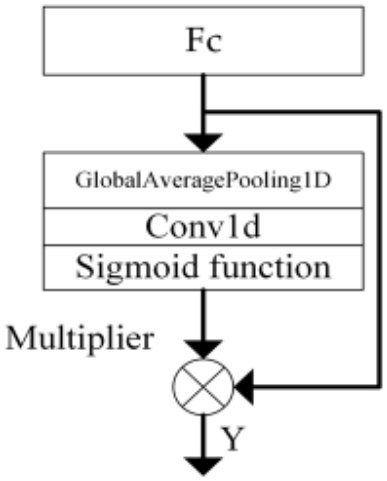
- Робочий процес моделі DC-GRU починається з попередньої обробки набору даних. Попередньо оброблені дані потім вводяться в модель CNN для вилучення просторових ознак.
- Згодом ці просторові ознаки подаються в GRU для вилучення часових ознак. Детальний процес реалізації проілюстровано на на рисунку 1, а архітектура моделі CNN-GRU зображена на рисунку 2



6

Модуль CA

- Через те, що різні дані трафіку можуть демонструвати однакові ознаки, важливість цих ознак для класифікації трафіку різна. Існуючі алгоритми виявлення аномалій трафіку не враховують ступінь, до якої ознаки трафіку в різних каналах впливають на результати класифікації. Для вирішення цієї проблеми використовується модуль CA (channel attention - увага до каналу), як показано на рисунку.



Модуль CA може адаптивно визначати ядро згортки k за допомогою одновимірної згортки, тим самим уникаючи необхідності ручного налаштування ядра згортки. Адаптивний розрахунок ядра згортки k виглядає наступним чином:

$$k = \left| \frac{\log_2 C}{\gamma} + \frac{b}{\gamma} \right|_{odd}$$

де C – розмірність каналу, γ = 2, b = 1, де непарне число позначає непарне число.

7

Набори даних та показники оцінювання

- Для навчання та валідації було використано набір даних CIC-IDS-2017. Цей набір даних містить дані про трафік з 3 липня 2017 року по 7 липня 2017 року, включаючи звичайний трафік та 14 типів атак. В експерименті враховувався звичайний трафік (BENIGN) та дев'ять типів атак. Співвідношення навчального набору до тестового набору становило 7:3.
- Набір даних CIC-IDS-2017 містить загалом 79 ідентифікаторів із 78 ознаками, а останній ідентифікатор використовувався для позначення того, чи відображають дані нормальну чи аномальну поведінку.

Матриця похибки		Прогнозоване значення	
		Позитивний клас	Негативний клас
Справжнє значення	Позитивний клас	TP	FN
	Негативний клас	FP	TN

Для оцінки ефективності моделі використовувалися такі показники, як точність, прецизійність, повнота та F1-оцінка. Формули мають наступний вигляд:

$$\text{Accuracy} = \frac{TP}{TP + TN + FP + FN} \qquad \text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN} \qquad F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

