VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ NEURONOVÝCH SÍTÍ PRO ANALÝZU SÍTĚ BITCOIN

USAGE OF NEURAL NETWORKS IN BITCOIN NETWORK ANALYSIS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MIROSLAV ŠAFÁŘ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN PLUSKAL, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

164715

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Šafář Miroslav, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Kybernetická bezpečnost
Název:	**Využití neuronových sítí pro analýzu sítě Bitcoin**
Kategorie:	Umělá inteligence
Akademický rok:	2024/25

Zadání:

1. Seznamte se s principy a základními mechanismy sítě Bitcoin a jejího blockchainu.
2. Prozkoumejte možnosti analýzy blockchainu sítě Bitcoin využívající neuronových sítí. Po konzultaci s vedoucím zvolte vhodný typ neuronové sítě pro použití v bodu 3.
3. Identifikujte vhodné případy užití analýzy blockchainu pomocí neuronových sítí.
4. Zvolte vhodné případy užití, navrhněte experimenty pro jejich realizaci a implementujte je s využitím datové sady z volně dostupných zdrojů nebo Vámi vytvořené na základě parametrů experimentů a konzultace s vedoucím.
5. Vyhodnoťte realizované experimenty a diskutujte dosažené výsledky.

Literatura:

- LI, Yang, Yue CAI, Hao TIAN, Gengsheng XUE a Zibin ZHENG, 2020. Identifying Illicit Addresses in Bitcoin Network. In: Zibin ZHENG, Hong-Ning DAI, Xiaodong FU a Benhui CHEN, ed. *Blockchain and Trustworthy Systems* [online]. Singapore: Springer, s. 99–111. ISBN 9789811592133. Dostupné z: doi:10.1007/978-981-15-9213-3_8
- LI, Zhiyuan a Enhan HE, 2023. Graph Neural Network-Based Bitcoin Transaction Tracking Model. *IEEE Access* [online]. **11**, 62109–62120. ISSN 2169-3536. Dostupné z: doi: 10.1109/ACCESS.2023.3288026
- SHEN, Jie, Jiajun ZHOU, Yunyi XIE, Shanqing YU a Qi XUAN, 2021. Identity Inference on Blockchain Using Graph Neural Network. In: Hong-Ning DAI, Xuanzhe LIU, Daniel Xiapu LUO, Jiang XIAO a Xiangping CHEN, ed. *Blockchain and Trustworthy Systems* [online]. Singapore: Springer, s. 3–17. ISBN 9789811679933. Dostupné z: doi:10.1007/978-981-16-7993-3_1

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Pluskal Jan, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	21.5.2025
Datum schválení:	14.10.2024

Abstrakt

Pseudonymita a absence centrální kontrolní entity u kryptoměny Bitcoin přitahuje pozornost zlomyslných aktérů, kteří chtějí zneužít blockchainové technologie k praní špinavých peněz, financování terorismu, k podvodům a nebo k nákupům nelegálního zboží na darknet tržištích. Identifikace podvodného jednání a nákupů nelegálního zboží představuje klíčový úkol bezpečnostních složek, přičemž detekce praní špinavých peněz a financování terorismu pak patří mezi hlavní priority institucí zaměřených na boj s finanční kriminalitou. Tato práce se proto věnuje využití hlubokého učení pro forenzní analýzu blockchainu sítě Bitcoin. Práce stanovuje hlavní výzvy, které je zapotřebí adresovat, aby mohly být tyto metody strojového učení využity ve forenzní praxi, a analyzuje obsah nejpoužívanější anonymizované datové sady využívané k trénování modelů na identifikaci transakcí spojených s ilegální aktivitou. Za pomoci reverzního inženýrství se mi podařilo deanonymizovat 47 % původních příznaků, a díky tomu identifikovat 100 % transakcí obsažených v této datové sadě. Tato diplomová práce přináší otevřenou datovou sadu, která opravuje zásadní nedostatky původní anonymizované datové sady, a poskytuje tak potřebný základ pro nasazení metod hlubokého učení v praktických forenzních aplikacích. Nakonec tato práce experimentálně vyhodnocuje a porovnává tradiční metody strojového učení s nejmodernějšími metodami hlubokého učení založenými na grafových neuronových sítí, a to jak na původní anonymizované datové sadě, tak na nově představené otevřené datové sadě.

Abstract

Pseudonymity and the absence of a central control entity in the Bitcoin cryptocurrency attract the attention of malicious actors seeking to exploit blockchain technology for money laundering, terrorism financing, fraud, and the purchase of illegal goods on darknet marketplaces. The identification of fraudulent activities and illegal purchases is a key task for security forces, while the detection of money laundering and terrorism financing is a primary focus for institutions combating financial crime. This thesis focuses on the use of deep learning for forensic analysis of the Bitcoin network blockchain. It outlines the main challenges that need to be addressed for machine learning methods to be effectively applied in forensic practice, and analyzes the content of the most commonly used anonymized dataset employed to train models for identifying transactions associated with illegal activity. Through reverse engineering, I was able to deanonymize 47 % of the original labels, which enabled the identification of 100 % of the transactions contained in this dataset. This thesis presents an open dataset that corrects critical flaws in the original anonymized dataset, thereby providing a necessary foundation for the deployment of deep learning methods in practical forensic applications. Finally, the thesis experimentally evaluates and compares traditional machine learning techniques with state-of-the-art deep learning methods based on graph neural networks, using both the original anonymized dataset and the newly introduced open dataset.

Klíčová slova

Bitcoin, blockchain, neuronové sítě, grafové neuronové sítě, analýza blockchainu, datová sada, reverzní inženýrství

Keywords

Bitcoin, blockchain, neural networks, graph neural networks, blockchain analysis, dataset, reverse engineering

Citace

ŠAFÁŘ, Miroslav. *Využití neuronových sítí pro analýzu sítě Bitcoin*. Brno, 2025. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal, Ph.D.

Využití neuronových sítí pro analýzu sítě Bitcoin

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci na téma Využití neuronových sítí pro analýzu sítě Bitcoin vypracoval samostatně pod vedením pana Ing. Jana Pluskala Ph.D. Při práci jsem využil generativní umělé inteligence ve formě nástroje ChatGPT a automatické korektury nástroje Overleaf, přičemž tyto nástroje byly využity ke kontrole pravopisu a ke stylistickým úpravám mnou samostatně napsaného textu. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Miroslav Šafář
18. května 2025

Poděkování

Chtěl bych poděkovat svému vedoucímu doktoru Janu Pluskalovi za jeho čas, cenné rady, informace a nápady, které mi pomohly zpracovat tuto práci, a za jeho ochotu se mnou konzultovat moji práci i o vánočních svátcích. Dále bych chtěl poděkovat své přítelkyni za to, že tady pro mě byla po celou dobu mého studia, respektovala můj perfekcionismus a zvládla se mnou přežít noci plné studia a práce. Chtěl bych také poděkovat svým rodičům, kteří mě celých pět let podporovali a kteří mi jsou vzorem po celý můj život. Děkuji vám všem, bez vás by tato práce nemohla vzniknout.

Obsah

Kapitola 1

Úvod

Příchod kryptoměn založených na technologii blockchainu způsobil v tradičně konzervativním finančním světě revoluci. Jejich vlastnosti, jako jsou pseudonymita, decentralizovanost, a tedy i absence jakékoliv centrální kontrolní entity, přitahují pozornost jak běžných uživatelů, tak i zlomyslných aktérů, kteří chtějí zneužít blockchainové technologie k ilegálním aktivitám, jako například praní špinavých peněz, financování terorismu, vydírání, podvody a nakupování ilegálního zboží na darknet tržištích nebo k obcházení ekonomických sankcí, jak také můžeme vidět v případě rusko-ukrajinské války. Zajištění integrity a bezpečnosti finančního systému proti takovýmto hrozbám je dnes jedním z kritických úkolů bezpečnostních složek a finančních institucí. Z tohoto důvodu se forenzní analýza blockchainu, tedy proces analýzy dat za účelem například *trasování prostředků*, *odhalování identit uživatelů* nebo *detekce různých vzorů*, stala zásadní oblastí forenzního výzkumu a aplikační praxe.

Hluboké učení (anglicky Deep Learning) se stalo dominantním směrem v oblasti výzkumu umělé inteligence a nabízí velmi slibné výsledky v mnoha aplikačních oblastech. Z tohoto důvodu se v rámci této práce zaměřuji na využití metod hlubokého učení pro forenzní analýzu blockchainu sítě Bitcoin. Práce *identifikuje případy užití* grafových neuronových sítí ve forenzní praxi a *analyzuje* dostupné datové sady, které mohou sloužit pro trénování nových modelů pro identifikované případy užití, přičemž se dále zaměřuje primárně na případ užití identifikace transakcí spojených s ilegální aktivitou.

Blockchain sítě Bitcoin lze přirozeně reprezentovat jako graf transakcí. Při klasifikaci transakcí na ty legální a na ty spojené s ilegální aktivitou pak lze pracovat se dvěma typy informací — s informacemi obsaženými v samotné klasifikované transakci a s informacemi o kontextu této transakce, tedy informacemi z okolních uzlů v grafu. Tato diplomová práce si klade za cíl ověřit dvě prvotní hypotézy související s klasifikací transakcí kryptoměny Bitcoin:

1. Využití informací z okolí transakce by mělo vést k lepšímu výkonu klasifikační metody.
2. Grafové neuronové sítě by měly dokázat efektivně agregovat informace z okolních uzlů transakce, a tak překonat základní agregační funkce, jako je například průměr.

Práce analyzuje, vyhodnocuje a porovnává tradiční metody strojového učení, konkrétně algoritmy Random Forest a XGBoost, s přístupy hlubokého učení využívajícími grafové neuronové sítě. Dnes nejpoužívanější datovou sadou pro trénování modelů k identifikaci transakcí spojených s ilegální aktivitou je anonymizovaná sada `Elliptic` [28], jak dokládá odborná literatura [1, 4, 5, 21, 26]. Tuto datovou sadu se mi podařilo za pomoci reverzního inženýrství deanonymizovat, čímž se podařilo odhalit její zásadní nedostatky, které brání

využití na této sadě natrénovaných modelů ve forenzní praxi. Tato práce proto představuje novou otevřenou datovou sadu `OpenElliptic`, která vychází z původní datové sady `Elliptic`, avšak opravuje identifikované nedostatky a může se tak stát rozšiřitelným základem pro vývoj forenzních aplikací.

Práce je členěna následovně. Popisem kryptoměny Bitcoin [19], základních použitých kryptografických primitiv a popisem jejího blockchainu se zabývá kapitola 2. Kapitola 2.3 pojednává o případech užití neuronových sítí pro analýzu blockchainu sítě Bitcoin a popisuje dostupné datové sady, nejvyužívanější datovou sadu `Elliptic` [28], z ní vytvořenou datovou sadu `Elliptic++` [6], datovou sadu `Elliptic2` [3] a sadu `BitcoinTemporalGraph` [22], které lze využít k realizaci identifikovaných případů užití. Jakožto nejslibnější metodě hlubokého učení pro analýzu blockchainu sítě Bitcoin — grafovým neuronovým sítím a jejich nejvýznamnějším architekturám se věnuje kapitola 3. Následující kapitola 4 se zabývá reverzním inženýrstvím anonymizované datové sady `Elliptic`, identifikací jejích nedostatků a tvorbou nové otevřené datové sady `OpenElliptic`. Kapitola 5 se věnuje porovnání nejmodernějších metod detekce transakcí spojených s nelegální činností, a to jak na dostupných datových sadách, tak i na nově vytvořené datové sadě `OpenElliptic`.