8

Значення точності, повноти та F1 для різних підмножин набору даних CIC-IDS-2017

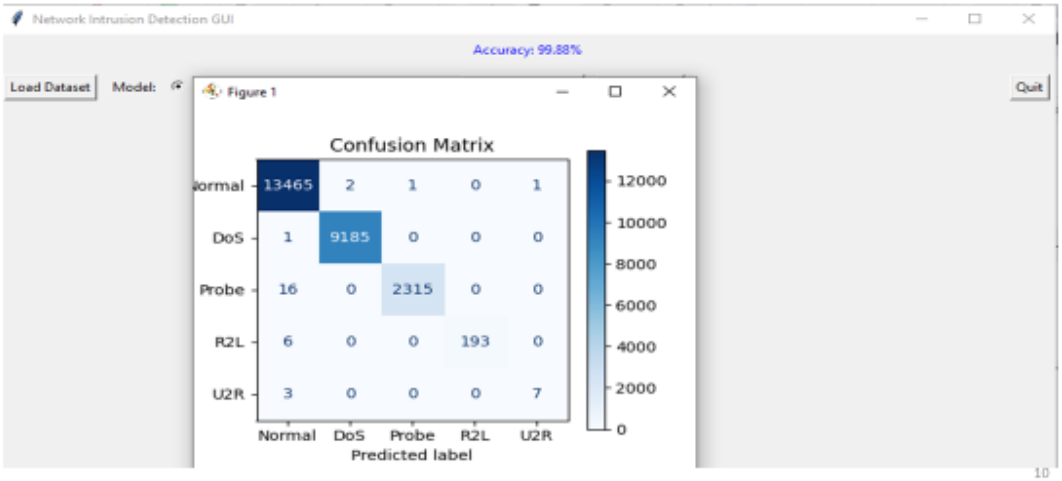
Щоб перевірити здатність DCGCANet до узагальнення, ми використали чотири підмножини P1-P4 CIC-IDS2017 як тестовий набір, і DCGCANet порівняли з моделлю LSTM на цих наборах даних. Результати представлені в таблиці.

DataSets	Pre		Recall		F1	
	LSTM	DCGCANet	LSTM	DCGCANet	LSTM	DCGCANet
P1	73.6%	99.3%	75.5%	99.4%	76.6%	99.4%
P2	73.1%	99.4%	73.2%	99.5%	73.1%	99.3%
P3	76.1%	99.6%	75.7%	99.7%	76.1%	99.7%
P4	68.3%	99.6%	70.3%	99.6%	68.3%	99.6%

З результатів порівняння, представлених у таблиці, можна побачити, що значення точності, повноти та F1 запропонованої DCGCANet на чотирьох підмножинах CIC-IDS2017 перевищували 99%, і ці експериментальні результати значно кращі, ніж результати моделі LSTM. Отже, можна зробити висновок, що DCGCANet має чудову здатність до узагальнення. Це пояснюється акцентом моделі DCGCANet на важливих характеристиках каналу шляхом зважування каналу, що покращує репрезентативні можливості моделі та призводить до покращеної здатності до узагальнення.

9

Реалізація системи



10

Результати запропонованого методу на виявлення даних NSL-KDD

Для подальшої перевірки здатності моделі до узагальнення, для валідації було використано набір даних KDD Cup99, результати яких наведено в таблиці.

Набір даних KDD Cup99 містить нормальну поведінку та чотири типи аномальної поведінки: звичайний трафік (Normal), атака відмови в обслуговуванні (DOS), атака зондування (Probe), атака віддаленого входу (R2L) та атака користувача до root (U2R).

Як показано в таблиці, точність та коефіцієнти виявлення для всіх показників п'яти типів атак у наборі даних KDD Cup99 перевищували 90% за допомогою запропонованої моделі.

Модель продемонструвала відмінну ефективність виявлення як на наборах даних CIC-IDS-2017, так і на KDD Cup99, що свідчить про високу здатність до узагальнення.

Attack Categories	Acc	Pre	Recall	F1
DoS	99.1%	99.2%	98.3%	97.9%
Normal	95.3%	95.6%	92.7%	93.3%
R2L	97.3%	97.3%	98.4%	98.0%
Probe	93.1%	93.2%	96.6%	95.2%
U2R	97.6%	97.6%	97.5%	97.6%

11

ВИСНОВКИ

В ході кваліфікаційної роботи було побудована система виявлення аномалій в трафіку.

Також, було вирішено наступні задачі:

- Проведено аналіз теоретичних досліджень та представлено актуальний метод виявлення аномалій в трафіку
- Запропоновано модель побудовану на використанні методів ШІ
- Обрано набори даних та показники оцінювання
- Проведено тестування та виявлені аномалії в трафіку

12

ДОДАТОК Б
ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

```
import pandas as pd
import numpy as np
import tkinter as tk
from tkinter import filedialog, messagebox
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay, accuracy_score

# Мапінг конкретних назв атак до загальних категорій
attack_category = {
    'normal': 'Normal',
    # DoS
    'back': 'DoS', 'land': 'DoS', 'neptune': 'DoS', 'pod':
'DoS', 'smurf': 'DoS', 'teardrop': 'DoS',
    # Probe
    'ipsweep': 'Probe', 'nmap': 'Probe', 'portsweep': 'Probe',
'satan': 'Probe',
    # R2L
    'ftp_write': 'R2L', 'guess_passwd': 'R2L', 'imap': 'R2L',
'multihop': 'R2L', 'phf': 'R2L',
    # U2R
    'buffer_overflow': 'U2R', 'loadmodule': 'U2R', 'perl':
'U2R', 'rootkit': 'U2R'
}

def preprocess_data(csv_path):
    """Завантаження та передобробка даних"""
    df = pd.read_csv(csv_path, header=None)
    # Імена 41 ознаки
    feature_names = [

"duration","protocol_type","service","flag","src_bytes","dst_byt
es","land",

"wrong_fragment","urgent","hot","num_failed_logins","logged_in",
"num_compromised",

"root_shell","su_attempted","num_root","num_file_creations","num
_shells",
```

```
"num_access_files","num_outbound_cmds","is_host_login","is_guest
_login",

"count","srv_count","serror_rate","srv_serror_rate","rerror_rate
","srv_rerror_rate",

"same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_c
ount",

"dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv
_rate",

"dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst
_host_serror_rate",

"dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_
rerror_rate"
    ]
    n_cols = df.shape[1]
    # Обробка додаткової колонки difficulty
    if n_cols == len(feature_names) + 2:
        df.columns = feature_names + ["label","difficulty"]
        df = df.drop(columns=["difficulty"])
    else:
        df.columns = feature_names + ["label"]
    # Мапінг міток (видалити '.' та перекласти)
    df['label'] =
df['label'].str.rstrip('.').map(attack_category)
    # Розділ X та y
    X = df.drop(columns=['label'])
    y = df['label']
    # one-hot кодування категоріальних ознак
    X = pd.get_dummies(X,
columns=['protocol_type','service','flag'], drop_first=True)
    # Масштабування числових ознак
    scaler = StandardScaler()
    num_cols = [c for c in X.columns if X[c].dtype in [np.int64,
np.float64]]
    X[num_cols] = scaler.fit_transform(X[num_cols])
    # Розбивка на тренувальний та тестовий набори (80/20),
стратифіковано за y
    X_train,X_test,y_train,y_test = train_test_split(
        X,y,test_size=0.2,random_state=42,stratify=y
    )
    return X_train,X_test,y_train,y_test

# --- GUI ---
results_df = None
csv_path = None

root = tk.Tk()
root.title("Детекція аномалій у трафіку")
```

```
root.geometry("920x650")

status_label = tk.Label(root, text="Датасет не завантажено.",
fg="blue")
status_label.pack(pady=5)

btn_frame = tk.Frame(root)
btn_frame.pack(fill=tk.X, pady=10)

# Кнопка завантаження датасету
def load_dataset():
    global csv_path
    path = filedialog.askopenfilename(
        title="Виберіть CSV-датасет", filetypes=[("CSV
файли","*.csv"),("Всі файли","*")]
    )
    if path:
        csv_path = path
        status_label.config(text=f"Завантажено:
{path.split('/')[-1]}")
    else:
        status_label.config(text="Завантаження скасовано.")