Kapitola 2

Kryptoměna Bitcoin

Kryptoměnu Bitcoin představil Satoshi Nakamoto v roce 2008 ve své práci Bitcoin: A Peer-to-Peer Electronic Cash System [19], přičemž o rok později, v roce 2009, zveřejnil implementaci tohoto decentralizovaného elektronického platebního systému. Jedná se o platební systém, jehož hlavním prvkem je neměnná, decentralizovaná účetní kniha transakcí, která se skládá ze za sebou jdoucích bloků transakcí svázaných pomocí hašů. Tento systém využívá svou vlastní logickou peer-to-peer síť, založenou na Bitcoin protokolu, ke komunikaci jednotlivých síťových uzlů, které *zveřejňují*, *zaznamenávají* a *verifikují* transakce. Takovýto síťový uzel může být buďto *úplným uzlem* (angl. full node), který si spravuje vlastní kopii celého blockchainu a účastní se konsensuálního mechanismu založeném na proof-of-work (PoW) principu, anebo *lehkým klientem* (angl. lightweight client, nebo také simplified-payment-verification client), který se připojuje k úplnému uzlu a slouží pouze k částečnému ověření transakcí a ke zveřejňování nových transakcí do sítě.

Kapitola 2.1 se zaměřuje na představení kryptografických primitiv, které jsou v rámci systému Bitcoin využívány. Kapitola 2.2 se věnuje Blockchainu, jakožto hlavnímu objektu zájmu této diplomové práce. Dalšími důležitými stavebními bloky systému Bitcoin jsou jeho konsensuální mechanismus založený na proof-of-work principu a Bitcoin protokol, jenž slouží jako základ pro logickou peer-to-peer síť, kterou tento systém využívá. Konsensuální mechanismus a Bitcoin protokol jsou však mimo rozsah této práce, jelikož pro její pochopení nejsou potřebné. Kapitola 2.3 se věnuje forenzní analýze blockchainu.

2.1 Kryptografická primitiva

Jak již slovo kryptoměna napovídá, bezpečnost kryptoměn je založena na využití kryptografie. Tato kapitola nastiňuje základní kryptografická primitiva, která tvoří základní stavební bloky systému kryptoměny Bitcoin, ať už pro zajištění *integrity dat*, *autorizaci transakcí* či k provozování jejího *konsensuálního protokolu*.

2.1.1 Kryptografická hašovací funkce

Kryptografická hašovací funkce (angl. hash function) je taková funkce $F : \{0,1\}^* \to \{0,1\}^n$, která splňuje následující podmínky:

1. F je aplikovatelná na argument libovolné délky,
2. její výstupní hodnota má fixní délku (např. 128, 160 či 256 bitů),

3. pro dané y je výpočetně nezvládnutelné nalézt takové x, že $F(x) = y$ (anglicky tzv. first pre-image resistance),

4. pro dané x je výpočetně nezvládnutelné nalézt takové x', že $F(x) = F(x')$ (anglicky tzv. second pre-image resistance),

5. je výpočetně nezvládnutelné nalézt takové x a x', aby platilo $x \neq x' \wedge F(x) = F(x')$ (anglicky tzv. collision resistance),

přičemž výstup hašovací funkce se nazývá haš (angl. hash), který unikátně reprezentuje vstupní data [31]. Z tohoto důvodu bývá haš často využíván pro zajištění integrity dat, jelikož jakákoli změna ve vstupních datech způsobí změnu i v rámci haše těchto dat, a tedy haš útočníkem upravené zprávy nebude odpovídat původnímu haši. Zde je však zapotřebí dodat, že haš nechrání před neoprávněnou změnou sám sebe. Proto, aby mohl být použit jako mechanismus pro zajištění integrity dat, je potřeba také zajistit integritu haše samotného. Toho lze docílit přidáním tajného klíče ke vstupním datům, kdy útočník nemá možnost změnit data a následně i jejich haš bez znalosti tohoto tajného klíče, čímž je zajištěna jejich integrita. Tento mechanismus se pak nazývá klíčovaný haš (angl. Hash-based Message Authentication Code — HMAC). Podrobné vysvětlení tohoto mechanismu je mimo rozsah této práce, jelikož tento typ haše není v rámci kryptoměny Bitcoin využíván.

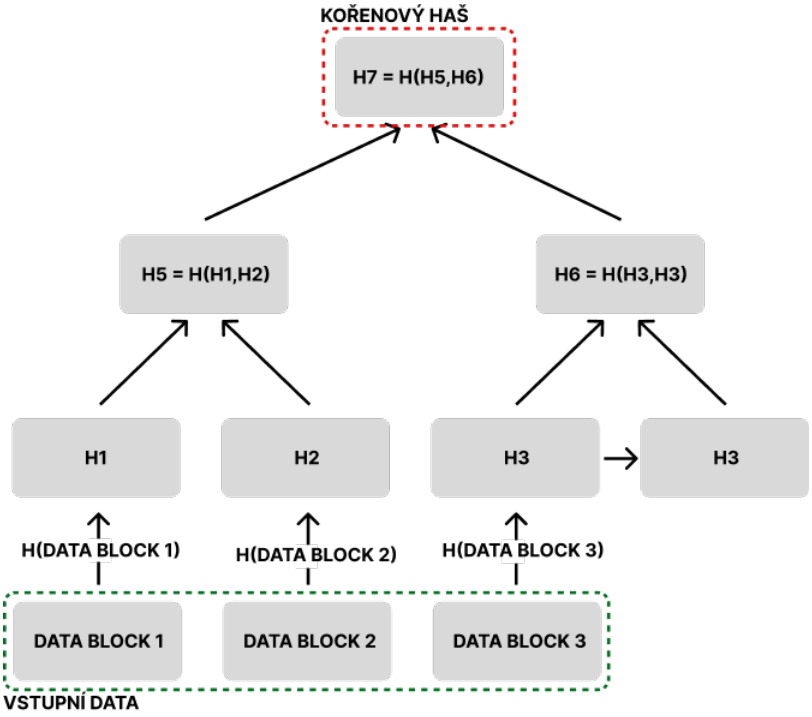
2.1.2 Merkleův strom

Jedním z nejdůležitějších kryptografických primitiv využitých v systému kryptoměny Bitcoin jsou Merkleovy stromy [18] (nebo také hašové stromy), které jsou použity pro zajištění integrity uložených transakcí v rámci blockchainu sítě Bitcoin. Jedná se o datovou strukturu — strom, přičemž každý uzel v tomto stromu nese hodnotu haše. Listové uzly nesou hodnotu haše jednoho ze vstupních datových bloků, nelistové uzly nesou hodnotu haše vytvořeného z hašů, které nesou jejich potomci. Kořenový uzel tohoto stromu pak nese hodnotu haše, který je tvořen všemi vstupními datovými bloky, a tedy může sloužit pro zajištění jejich integrity. Tento haš se nazývá *kořenový haš* (angl. root hash nebo také master hash). Všechny listové uzly se pak nacházejí na stejné úrovni. Dále, pokud nastane situace, kdy má nějaký uzel méně potomků než je v daném stromu maximum (v případě binárního stromu tedy pouze jednoho potomka), tak se jeden z jeho potomků zreplikuje, aby bylo docíleno úplného zaplnění. Příklad této situace lze vidět na obrázku 2.1.

2.2 Blockchain

Blockchain (česky také blokořetěz[1]) je decentralizovaná databáze transakcí, která je replikována na každém z uzlů sítě Bitcoin. Úplná kopie této databáze obsahuje všechny transakce, které byly v rámci sítě Bitcoin vykonány. Síť Bitcoin je díky tomu transparentní, jelikož každý její uživatel má možnost získat úplnou kopii jejího blockchainu a následně ověřit jakoukoliv transakci či spočítat hodnotu odpovídající nějaké specifické adrese v konkrétním čase. Blockchain se skládá ze za sebou jdoucích bloků, svázaných pomocí SHA-256 hašů, kdy hlavička každého bloku obsahuje haš hlavičky předchozího bloku[2] (ilustrace tohoto spojení do lineárního řetězce bloků je vizualizována na obrázku 2.2), což zajišťuje integritu celého blockchainu a zároveň společně s pravidlem nejdelšího řetězce zajišťuje nezvratnost

[1] V rámci této práce považuji slovo „blockchain" jako slovo přejaté do češtiny, které je dnes běžně užíváno.
[2] Přesněji se jedná o haš haše hlavičky předcházejícího bloku, tedy SHA256 (SHA256 ($BlockHeader$))

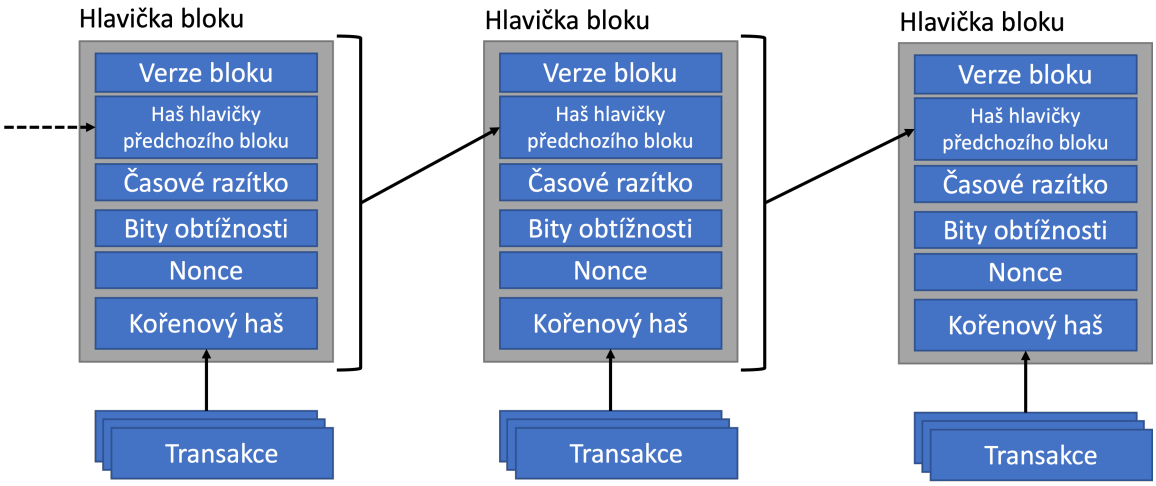


Obrázek 2.1: Ilustrace binárního Merkleova stromu. Haše jednotlivých vstupních datových bloků jsou na každé úrovni stromu po dvojicích kombinovány až do nejvyšší úrovně, kde vznikne kořenový haš, který je závislý na všech vstupních blocích. Obrázek dále ilustruje situaci lichého počtu uzlů na listové úrovni, kdy musí dojít k replikaci posledního uzlu $H3$, aby mohl vzniknout haš vyšší úrovně.

transakcí. Jediným blokem, který neobsahuje haš hlavičky předcházejícího bloku, je blok číslo 0, tzv. *blok geneze* (angl. genesis block), jenž je počátkem tohoto řetězce bloků a je pevně zakódován do softwaru systému kryptoměny Bitcoin [2]. Každý takový blok obsahuje dále samotné transakce, jejichž integrita je zajištěna za pomoci kořenového haše Merkleova stromu (viz kapitola 2.1.2), uloženého v hlavičce bloku, vytvořeného ze všech transakcí, které tento blok obsahuje.

2.2.1 Blok

Každý blok v blockchainu je tvořen *hlavičkou*, *transakcemi* a *metadaty*, jak lze vidět v tabulce 2.1. Samotná hlavička se pak skládá z 6 položek — *verze*, *haše* předcházejícího bloku, *kořenového haše* Merkleova stromu založeného na všech transakcích obsažených v daném bloku, *časového razítka*, kdy byl blok vytvořen, aktuálního *obtížnostního cíle*, který musí haš hlavičky splnit, aby byl blok platný, a *nonce*. Přesný formát hlavičky společně s velikostmi jednotlivých polí lze vidět v tabulce 2.2. První transakcí v každém bloku je tzv. *coinbase transakce*, přičemž tato transakce nemá žádné vstupy a její výstupy jsou odměnou pro těžaře, kterému se povedlo tento blok vytvořit. Dále jsou transakce seřazeny tak, aby v případě, kdy nějaká transakce využívá na svém vstupu výstupu jiné transakce obsažené v tom samém bloku, byla transakce, jejíž výstup je využit, zařazena před transakcí, která tento výstup spotřebovává. Formátu samotných transakcí se dále věnuje kapitola 2.2.2.



Obrázek 2.2: Obrázek ilustruje svázání jednotlivých za sebou jdoucích bloků za pomoci haše hlavičky předchozího bloku. Dále lze na obrázku vidět kořenový haš, který je součástí hlavičky každého bloku a zajišťuje integritu transakcí uložených v daném bloku.

Pro výpočet kořenového haše všech transakcí obsažených v bloku je využit binární Merkleův strom (jak byl představen v kapitole 2.1.2), přičemž každá transakce tvoří jeden datový vstupní blok. Jinými slovy, listy Merkleova stromu tvoří haše každé z transakcí obsažených v bloku, což odpovídá jejich identifikátoru *TXID*, a kořenový haš obsažený v hlavičce bloku odpovídá kořenovému haši tohoto stromu. Jako hašovací funkce je využito dvojitého aplikování hašovací funkce SHA-256, tedy $H(x) = \text{SHA-256}\,(\text{SHA-256}\,(x))$, a haše potomků jsou kombinovány za pomocí spojení bytových posloupností jednotlivých hašů, tedy $H(x, y) = \text{SHA-256}(\text{SHA-256}(x||y))$, kde $||$ značí bitovou konkatenaci.

Pole	Popis	Velikost
magické číslo	vždy nabývá hodnoty 0xD9B4BEF9	4 byty
velikost bloku	počet následujících bytů bloku	4 byty
hlavička bloku	viz tabulka 2.2	80 bytů
počet transakcí	počet transakcí v bloku	1 – 9 bytů
transakce	neprázdný seznam transakcí	variabilní

Tabulka 2.1: Struktura bloku Bitcoin blockchainu

2.2.2 Transakce

V rámci blockchainu sítě Bitcoin můžeme rozlišovat dva typy transakcí — tzv. *coinbase transakce*, které tvoří odměnu pro těžaře za vytvoření bloku, a *klasické transakce*, které slouží k přesunu jednotek bitcoinu mezi uživateli. Tato kapitola se dále věnuje převážně klasickému typu transakcí, jejichž klasifikací a zpracováním se zabývá tato práce.

Datová struktura transakce je uvedena v tabulce 2.3. Každá transakce se skládá z informace o její verzi, seznamu vstupních transakcí, seznamu výstupních transakcí a časového zámku. Transakce může dále obsahovat i údaje o svědectví — witness data, což je indikováno přítomností příznaků, které dnes v takovém případě vždy odpovídají hodnotě 0x0001. Witness data byla přidána v roce 2017 v rámci Segregated Witness (SegWit) forku sítě [7],

Pole	Popis	Velikost
verze	číslo verze bloku	4 byty
haš předcházejícího bloku	SHA-256(SHA-256($BlockHeader$))	32 bytů
kořenový haš	kořenový haš Merkleova stromu	32 bytů
časové razítko	počet sekund od počátku Unix epochy	4 byty
bity obtížnosti	aktuální obtížnostní cíl v kompaktním formátu	4 byty
nonce	32-bitové číslo, začíná se na 0	4 byty

Tabulka 2.2: Struktura hlavičky bloku Bitcoin blockchainu.

a proto je jejich přítomnost v transakcích volitelná. Cílem této změny bylo umožnění oddělení informací potřebných k ověření vstupní transakce, jako jsou podpisy a skripty, od ostatních údajů, které jsou potřebné k určení dopadu dané transakce na celý decentralizovaný systém. Jinými slovy, šlo o rozlišení mezi daty potřebnými pro ověřovací proces a daty určujícími aktuální stav sítě. Oddělení těchto dat dále umožnilo vytvořit řetězce závislosti nepotvrzených transakcí, což vytvořilo podmínky pro vznik sítí druhé vrstvy, jako je Lightning Network.[3]

Každá transakce je identifikována jednoznačným identifikátorem TXID, který odpovídá haši serializované transakce bez příznaků a witness dat. TXID transakcí jsou následně využita pro konstrukci Merkleova stromu a vypočtení kořenového haše, který zajišťuje integritu uložených transakcí v rámci blockchainu sítě Bitcoin. Dále každá transakce má WTXID, které odpovídá haši celé transakce a chrání tak integritu i witness dat, která z důvodů zpětné kompatibility nejsou chráněna pomocí TXID. WTXID transakcí jsou stejně jako TXID využity pro konstrukci Merkleova stromu a výsledný WTXID kořenový haš je uložen v rámci skriptu coinbase transakce. WTXID coinbase transakce odpovídá samým nulám, aby bylo možné vypočítat WTXID kořenového haše bez znalosti haše coinbase transakce [7].

Pole	Popis	Velikost
verze	verze transakce	4 byty
příznaky	volitelné, vždy 0x0001, značí přítomnost witness dat	0-2 byty
počet vstupů	počet vstupních transakcí	1-9 bytů
list vstupů	seznam vstupních transakcí	variabilní
počet výstupů	počet výstupních transakcí	1-9 bytů
list výstupů	seznam výstupních transakcí	variabilní
witness data	witness ke každému vstupu, pokud je nastaven witness příznak	variabilní
časový zámek	výška bloku nebo časové razítko, kdy je transakce považována za finální	4 byty

Tabulka 2.3: Struktura transakce po aktualizaci SegWit. Příznaky a witness data jsou z důvodu zpětné kompatibility volitelné, avšak jejich použití vede k menšímu poplatku za danou transakci.

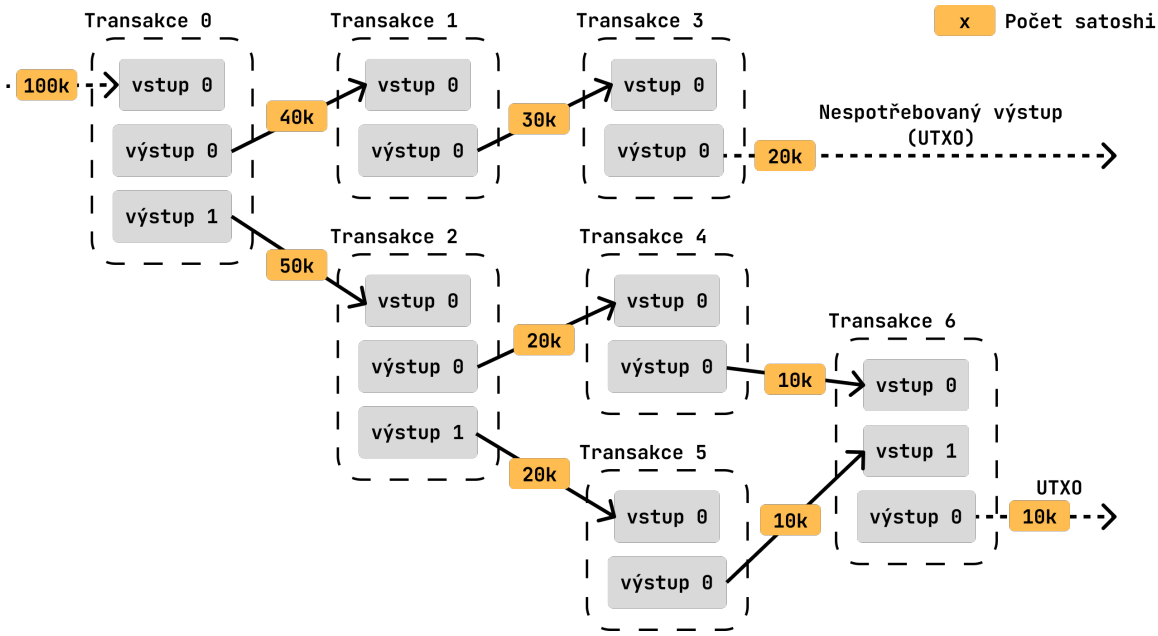
Každá transakce má své vstupy a své výstupy, přičemž součet všech jednotek bitcoinů na vstupu musí být vždy větší nebo roven jednotkám na výstupu (jedinou výjimkou je

[3] https://lightning.network/

coinbase transakce). Nevyužité jednotky bitcoinů ze vstupů slouží jako odměna pro těžaře, který vytvořil blok obsahující tuto transakci.

Autorizace transakcí je v síti Bitcoin zajištěna za pomoci skriptovacího jazyka *Bitcoin Script* v kombinaci s *digitálním podpisem*. Jazyk Bitcoin Script je založen na práci se zásobníkem, nedovoluje smyčky, skoky a ani rekurzivní volání, a tudíž není Turingovsky úplný. Jednotlivé nespotřebované výstupy transakcí, tzv. *UTXO* (angl. unspent transaction output), jsou uzamčeny skriptem *scriptPubKey*. Pro využití takového výstupu je potřeba v rámci transakce, jež jej spotřebovává, poskytnout skript *scriptSig*, který je po kombinaci se *scriptPubKey* interpretován, a pouze v případě, kdy se po jeho interpretaci nachází na zásobníku číslo 1, je použití tohoto výstupu platné. Každý výstup lze spotřebovat pouze jednou. Graf transakcí lze vidět na obrázku 2.3.

V současné době z bezpečnostních důvodů tyto skripty nemohou být libovolné a musí nabývat předem specifických forem. *Bitcoinové adresy*, na které uživatelé posílají prostředky, jsou pouze uživatelsky přívětivým zakódováním použité formy skriptu (pay-to-public-key-hash P2PKH, pay-to-script-hash P2SH, a další) společně s potřebnými daty, jako je haš veřejného klíče u P2PKH nebo haš skriptu u P2SH, a kontrolním součtem pro minimalizaci chyby uživatele při zadávání cílové adresy.



Obrázek 2.3: Ilustrace grafu transakcí kryptoměny Bitcoin. Vstupem každé z transakcí jsou výstupy předchozích transakcí, přičemž každý z výstupů může být využit maximálně jednou.

2.3 Možnosti analýzy blockchainu pomocí neuronových sítí

Tato kapitola se věnuje různým možnostem využití neuronových sítí pro forenzní analýzu blockchainu sítě Bitcoin. Dále představuje dostupné datové sady, které je možné použít pro trénování neuronových sítí k danému forenznímu účelu.

Vícevrstvého perceptronu (MLP) společně s algoritmem Random Forest využili Lee a spol. [14] pro klasifikaci transakcí na legální a nelegální, přičemž za nelegální transakce

považovali transakce spojené s obchody na darknet tržišti Silk Road. Vícevrstvý perceptron, společně s rekurentní neuronovou sítí LSTM (angl. long short-term memory) využitými pro vytvoření auto-encoder architektury k zachycení časových informací, pro klasifikaci adres na adresy bezúhonné a adresy spojené s ilegálními aktivitami, převážně s ransomwarem a vydíráním, využívají Li a spol. [15]. Obě tyto práce nevyužívaly grafové informace a spoléhaly se u klasifikace pouze na příznaky získané z lokálních vlastností transakcí (respektive adres).

Weber a spol. [28] se pokusili adresovat problém nevyužívání grafových informací pro detekci transakcí spojených s ilegálními aktivitami, a to představením datové sady `Elliptic` a vyhodnocením různých metod strojového učení a grafových konvolučních sítí na této datové sadě. Nejlepší metodou pro klasifikaci na této datové sadě byl vyhodnocen algoritmus Random Forest, to však ale pouze v případě, kdy bylo využito i expertem ručně vytvořených příznaků z okolí dané transakce. Grafové neuronové sítě se zde ukázaly jako metoda, která má potenciál tyto expertem vytvořené příznaky nahradit. Využitím grafových konvolučních sítí společně s časovou informací o transakcích obsažených v datové sadě `Elliptic` se zabývali Pareja a spol. [21] a představili novou architekturu grafové neuronové sítě EvolveGCN, která využívá kombinace grafové konvoluční sítě s rekurentní neuronovou sítí LSTM nebo GRU. Alarab a spol. [1] představili grafovou neuronovou síť temporal-GCN, která také kombinuje grafovou konvoluční síť s rekurentní neuronovou sítí LSTM, přičemž pro její trénování využívají aktivního učení.