load_btn = tk.Button(btn_frame, text="Завантажити датасет",
command=load_dataset)
load_btn.pack(side=tk.LEFT, padx=5)

# Вибір моделі
model_var = tk.StringVar(value="RF")
tk.Label(btn_frame, text="Модель:").pack(side=tk.LEFT, padx=5)
rf_radio = tk.Radiobutton(btn_frame, text="Випадковий ліс",
variable=model_var, value="RF")
lr_radio = tk.Radiobutton(btn_frame, text="Логістична регресія",
variable=model_var, value="LR")
rf_radio.pack(side=tk.LEFT)
lr_radio.pack(side=tk.LEFT, padx=5)

# Кнопка запуску класифікації
def run_classification():
    global results_df
    if not csv_path:
        messagebox.showwarning("Попередження","Завантажте
датасет перед роботою.")
        return
    try:
        X_train,X_test,y_train,y_test =
preprocess_data(csv_path)
    except Exception as e:
        messagebox.showerror("Помилка","Не вдалося обробити
дані:\n" + str(e))
        return
    # Вибір класифікатора
    if model_var.get()=="RF":
```

```
        clf = RandomForestClassifier(n_estimators=100,
random_state=42)
    else:
        clf = LogisticRegression(max_iter=500, random_state=42)
    # Навчання та передбачення
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    y_proba = clf.predict_proba(X_test)
    # Точність
    acc = accuracy_score(y_test,y_pred)
    status_label.config(text=f"Точність: {acc:.2%}")
    # Графіки: матриця похибки + гістограми
    labels = ['Normal','DoS','Probe','R2L','U2R']
    cm = confusion_matrix(y_test,y_pred,labels=labels)
    fig, axes = plt.subplots(1,3,figsize=(18,5))
    # Матриця похибки
    disp = ConfusionMatrixDisplay(cm, display_labels=labels)
    disp.plot(ax=axes[0], cmap=plt.cm.Blues)
    axes[0].set_title("Матриця похибки")
    # Гістограма справжніх міток
    uniq, cnt = np.unique(y_test, return_counts=True)
    axes[1].bar(uniq, cnt, color='skyblue')
    axes[1].set_title("Розподіл справжніх міток")
    axes[1].set_xlabel("Клас")
    axes[1].set_ylabel("Кількість")
    axes[1].tick_params(axis='x', rotation=45)
    # Гістограма передбачених
    uniq_p, cnt_p = np.unique(y_pred, return_counts=True)
    axes[2].bar(uniq_p, cnt_p, color='salmon')
    axes[2].set_title("Розподіл передбачених міток")
    axes[2].set_xlabel("Клас")
    axes[2].set_ylabel("Кількість")
    axes[2].tick_params(axis='x', rotation=45)
    plt.tight_layout()
    plt.show()
    # Підготовка результатів для збереження
    data = {'Справажні':y_test.to_numpy(), 'Передбачені':y_pred}
    for idx,cls in enumerate(clf.classes_):
        data[f"Ймовірність_{cls}"] = y_proba[:,idx]
    results_df = pd.DataFrame(data)
    save_btn.config(state=tk.NORMAL)

run_btn = tk.Button(btn_frame, text="Запустити класифікацію",
command=run_classification)
run_btn.pack(side=tk.LEFT, padx=5)

# Кнопка збереження результатів
def save_results():
    global results_df
    if results_df is None:
        return
    path = filedialog.asksaveasfilename(defaultextension=".csv",
                                        filetypes=[("CSV
```

```
файли","*.csv")],
                                                title="Зберегти
результати")
    if path:
        try:
            results_df.to_csv(path, index=False)
            messagebox.showinfo("Збережено", f"Результати
збережено в:\n{path}")
        except Exception as e:
            messagebox.showerror("Помилка","Не вдалося
зберегти:\n" + str(e))

save_btn = tk.Button(btn_frame, text="Зберегти результати",
command=save_results, state=tk.DISABLED)
save_btn.pack(side=tk.LEFT, padx=5)

# Кнопка виходу
quit_btn = tk.Button(btn_frame, text="Вихід", command=root.quit)
quit_btn.pack(side=tk.RIGHT, padx=5)

root.mainloop()
```