Chai a spol. [4] zdůraznili problém, že grafové neuronové sítě se z pohledu grafové spektrální analýzy chovají jako filtry dolní propustnosti, které dobře zachytávají vlastnosti rozprostřené mezi více sousedními uzly, tedy vlastnosti, v nichž si jsou sousední uzly podobné, avšak nedokážou dobře pracovat s vlastnostmi uzlů, které se výrazně liší od svého okolí, tedy s abnormalitami, za které transakce spojené s ilegální činností považujeme. Chai a spol. se tento problém snaží adresovat v rámci jimi navržené architektury Adaptive Multi-frequency filtering graph neural network for graph anomaly detection (AMNet), která cílí na zachycení jak nízkofrekvenčních, tak vysokofrekvenčních signálů a na jejich adaptivní kombinaci do výsledného vektoru vnoření za pomoci attention mechanismu.

Duan a spol. [5] se pokusili tento problém adresovat v rámci jimi navržené architektury DGA-GNN, kdy představili dynamic grouping aggregation mechanizmus, kdy jsou vytvořeny 3 skupiny uzlů — všechny uzly, uzly s vysokou mírou pravděpodobnosti bezúhonné a uzly s vysokou mírou pravděpodobnosti spojené s ilegální aktivitou. Každá z těchto skupin je poté nezávisle na sobě agregována, za pomoci grafové neuronové vrstvy inspirované architekturou GraphSAGE, do vektoru příznaků dané skupiny. Tyto vektory jsou následně pro každý uzel agregovány do výsledného vektoru vnoření uzlu s využitím jednoduchého attention mechanismu založeného na vícevrstvém perceptronu.

Dále se detekci transakcí spojených s ilegální činností věnují Wang a spol. [26], kteří v rámci své práce představili rámec RMGANets,[4] který je založen na posilovaném učení společně s attention mechanismem, přičemž tato architektura dle slov autorů dosahuje ke konci roku 2024 state-of-the-art výsledků.

Využitím *self-supervised* učení pro tvorbu vnoření uzlů se zabývá práce Lo a spol. [17], kdy představují rámec *Inspection-L*, který využívá *self-supervised* metodu *Deep Graph Infomax* (DGI) [25] společně se zapojením grafové neuronové sítě *Graph Isomorphism Network* (GIN) [30] jako *enkodéru* pro obohacení příznaků uzlů v transakčním grafu, a takto nově získané příznaky poté klasifikují za pomoci *Random Forest* klasifikátoru, čímž dosahují na datové sadě `Elliptic` lepších výsledků než všechny dříve zmíněné metody.

[4] V rámci této práce byli kontaktování autoři, aby poskytli datovou sadu společně se zdrojovými kódy své architektury k ověření jejich výsledků, avšak dodnes neposkytli odpověď.

V neposlední řadě je třeba zmínit práci Song a spol. [23] a jimi představenou architekturu *Social Attention Graph Neural network* (SGNN). Song a spol. navazují na práci Elmougy a Liu [6], kteří argumentují, že při boji proti praní špinavých peněz je lepší než samotné transakce klasifikovat adresy, a poskytují otevřenou datovou sadu aktérů `Elliptic++`, která vychází z označených transakcí z datové sady `Elliptic`. Song a spol. dále tvrdí, že se při klasifikaci transakčního grafu ztrácí sociální informace z adres zapojených do daných transakcí, kdy dále rozlišují *transakční spojení* a *spojení identity*. *Transakční spojení* v tomto kontextu označují vztah mezi adresami a transakcemi, tedy informaci o tom, které adresy se účastní kterých transakcí. Naproti tomu *spojení identity* reprezentují vztahy mezi jednotlivými adresami, které se společně vyskytují na vstupu nějaké z transakcí. Tato skutečnost naznačuje, že adresy pravděpodobně patří stejnému uživateli, protože k podepsání takové transakce je třeba disponovat odpovídajícími soukromými klíči. Tento princip je známý jako *Common Spend* heuristika, běžně využívaná při shlukování adres. Song a spol. klasifikují adresy na legální a ilegální. Následně využívají Random Forest klasifikátoru ke klasifikaci samotných transakcí v transakčním grafu, přičemž příznaky jednotlivých transakcí extrahují z predikovaných tříd zúčastněných adres. V oblasti klasifikace transakcí dosahuje v současnosti architektura Social Attention Graph Neural Network (SGNN) nejlepších výsledků.

Zatímco všechny dosud představené modely pracovaly na problému klasifikace jednotlivých transakcí, Bellei a spol. [3] představují nový problém, a to klasifikaci podgrafů v grafu entit, tedy v grafu, kde uzly reprezentují shluky adres a hrany reprezentují transakce mezi nimi. Samotné podgrafy se pak dělí na podgrafy bezúhonné a podgrafy vykazující známky praní špinavých peněz, přičemž podle Bellei a spol. tento problém více odpovídá regulačním požadavkům proti praní špinavých peněz. Bellei a spol. představují novou největší datovou sadu pro klasifikaci podgrafů na světě — `Elliptic2`, přičemž na této datové sadě zkouší dostupné metody klasifikace podgrafů a s modelem neuronové sítě GLASS [27] dosahují nejlepších výsledků klasifikace podgrafů, které významně převyšují výsledky ostatních použitých modelů GNN-Seg a Sub2vec. Hu a spol. [12] se věnují více třídní klasifikaci podgrafů transakčního grafu s cílem identifikovat služby mixérů, přičemž k tomu využívají kombinace graph2vec [20] a vícevrstvého perceptronu.

Hlavními případy užití neuronových sítí pro forenzní analýzu blockchainu sítě Bitcoin tedy jsou — *detekce transakcí spojených s ilegální aktivitou*, *detekce praní špinavých peněz* pomocí klasifikace podgrafů, *identifikace služeb mixérů*, také jako problém klasifikace podgrafů transakčního grafu sítě Bitcoin, a dále problém predikce spojení v transakčním grafu k *deanonymizaci transakcí*, kterému se věnuje práce Li a He [16], avšak u které není zcela zřejmé využití ve forenzní praxi.

Dostupné datové sady

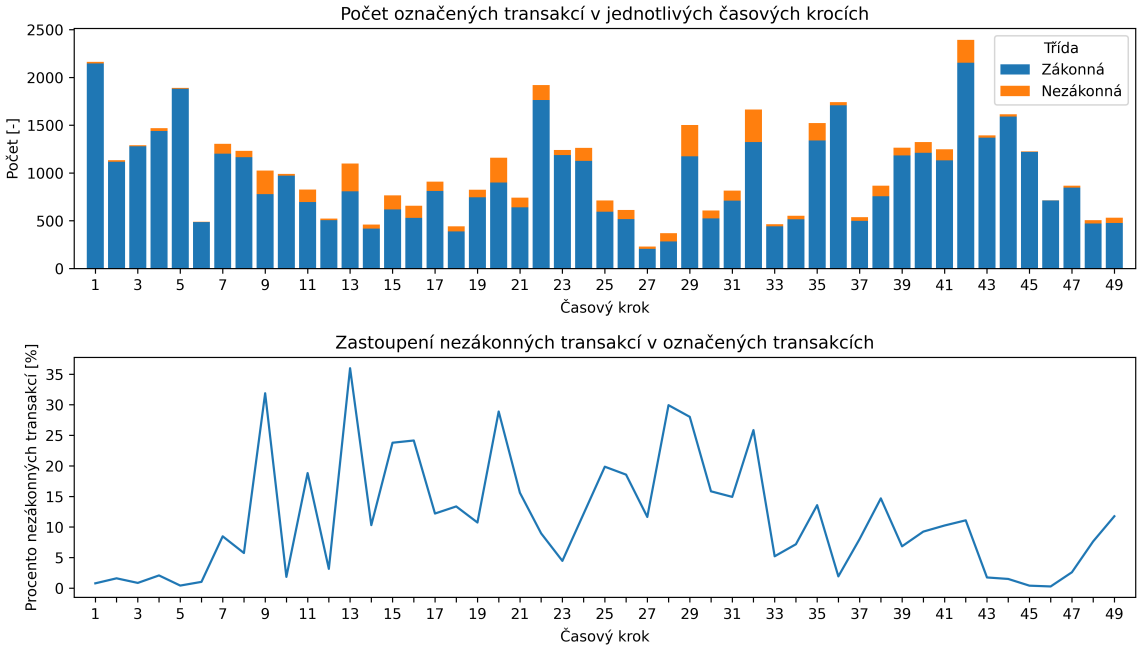
V současné době existuje několik dostupných datových sad, které lze využít pro trénování neuronových sítí za účelem forenzní analýzy blockchainu sítě Bitcoin. Nejpoužívanější datovou sadou je sada `Elliptic`, kterou lze využít pro trénování klasifikátoru transakcí. Dále se jedná o datovou sadu `Elliptic++`, která vedle označeného transakčního grafu obsahuje i sadu aktérů, kterou lze využít k analýze chování samotných bitcoinových adres. V neposlední řadě se jedná o největší datovou sadu označených podgrafů `Elliptic2` a datovou sadu `BitcoinTemporalGraph`, která obsahuje graf shluků adres a interakce mezi nimi.

Datová sada `Elliptic`

Datová sada `Elliptic` byla představena v Weber a spol. [28]. Graf datové sady `Elliptic` se skládá z 49 nespojených podgrafů, kdy každý z těchto podgrafů reprezentuje určitý časový úsek v síti, úseky jsou chronologicky seřazeny, jednotlivé úseky jsou od sebe vzdáleny asi 2 týdny a každý úsek obsahuje transakce, které byly v účetní knize zaznamenány během tříhodinového okna.

Transakce jsou reprezentovány uzly těchto grafů a hrany reprezentují tok prostředků mezi transakcemi, tedy hrana $(u, v) \in V$ značí, že transakce v využívá na svém vstupu výstup transakce u. Jedná se tedy o orientovaný graf transakcí. Datová sada `Elliptic` mapuje tyto transakce k entitám reálného světa, a to buďto k legálním službám, jako jsou kryptoměnové směnárny nebo těžaři, anebo k entitám spojeným s nelegální činností, jako je praní špinavých peněz, podporování terorismu nebo provozování darknet tržiště.

Konkrétně se tato datová sada skládá z 203 769 uzlů a 234 355 orientovaných hran, z čehož 2 % (4 545) tvoří transakce spojené s ilegálními aktivitami a 21 % (42 019) jsou označeny jako legitimní. Počet označených transakcí v jednotlivých časových krocích je zobrazen v grafu na obrázku 2.4. Jedná se tedy o částečně označenou datovou sadu, která je určena k trénování modelů strojového učení ke klasifikaci transakcí na ty spojené s ilegálními aktivitami a na ty legitimní.



Obrázek 2.4: Grafy zobrazující zastoupení označených transakcí v jednotlivých časových krocích datové sady `Elliptic`.

Ke každému uzlu grafu je přiřazen vektor 166 příznaků, přičemž prvních 94 příznaků odpovídá lokálním informacím o dané transakci, jako je *počet vstupů*, *počet výstupů*, *poplatek* a *objem transakce*. Dále, jak se podařilo zjistit v této práci, obsahují lokální informace i agregované vlastnosti adres zapojených do dané transakce, konkrétně se jedná například o *životnost dané adresy*, počet transakcí, kde adresa *vystupuje na vstupu*, a počet transakcí, kde adresa *vystupuje na výstupu*. Zbylých 72 příznaků odpovídá agregovaným informacím

o vstupních a výstupních transakcích této transakce (jeden krok zpět a jeden krok vpřed), přičemž pro agregaci příznaků je většinou využito minima, maxima a průměru.

Na základě zjištění této práce je třeba zdůraznit, že všechny informace o adresách, stejně jako agregované údaje z výstupních transakcí, byly vypočítány ke stavu blockchainu v okamžiku tvorby datové sady, konkrétně k bloku 575 059 (viz kapitola 4.1).

Anonymita jednotlivých příznaků způsobuje v zásadě dva problémy této datové sady:

1. nevysvětlitelnost rozhodnutí natrénovaného modelu,
2. nemožnost vytvořit vektor příznaků pro libovolnou transakci, tedy nemožnost rozšíření datové sady o nové transakce a nemožnost forenzního využití takto natrénovaného modelu pro libovolnou zájmovou transakci.

Datová sada `Elliptic++`

Datová sada `Elliptic++` byla představena v roce 2023 v rámci práce Elmougy a Liu [6]. Vychází z datové sady `Elliptic` a samotná datová sada se skládá ze dvou částí — ze sady transakcí a sady aktérů (angl. Actors dataset). Sada transakcí obsahuje 99,5 % identifikovaných transakcí[5] původní datové sady `Elliptic`, přičemž jejich příznaky jsou rozšířeny o 17 neanonymizovaných lokálních příznaků, což dle tvrzení a experimentů autorů napomáhá robustnosti a vysvětlitelnosti trénovaných klasifikačních modelů. Datová sada transakcí však nijak neadresuje problém, kdy bez znalosti všech příznaků nelze vytvořit vstupní vektor příznaků libovolné transakce, což efektivně znemožňuje využití této sady pro vývoj forenzní aplikace.

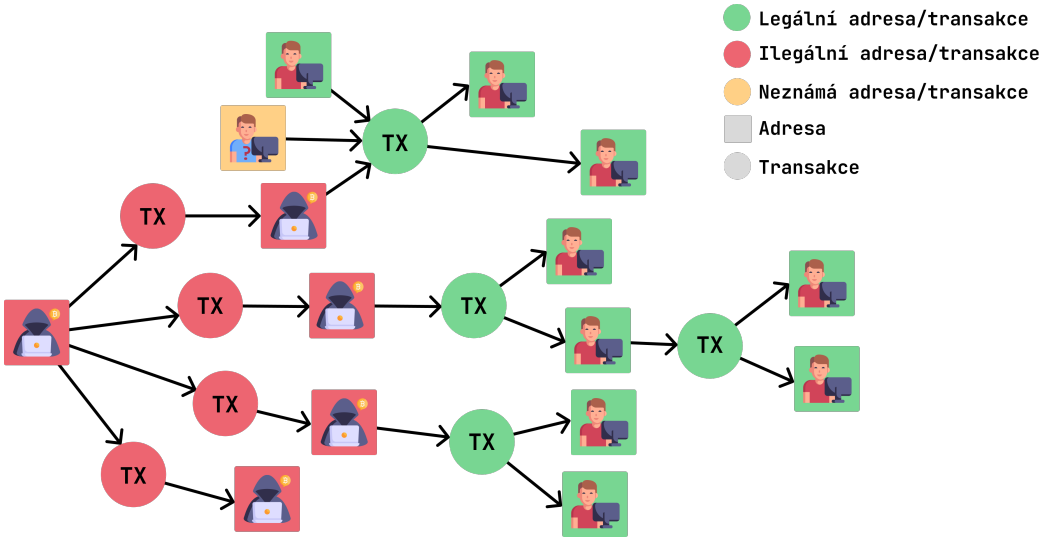
Významnějším přínosem práce Elmougy a Liu [6] je druhá část této datové sady — sada aktérů. Jedná se o heterogenní graf skládající se z 822 924 uzlů reprezentujících adresy a 202 804 uzlů reprezentujících transakce z původní datové sady `Elliptic`. Dále se zde nachází 3 typy hran — hrany mezi dvěma adresami reprezentující interakci mezi nimi v podobě transakce, dále hrany mezi uzlem adresy a uzlem transakce reprezentující výskyt dané adresy na vstupu transakce a nakonec hrany reprezentující výskyt adresy na výstupu transakce. Tento graf lze následně převést jak do *adres-transakčního grafu* (který lze vidět na obrázku 2.5), tak do formy *grafu interakcí aktérů*, kdy se graf bude skládat pouze z uzlů adres a hran mezi nimi.

Uzly adres jsou označeny na základě označení transakcí, se kterými daná adresa sousedí. Pokud sousedí alespoň s jednou transakcí, která je v původní datové sadě označena jako transakce spojená s ilegální aktivitou, tak je i samotná adresa označena jako adresa spojená s ilegální aktivitou. Dále, pokud je poměr sousedících legálních transakcí a sousedících neoznačených transakcí větší než tento poměr v celé datové sadě, je uzel adresy označen jako legální. V opačném případě není označen. Celkově je označeno 32,2 % adres, kdy 1,7 % tvoří adresy spojené s ilegální aktivitou a 30,5 % tvoří adresy označené jako legální.

Datová sada `Elliptic2`

Datová sada `Elliptic2` byla představena v roce 2024 v práci Bellei a spol. [3] jako datová sada pro trénování modelů strojového učení k identifikaci kryptoměnových prostředků pocházejících z ilegálních aktivit, což vyžadují regulační požadavky proti praní špinavých peněz v mnoha zemích světa. Tento problém reprezentují jako problém klasifikace podgrafů

[5] https://www.kaggle.com/datasets/alexbenzik/deanonymized-995-pct-of-elliptic-transactions



Obrázek 2.5: Ilustrace adres-transakčního grafu. Uzly transakcí jsou na obrázku reprezentovány jako kruh a uzly jednotlivých adres jako čtverce. Hrany mezi těmito uzly představují participaci dané adresy v dané transakci, a to v případě hrany směřující do transakce na jejím vstupu, respektive u hrany směřující z uzlu transakce na jejím výstupu.

v grafu entit, kdy se v reálném případě bude na konci tohoto podgrafu jednat o nějakou legitimní službu, například kryptoměnovou směnárnu, která se těmito regulačními požadavky musí řídit.

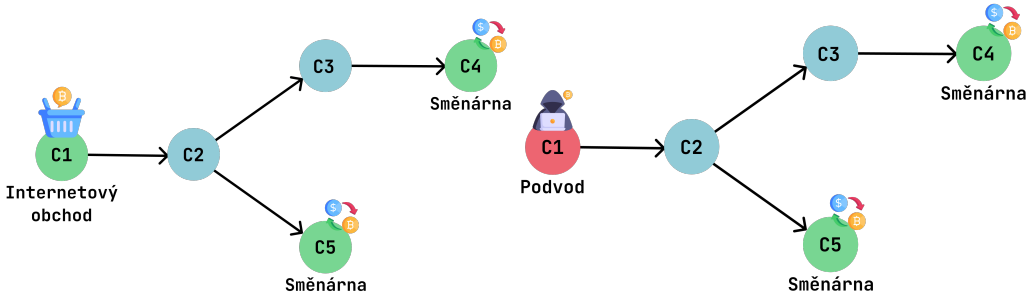
Tato datová sada zahrnuje 49 milionů uzlů reprezentujících shluky adres a 196 milionů hran, které znázorňují transakce mezi nimi. Každý uzel je popsán 43 příznaky, mezi něž patří například velikost shluku či počet realizovaných transakcí. Hranám náleží 95 příznaků, zahrnujících mimo jiné objem transakce, výši poplatku a časové razítko. Jedná se stejně jako u první verze této datové sady o anonymizovanou datovou sadu, přičemž jednotlivé příznaky uzlů a hran jsou rozřazeny do košů, aby byla znemožněna rekonstrukce původních příznaků.

V rámci označování této datové sady autoři definovali 4 druhy cest mezi dvěma uzly. Pokud se jedná o cestu z legálního uzlu do legálního uzlu, jedná se o *legální cestu*. V opačném případě, kdy se jedná o cestu z nelegálního uzlu do nelegálního, se jedná o *cestu nelegální*. V případě cesty z nelegálního uzlu do legálního se jedná o *podezřelou cestu*. V situaci, kdy cesta končí v neoznačeném uzlu, se jedná o cestu *neutrální*. Na základě toho pak `Elliptic2` rozlišuje 2 typy podgrafů:

- legální podgraf — podgraf, který obsahuje alespoň jednu legální cestu a dále pouze neutrální cesty (jak lze vidět na obrázku 2.6a),
- podezřelý podgraf — podgraf, který obsahuje alespoň jednu nelegální či podezřelou cestu a žádnou cestu legální (jak lze vidět na obrázku 2.6b).

Datová sada `BitcoinTemporalGraph`

Datová sada `BitcoinTemporalGraph` je největší datovou sadou svého druhu, která byla zveřejněna v práci Schnoering a spol. [22]. Jedná se o částečně označenou datovou sadu reálných entit (organizací, jednotlivců nebo institucí) reprezentovaných uzly grafu a orientovanými hranami mezi nimi, reprezentujícími jejich interakce, přesun prostředků v podobě



(a) Legální podgraf (b) Podezřelý podgraf

Obrázek 2.6: Ilustrace (a) legálního podgrafu a (b) podezřelého podgrafu z datové sady `Elliptic2`. Podgraf vlevo (a) obsahuje legální cestu z uzlu reprezentujícího shluk adres internetového obchodu ke shluku adres kryptoměnové směnárny, a proto je označen jako *legální*. Podgraf vpravo (b) obsahuje podezřelou cestu z ilegálního shluku adres, které byly využity v rámci nějakého podvodu, do legálního uzlu (směnárny), a proto je označen jako *podezřelý*, protože se pravděpodobně jedná o pokus o praní špinavých peněz.

transakcí v rámci blockchainu sítě Bitcoin. Tento graf se pak skládá z 252 milionů uzlů a 785 milionů hran mezi nimi, přičemž 34 098 uzlů je označeno typem dané entity. Konkrétně tato datová sada rozlišuje 11 typů entit — jednotlivec, sázky, ransomware, hazard, kryptoměnová směnárna, těžba, Ponziho schéma, tržiště, faucet (propagační nástroj, který odměňuje uživatele malým množstvím bitcoinů za splnění nějakého úkolu nebo shlédnutí reklamy), most (protokol, který zajišťuje směnu aktiv mezi sítí Bitcoin a jinou kryptoměnovou sítí) a v neposlední řadě mixér.

Na rozdíl od všech tří předchozích datových sad, kde byly uzly, respektive podgrafy, označeny na základě označení shluků adres společnosti `Elliptic Enterprises Limited`, v tomto případě bylo označení provedeno plně automaticky za využití velkého jazykového modelu (LLM) GPT-4o-mini v podobě využití služby ChatGPT[6] od společnosti OpenAI.

Samotná označení uzlů (reálných entit) byla odvozena z označení bitcoinových adres, které autoři získali analýzou dat z diskuzního fóra BitcoinTalk.[7] Pomocí techniky web scrapingu shromáždili přibližně 14 milionů příspěvků, ve kterých vyhledávali výskyty bitcoinových adres. Pro každou nalezenou adresu následně s využitím velkého jazykového modelu analyzovali její textový kontext a přiřadili jí štítek reprezentující typ reálné entity. Je důležité podotknout, že sami autoři uvádějí, že se nejedná o bezchybný proces a že extrakce označení adres má přesnost mezi 83 % až 98 %, v závislosti na typu entity. Tato označení byla dále rozšířena pomocí externích zdrojů, například veřejně dostupných seznamů adres kryptoměnových směnáren, které tyto subjekty zveřejňují jako součást důkazů o rezervách, nebo seznamů adres spojených s různými Ponziho schématy a podvodnými aktivitami.

Na základě těchto označených adres pak autoři identifikovali reálné entity, tedy shluky adres, které tvoří uzly v grafu. Pokud shluk obsahoval alespoň jednu označenou adresu, byl celý shluk označen stejným štítkem. V případech, kdy shluk obsahoval více adres s různými štítky, nebyl označen vůbec, aby byla zachována integrita této datové sady.

Na závěr je dobré zmínit, že se jedná o *otevřenou datovou sadu*, kdy jsou zveřejněny jak všechny příznaky jednotlivých uzlů, tak označení všech adres, ze kterých byla označení uzlů vyvozena.

[6] https://chatgpt.com/
[7] https://bitcointalk.org/

Kapitola 3

Grafové neuronové sítě

Hluboké učení (angl. Deep Learning) se v posledním desetiletí stalo dominantním směrem v rámci výzkumu umělé inteligence. Ačkoliv klasické metody hlubokého učení, například vícevrstvý perceptron (MLP), konvoluční neuronové sítě (KNN) nebo transformery, dosahují velice slibných výsledků u úloh nad euklidovskými daty, jako jsou obrázky nebo sekvence (časové řady, text), mnoho komplexních systémů lze přirozeněji či lépe reprezentovat za pomoci grafových struktur, na které tradiční metody nestačí, jelikož vyžadují pravidelnou strukturu vstupních dat (vstupem vícevrstvého perceptronu je vektor předem dané velikosti, vstupem konvoluční neuronové sítě je tensor dané velikosti). Grafové neuronové sítě, jako nový přístup hlubokého učení pro práci s grafovými strukturami, nabízejí slibné výsledky v rámci mnoha aplikačních domén [29].

Blockchain sítě Bitcoin lze reprezentovat jako graf transakcí, reprezentovaných uzly, a hranami mezi nimi reprezentujícími toky prostředků, jak bylo zmíněno v kapitole 2. Grafové neuronové sítě jsou proto přirozenou volbou přístupu hlubokého učení k analýze blockchainu sítě Bitcoin.

Kapitola 3.1 se věnuje formální definici grafu a na ní navazuje kapitola 3.2, která popisuje základní principy využité v grafových neuronových sítích. Následující kapitoly se poté zabývají popisem nejvýznamnějších typů těchto sítí, jako jsou grafové konvoluční sítě [13] (viz kapitola 3.3), architektura *GraphSAGE* [9] (viz kapitola 3.4) a architektura *graph attention network* [24] (viz kapitola 3.5).

3.1 Graf a jeho reprezentace

Graf se skládá z množiny uzlů a množiny hran mezi nimi. Uzly reprezentují entity, hrany reprezentují vztahy mezi těmito entitami. Uzly a hrany formují topologickou strukturu grafu. Kromě samotné struktury lze k jednotlivým uzlům (případně i k hranám či k celému grafu) přiřadit další informace — vektor příznaků uzlu (anglicky node features), respektive vektor příznaků hran a vektor příznaků celého grafu [29].

3.1.1 Formální definice grafu

Formálně je orientovaný graf G definován jako dvojice $G = (V, E)$, kde V značí množinu uzlů a E značí množinu hran. Hranu směřující z uzlu $v \in V$ do uzlu $u \in V$ označujeme jako dvojici $(v, u) \in V \times V$. Hrany takového grafu lze také reprezentovat za pomoci matice sousednosti $A \in \mathbb{R}^{|V| \times |V|}$. K sestavení takové matice je potřeba seřadit uzly, aby index každého uzlu odpovídal konkrétnímu řádku (respektive sloupci) v matici sousednosti, index

uzlu $v \in V$ označíme i_v, potom můžeme existenci hrany mezi uzly $u, v \in V$ reprezentovat následovně: $A[i_v, i_u] = 1 \Leftrightarrow (v, u) \in E$. Pro zjednodušení notace $A[v, u]$ značí $A[i_v, i_u]$. Příznaky uzlů (angl. node features) formálně definujeme jako matici $X \in \mathbb{R}^{|V| \times d}$, kde d je počet příznaků každého uzlu a vektor $\mathbf{x}_u \in \mathbb{R}^d$ pak bude značit vektor příznaků konkrétního uzlu u [10].

3.1.2 Vnoření uzlů, hran a celého grafu

Mezi základní řešené problémy souvisejícími s grafy patří *klasifikace uzlů*, *predikce vztahů mezi nimi* (predikce hran), *shlukování*, *detekce komunit* a v neposlední řadě také *klasifikace, regrese a shlukování nad celými grafy*. Tyto problémy je možné řešit za využití metody učení se grafové reprezentace, jejíž úkolem je naučení se mapovací funkce, která jednotlivé uzly (respektive hrany či celý graf) a jejich příznaky společně se strukturální informací grafu zobrazí do nízkorozměrného prostoru, tzv. prostoru na vnoření (angl. embedding space). Tato funkce tedy daný uzel společně s jeho příznaky převede na nízkorozměrný vektor, tzv. vnoření uzlu (angl. node embedding). Pro takto reprezentované uzly (respektive hrany či grafy) lze následně využít tradičních metod hlubokého učení, jako je vícevrstvý perceptron (MLP), nebo klasických metod strojového učení, jako je například algoritmus k-nejbližších sousedů (kNN). V rámci této práce vektor $\mathbf{z}_u$ značí vnoření uzlu u [10].

3.2 Základní principy grafových neuronových sítí

Tato kapitola představuje základní principy a mechanismy používané pro navrhování, trénování a inferenci grafových neuronových sítí. Nejprve kapitola 3.2.1 představuje *message passing* mechanismus, jako jeden ze způsobů popisu vrstev grafových neuronových sítí, následně kapitola 3.2.2 popisuje základní grafovou neuronovou síť za pomoci message passing mechanismu. *Over-smoothing* problému u hlubokých grafových neuronových sítí, a jak ho vyřešit za pomoci *přeskakujících spojení* (angl. skip-connection), se věnuje kapitola 3.2.4.

3.2.1 Message Passing Framework

Podle Gilmer a spol. [8] a Hamilton [10], spadá většina grafových neuronových sítí do rodiny neuronových sítí předávajících si zprávy (angl. message passing neural networks — MPNNs). Tyto neuronové sítě lze vyjádřit za pomoci funkce předávání zpráv ve tvaru:[1]

$$\mathbf{m}_{N(u)}^{l} = \text{AGGREGATE}^{l}\left(\left\{\text{MESSAGE}\left(\mathbf{h}_{v}^{l}, \mathbf{h}_{u}^{l}, \mathbf{e}_{v}u\right), \forall v \in N(u)\right\}\right) \tag{3.1}$$

$$\mathbf{h}_{u}^{l+1} = \text{UPDATE}^{l}\left(\mathbf{h}_{u}^{l}, \mathbf{m}_{N(u)}^{l}\right), \tag{3.2}$$

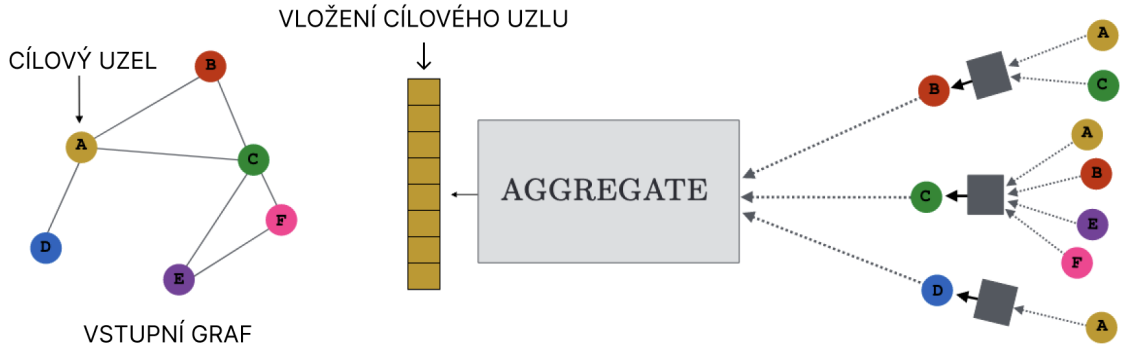
kde UPDATE, MESSAGE a AGGREGATE jsou jakékoliv diferenciovatelné funkce (jako například suma, průměr nebo i neuronová síť) a $\mathbf{m}_{N(u)}^{l}$ je agregovaná zpráva z uzlů ze sousedství uzlu u. Při každé iteraci l grafové neuronové sítě, přičemž každá vrstva grafové neuronové sítě vykonává jednu iteraci message passing mechanismu, agreguje funkce AGGREGATE zprávy ze sousedních uzlů získané pomocí MESSAGE funkce a vygeneruje agregovanou zprávu $\mathbf{m}_{N(u)}^{l}$. Následně je tato zpráva předána funkci UPDATE, která tuto

[1] Původní článek Gilmer a spol. [8] definuje pouze UPDATE funkci a MESSAGE funkci, přičemž výsledná zpráva $m_{N(u)}^{l}$ je sumou výsledků MESSAGE funkce pro každého souseda, avšak zobecnění sumy na funkci AGGREGATE odpovídá práci Hamilton a spol. [9] a dovoluje vyjádřit více neuronových sítí pomocí tohoto frameworku.

agregovanou zprávu nesoucí informace ze sousedství uzlu u zkombinuje s jeho vektorem příznaků $\mathbf{h}_u^l$ a vytvoří tak aktualizovaný vektor příznaků uzlu u — $\mathbf{h}_u^{l+1}$. Pro první iteraci ($l = 0$) odpovídá vektor příznaků vstupnímu vektoru příznaků daného uzlu, tedy $\mathbf{h}_u^l = \mathbf{x}_u$. Po určitém, předem daném počtu iterací K (odpovídá počtu vrstev grafové neuronové sítě) lze využít výstup poslední vrstvy jako výsledný vektor vnoření uzlu, tedy

$$\mathbf{z}_u = \mathbf{h}_u^K, \tag{3.3}$$

který poté může být vstupem do vícevrstvého perceptronu pro účel klasifikace uzlů, nebo využit například pro shlukování. Vizualizaci tohoto procesu pro dvouvrstvou grafovou neuronovou síť lze vidět na obrázku 3.1.



Obrázek 3.1: Ilustrace ukazuje, jak vektor vnoření jednoho cílového uzlu agreguje zprávy z jeho okolí, čímž výsledný vektor vnoření uzlu obsahuje jak informace z jeho vstupních příznaků, tak i informace o struktuře grafu a příznacích sousedních uzlů. Převzato a upraveno z literatury [10].

Podle Hamilton [10], je častým zjednodušením message passing mechanismu situace, kdy jsou do grafu přidány smyčky, a tedy AGGREGATE funkce na svém vstupu má množinu všech $v \in N(u)$ společně s uzlem u. Rovnice pro agregovanou zprávu pak vypadá následovně:

$$\mathbf{m}_{N(u)}^l = \text{AGGREGATE}^l \left(\left\{ \text{MESSAGE} \left(\mathbf{h}_v^l, \mathbf{h}_u^l, \mathbf{e}_v u \right), \forall v \ \in N(u) \cup u \right\} \right), \tag{3.4}$$

kde význam všech symbolů odpovídá těm z rovnice 3.1. UPDATE funkce pak většinou odpovídá prostému přiřazení agregované zprávy za výsledný vektor vnoření cílového uzlu. Toto zjednodušení může grafové neuronové síti pomoci k lepší generalizaci, avšak snižuje to její vyjadřovací schopnost, jelikož informace ze sousedů uzlů nemohou být odlišeny od informací uzlu samotného, což může vést k tzv. over-smoothing problému, kterému se dále věnuje kapitola 3.2.4.

3.2.2 Základní grafová neuronová síť

Za pomoci message passing mechanismu si můžeme nyní definovat základní grafovou neuronovou síť následovně:

$$\mathbf{m}_{N(u)}^l = \sum_{v \ inN(u)} \mathbf{h}_v^l, \tag{3.5}$$

$$\text{UPDATE} \left(\mathbf{h}_u^l, \mathbf{m}_{N(u)}^l \right) = \sigma \left(\mathbf{W}_{self}^l \mathbf{h}_u^l + \mathbf{W}_{neigh}^l \mathbf{m}_{N(u)}^l \right), \tag{3.6}$$

kde $\mathbf{W}_{self}^l$ a $\mathbf{W}_{neigh}^l$ značí učitelné váhové matice vrstvy l a význam ostatních symbolů odpovídá těm z rovnice 3.1 [10].

3.2.3 Normalizace okolí

Ačkoliv by grafová neuronová síť definovaná v kapitole 3.2.2 fungovala, suma všech vektorů příznaků sousedů může způsobit velké rozdíly mezi uzly s malým stupněm a uzly se stupněm velkým, což by dále mohlo způsobit numerickou nestabilitu a složitost trénování takové neuronové sítě. Řešením tohoto problému je normalizace této hodnoty, v nejjednodušším případě bude tedy suma všech vektorů příznaků sousedů vydělena počtem sousedů, suma bude tedy nahrazena průměrem těchto vektorů. Kipf a spol. [13] dále využili symetrické normalizace ve své architektuře grafových konvolučních sítí (více v kapitole 3.3) a Hamilton a spol. [9] ve svých experimentech s GraphSAGE frameworkem vyzkoušeli, vedle průměru a symetrické normalizace, i pooling agregátor založený na vícevrstvém perceptronu a pokročilejší agregátor založený na rekurentní neuronové síti LSTM, jak je dále uvedeno v kapitole 3.4.

Je důležité podotknout, že normalizace není vždy vyžadována a může být i nechtěná, jelikož se s ní ztrácí rozdíl mezi uzly s jedním sousedem a uzly s desítkami sousedů. Obecně se dá říci, že normalizace je většinou užitečná ve chvíli, kdy informace v příznacích uzlů jsou cennější než strukturální informace grafu [10].

3.2.4 Over-smoothing problém

Jedním z častých problémů základních architektur grafových neuronových sítí a sítí využívajících smyček bývá tzv. over-smoothing problém, kdy po několika iteracích grafové neuronové sítě se vektory příznaků jednotlivých uzlů stanou velmi podobnými a od sebe prakticky nerozeznatelnými. Jako možné řešení tohoto problému je využití přeskakujících spojení (angl. skip connections), které nám dovolí zachovat informace ze vstupního vektoru příznaků cílového uzlu. Mezi přeskakující spojení můžeme zařadit residuální spojení, inspirované jeho využitím pro hluboké konvoluční neuronové sítě při klasifikaci obrazu [11], nebo jednoduchou konkatenaci vektorů příznaků, jak navrhli Hamilton a spol. [9] v rámci GraphSAGE architektury (kapitola 3.4).

3.3 Grafové konvoluční sítě

Grafové konvoluční sítě (angl. graph convolution network) představili Kipf a spol. [13] s motivací prvořádové aproximace lokalizovaných grafových spektrálních filtrů, přičemž tuto síť popsali jako síť skládající se z grafových konvolučních vrstev jdoucích za sebou a daných následujícím propagačním pravidlem:

$$H^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}\right), \tag{3.7}$$

kde $\tilde{A} = a\ + I_N$ je matice sousednosti neorientovaného grafu G s přidanými smyčkami. I_N je matice identity, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ a $W^l \in \mathbb{R}^{F_{out} \times F_{in}}$ je učitelná váhová matice vrstvy l. $\sigma(\cdot)$ značí aktivační funkci (typicky ReLU) a $H^l \in \mathbb{R}^{N \times F_{in}}$ je matice aktivací dané vrstvy, přičemž pro první grafovou konvoluční vrstvu tato matice odpovídá vstupním příznakům uzlů, tedy $H^0 = X$, N je počet uzlů a F_{in} (respektive F_{out}) počet vstupních (respektive výstupních) příznaků každého uzlu.

Kipf a spol. [13] předpokládají využití takovéto architektury pro semi-supervised učení v transduktivním režimu, tedy v režimu, kdy je dostupný již celý graf společně s jeho strukturou a jsou dostupná označení pouze části uzlů, přičemž úkolem je označit všechny

zbývající neoznačené uzly. Tento režim má však nevýhodu, že nedokáže pracovat s dříve neviděnými uzly nebo hranami, a není tak použitelný na problémy, kdy se graf rychle vyvíjí v čase a není efektivní pouštět trénování po každé změně takového grafu znovu, jak zdůraznili Hamilton a spol. [9]. Propagační pravidlo pro celý graf dané rovnicí 3.7 však lze jednoduše převést na následující message passing funkci:

$$\mathbf{h}_u^{l+1} = \mathbf{m}_{N(u)}^l = \sigma\left(\mathbf{W}^l \sum_{v \in N(u) \cup \{u\}} \frac{\mathbf{h}_v^l}{\sqrt{|N(u)||N(v)|}}\right), \tag{3.8}$$

kde $\mathbf{m}_{N(u)}^l$ značí agregovanou zprávu, $\mathbf{W}^l \in \mathbb{R}^{F_{out} \times F_{in}}$ značí učitelnou matici vah vrstvy grafové konvoluční vrstvy l, $\mathbf{h}_u^l \in \mathbb{R}^{F_{in}}$ značí vektor příznaků uzlu u na vstupu a $\mathbf{h}_u^{l+1} \in \mathbb{R}^{F_{out}}$ značí vektor příznaků uzlu u na výstupu grafové konvoluční vrstvy l a $\sigma(\cdot)$ značí aktivační funkci. Tuto message passing funkci lze využít v induktivním režimu společně s mini-batch trénováním, které představili Hamilton a spol. [9]. Ve skutečnosti je dnes kvůli větší efektivitě nejvíce využívána varianta mini-batch trénování s maticovým násobením, tedy s původním přechodovým pravidlem, které definovali Kipf a spol. [13], avšak aplikovaným pouze na malý subgraf daný mini-batch strategií.

3.4 Architektura GraphSAGE

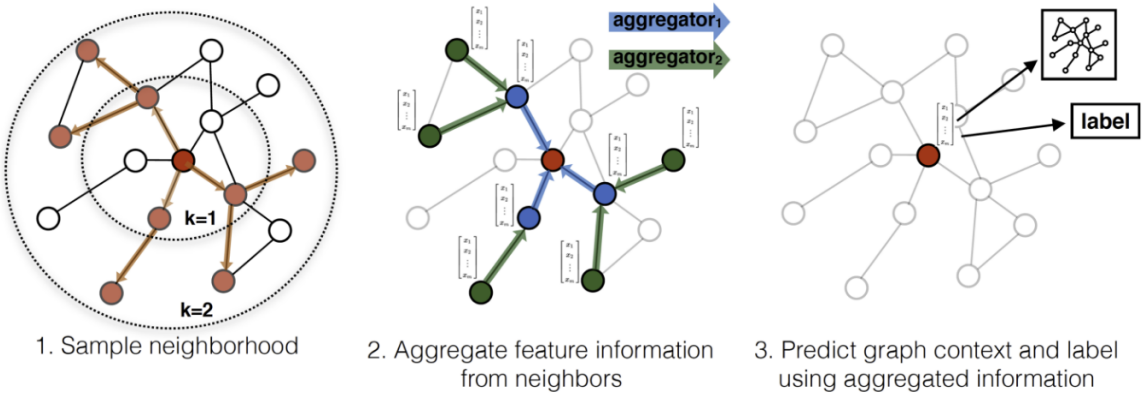
Hamilton a spol. [9] představili framework *GraphSAGE*, který se zaměřuje na *induktivní učení reprezentací uzlů* (tj. schopnost generalizovat na neviděná data) pomocí *grafových neuronových sítí* a *škálovatelnost* grafových neuronových sítí. Tato metoda pro každý cílový uzel *vzorkuje pevný počet sousedů* na každé vzdálenostní úrovni $k = 1, 2, ..K$, až po specifikovanou maximální vzdálenost K. Následně agreguje vektory příznaků těchto navzorkovaných sousedů pomocí *trénovatelné agregační funkce* (viz message passing framework), čímž vzniká finální vektor vnoření uzlu (viz obrázek 3.2). Autoři v rámci jejich práce navrhují 4 různé typy trénovatelné agregační funkce — agregační funkci odpovídající *grafové konvoluční vrstvě*, což odpovídá rovnici 3.8, *průměrovací agregační funkci*, *pooling agregační funkci*, která je založená na vícevrstvém perceptronu následovaným *max* (nebo také *mean*) funkcí. Poslední agregační funkcí je agregační funkce založená na *rekurentní neuronové síti LSTM*, přičemž tato síť není přirozeně invariantní vůči permutaci, což GraphSAGE řeší jednoduchým použitím náhodné permutace sousedů daného uzlu. Z experimentů, které Hamilton a spol. provedli, vyplývá, že nejlepších výsledků dosahují LSTM a pooling agregátory.

3.5 Architektura Graph Attention Network

Architektura Graph Attention Network vychází z architektury grafových konvolučních sítí, přičemž na rozdíl od grafových konvolučních sítí, které považují každý sousední uzel za stejně důležitý, dávají jednotlivým sousedním uzlům různé důležitosti na základě attention mechanismu.[2] Hlavním stavebním kamenem této architektury je grafová attention vrstva, přičemž síť architektury graph attention network se bude skládat z několika takovýchto vrstev.

Nechť vstupem grafové attention vrstvy je množina příznaků uzlů $H^l = \{h_1^l, h_2^l, ..., h_N^l\}$, kde $h_i^l \in \mathbb{R}^{F_{in}}$ značí vektor příznaků uzlu i na vstupu vrstvy l, N značí počet uzlů grafu,

[2] Česky mechanismus pozornosti, avšak attention je častěji používaný pojem a překlad tohoto pojmu by mohl způsobit zmatení znalého čtenáře.



Obrázek 3.2: Ilustrace vzorkování a agregace použité v rámci metody GraphSAGE. Převzato z literatury [9].

F_{in} je počet příznaků každého uzlu na vstupu vrstvy l. Dále nechť výstupem této vrstvy je množina příznaků uzlů $H^{l+1} = \{h_1^{l+1}, h_2^{l+1}, h_3^{l+1}, ..., h_N^{l+1}\}$, kde $h_i^{l+1} \in \mathbb{R}^{F_{out}}$ značí vektor příznaků uzlu na výstupu vrstvy l, F_{out} je počet příznaků každého uzlu na výstupu vrstvy l. A nechť $N(i)$ značí množinu uzlů sousedících s uzlem i, tedy $j \in N(i) \Leftrightarrow (i,j) \in E$.

Aby tato vrstva měla dostatečné vyjadřovací schopnosti k transformaci vstupních příznaků uzlů na příznaky vyšší úrovně, je její součástí učitelná lineární transformace, parametrizovaná váhovou maticí $\mathbf{W} \in \mathbb{R}^{F_{out} \times F_{in}}$. Tato transformace je prvním krokem ve výpočtu grafové attention vrstvy, kdy je vstupní vektor příznaků uzlu h_i^l zleva vynásoben touto váhovou maticí. Následně je na takto získaných příznacích proveden výpočet attention koeficientů e_{ij} pomocí sdíleného self-attention mechanismu (česky mechanismu sebe pozornosti) $a : \mathbb{R}^{F_{out}} \times \mathbb{R}^{F_{out}} \to \mathbb{R}$:

$$e_{ij} = a(\mathbf{W}h_i^l, \mathbf{W}h_j^l), \tag{3.9}$$

kde e_{ij} odpovídá attention koeficientu mezi uzly i a j. Tento koeficient indikuje důležitost příznaků uzlu j pro uzel i. Stojí za to zdůraznit, že obecně nic nebrání výpočtu attention koeficientu mezi všemi uzly, avšak tím by nebyla využita strukturální informace grafu. V rámci grafové attention vrstvy jsou proto brány v potaz pouze uzly sousedící s uzlem i, tedy uzly $j \in N(i)$, a je proveden maskovaný attention mechanismus, čímž je do tohoto mechanismu vložena i samotná struktura grafu. Tyto koeficienty se následně normalizují za pomoci softmax funkce:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})}, \tag{3.10}$$

aby byly jednoduše porovnatelné pro všechny uzly v rámci celého grafu [24].

Veličković a spol. [24] využili pro attention mechanismus jednovrstvou dopřednou neuronovou síť, parametrizovanou váhovým vektorem $\vec{a} \in \mathbb{R}^{2F_{out}}$ s LeakyReLU aktivační funkcí, jak lze vidět na obrázku 3.3a, a výsledný vzorec pro výpočet normalizovaných attention koeficientů pak odpovídá rovnici 3.11. Avšak samotná grafová attention vrstva tento způsob nevyžaduje a je kompatibilní s jakýmkoliv attention mechanismem.

$$\alpha_{ij} = \frac{exp(LeakyReLU(\vec{a}^T[\mathbf{W}h_i^l || \mathbf{W}h_j^l]))}{\sum_{k \in N(i)} exp(LeakyReLU(\vec{a}^T[\mathbf{W}h_i^l || \mathbf{W}h_k^l]))}, \tag{3.11}$$

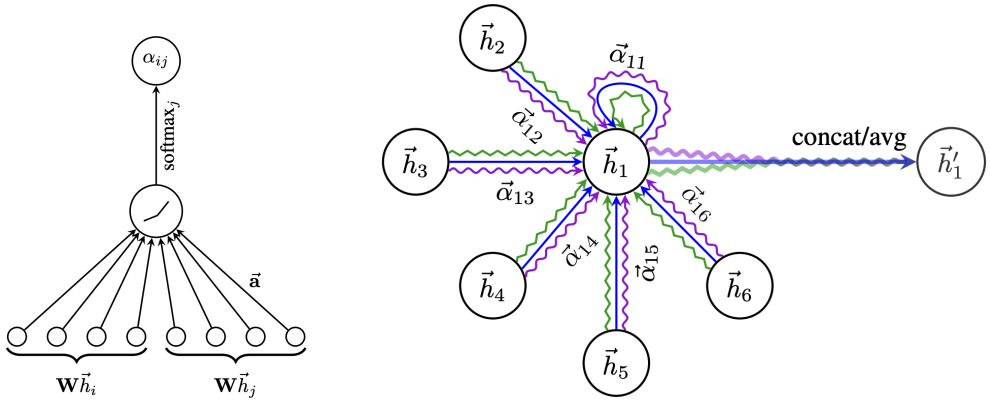
Po vypočtení normalizovaných attention koeficientů je výsledný vektor vkládání uzlu vypočten jako lineární kombinace těchto koeficientů se vstupními vektory příznaků odpovídajících uzlů a případném aplikování nelinearity σ:

$$h_i^{l+1} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} \mathbf{W} h_j^l\right). \tag{3.12}$$

Tento mechanismus lze tradičně rozšířit na vícehlavový attention výpočet, kdy je provedeno K nezávislých attention výpočtů a výsledné vektory příznaků těchto hlav jsou následně spojeny za sebe do výsledného výstupního vektoru příznaků, čemuž odpovídá rovnice 3.13, nebo jsou tyto vektory zprůměrovány a až poté je aplikována případná nelinearita, čemuž odpovídá rovnice 3.14. Ilustraci tohoto procesu lze vidět na obrázku 3.3b [24].

$$h_i^{l+1} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in N(i)} \alpha_{ij}^k \mathbf{W}^k h_j^l\right). \tag{3.13}$$

$$h_i^{l+1} = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N(i)} \alpha_{ij}^k \mathbf{W}^k h_j^l\right). \tag{3.14}$$



(a) (b)

Obrázek 3.3: Ilustrace ukazující (a) výpočet pozornosti $a(\mathbf{W}h_i, \mathbf{W}h_j)$ mezi dvěma uzly i, j, parametrizován učitelným vektorem $\vec{a}$. (b) výpočet výsledného vektoru vkládání uzlu 1. Každá barva značí nezávislý výpočet pozornosti, přičemž jednotlivé výsledky jsou následně buďto spojeny za sebe nebo zprůměrovány do výsledného vektoru vkládání uzlu 1. Převzato z literatury [24].

Kapitola 4

Otevřená datová sada

Tato kapitola se zabývá tvorbou nové otevřené datové sady, která se může stát novým stavebním blokem automatických systémů k detekci nelegálních transakcí založených na strojovém učení s učitelem. Tato datová sada je postavena na základech datové sady `Elliptic`, přičemž opravuje zásadní nedostatky této datové sady, které tato práce identifikovala.

Díky zachování původních označení transakcí a identického rozdělení klasifikovaných transakcí do *trénovací*, *validační* a *testovací datové sady* je zajištěna možnost přímého porovnání výkonu jednotlivých modelů strojového učení mezi původní a nově vytvořenou datovou sadou, nevezmeme-li v úvahu chyby v izolaci testovací datové sady (viz kapitola 4.2), které naměřené výsledky na datové sadě `Elliptic` minimálně zpochybňují.

Následující text se v kapitole 4.1 věnuje reverznímu inženýrství datové sady `Elliptic`, poté jsou v rámci kapitoly 4.2 shrnuty nedostatky této datové sady, které se snaží navržená datová sada `OpenElliptic` adresovat. Identifikací konkrétních transakcí obsažených v datové sadě `Elliptic` se zabývá kapitola 4.3. Kapitola 4.4 se pak zabývá samotnou tvorbou datové sady `OpenElliptic`, konkrétně popisuje moduly, které bylo třeba navrhnout, aby mohla být tato datová sada sestavena. Popisu této datové sady a všech příznaků jednotlivých transakcí se věnuje kapitola 4.5.

4.1 Reverzní inženýrství datové sady `Elliptic`

Prvním klíčovým krokem při tvorbě otevřené alternativy k datové sadě `Elliptic` bylo porozumění datům původní datové sady a alespoň částečné odhadnutí jejího obsahu. K tomu jsem využil veřejně dostupné datové sady identifikovaných transakcí,[1] která obsahuje překlady identifikátorů transakcí z datové sady `Elliptic` na haše odpovídajících transakcí. Ke každé z těchto transakcí jsem následně získal příslušný obsah za pomoci volání aplikačního rozhraní aplikace Blockbook.[2]

Jedním z prvních zjištění explorační analýzy bylo, že všechny atributy mají průměr blízký nule a směrodatnou odchylku jedna. Jedinou výjimku tvoří první atribut (FEAT-1[3]), který jako jediný není anonymizován a jedná se o časový krok, ve kterém se transakce nachází. Z toho můžeme usoudit, že jednotlivé příznaky transakcí byly pravděpodobně anonymizovány za pomoci *z-score normalizace* s cílovým průměrem 0 a směrodatnou od-

[1] https://www.kaggle.com/datasets/alexbenzik/deanonymized-995-pct-of-elliptic-transactions/discussion/193249

[2] https://github.com/trezor/blockbook/tree/master

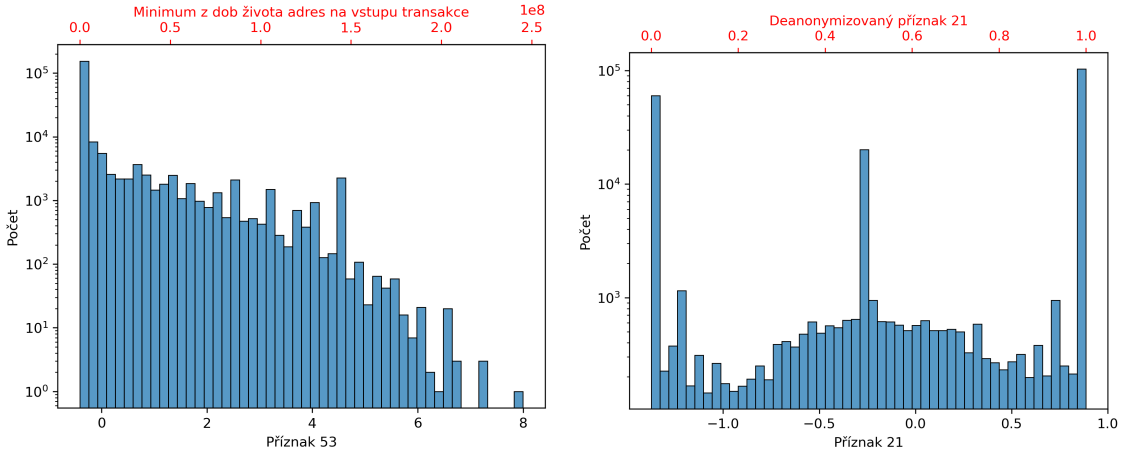
[3] Jednotlivé atributy budeme číslovat od jedné, což odpovídá jejich indexu v tabulce této datové sady.

chylkou 1, tedy dle následující rovnice:

$$y = \frac{(x - \mu)}{\sigma}, \tag{4.1}$$

kde x značí původní hodnotu příznaku, y značí normalizovanou hodnotu příznaku, μ značí průměr a σ značí směrodatnou odchylku.

V rámci explorační analýzy jsem vypozoroval, že většina distribučních grafů má jednu ze dvou podob — buďto se jedná o exponenciální rozdělení, které je zdola ohraničeno, jak je vyobrazeno na obrázku 4.1a, anebo se jedná většinou o oboustranně ohraničené rozdělení, kdy většina hodnot nabývá minima, maxima anebo nabývá hodnoty ležící přesně v polovině rozsahu hodnot, což je zřejmé z obrázku 4.1b.



(a) (b)

Obrázek 4.1: Distribuční graf (a) příznaku 53, který byl denormalizován a u kterého byl identifikován jeho význam — minimum ze životnosti jednotlivých adres obsažených v dané transakci na vstupu, (b) příznaku 21, který byl denormalizován, ale jeho význam nebyl plně odhalen (bylo zjištěno, že se jedná o příznak související s rozdělením výstupů dané transakce, avšak nebyla identifikována konkrétní formule výpočtu).

Příznak s exponenciálním rozdělením hodnot

V prvním případě, kdy distribuce hodnot příznaku připomíná exponenciální rozdělení a jsou zdola ohraničené, jsem nejprve udělal naivní předpoklad, že se jedná o přirozená čísla, jejichž dolní hranicí je nula. Postup deanonymizace významu příznaku se skládal ze dvou kroků — denormalizace a odhalení významu příznaku.

Denormalizace příznaku

- *Přirozená čísla.* Jak již bylo zmíněno dříve, samotné hodnoty příznaků jsou pravděpodobně anonymizovány za pomoci z-score normalizace. Pak nechť máme dvě různé anonymizované hodnoty y_1, y_2, které odpovídají původním příznakům x_1, x_2, dle následující rovnice:

$$y_1 = \frac{x_1 - \mu}{\sigma}, \text{respektive} \tag{4.2}$$

$$y_2 = \frac{x_2 - \mu}{\sigma}, \tag{4.3}$$

kde význam μ a σ odpovídá významu v rovnici 4.1. Rozdíl původních hodnot d_x pak definujeme následovně:

$$d_x = x_2 - x_1. \tag{4.4}$$

Dále nechť máme rozdíl anonymizovaných hodnot d_y, definovaný jako:

$$d_y = y_2 - y_1, \tag{4.5}$$

kde y_1, y_2 jsou anonymizované hodnoty. Dosadíme-li rovnici 4.2 do rovnice 4.5, získáme:

$$d_y = \frac{x_2 - \mu}{\sigma} - \frac{x_1 - \mu}{\sigma}, \tag{4.6}$$

což lze dále převést na jeden jediný zlomek:

$$d_y = \frac{(x_2 - \mu) - (x_1 - \mu)}{\sigma}, \tag{4.7}$$

$$d_y = \frac{x_2 - \mu - x_1 + \mu}{\sigma}, \tag{4.8}$$

$$d_y = \frac{x_2 - x_1}{\sigma}. \tag{4.9}$$

Můžeme si všimnout, že čitatel zlomku na pravé straně rovnice 4.9 odpovídá pravé straně rovnice 4.4, rovnici lze tedy převést do tvaru:

$$d_y = \frac{d_x}{\sigma}, \tag{4.10}$$

ze kterého vyplývá, že změna tohoto rozdílu d_x při z-score normalizaci závisí pouze na škálovacím parametru σ. Pak pro nejmenší rozdíl mezi dvěma různými hodnotami normalizovaných čísel odpovídá přeškálovanému nejmenšímu rozdílu mezi dvěma různými hodnotami v původním rozsahu, tedy platí:

$$d_{ymin} = \frac{d_{xmin}}{\sigma}. \tag{4.11}$$

Dále můžeme využít předpokladu, že původní hodnoty jsou přirozená čísla, a tedy nejmenší možný rozdíl mezi dvěma odlišnými hodnotami se rovná 1. Z toho plyne, že odpovídající nejmenší rozdíl dvou odlišných normalizovaných hodnot d_{ymin} odpovídá původní hodnotě 1, tedy $d_{xmin} = 1$. Po dosazení do rovnice 4.10, dostaneme rovnici pro výpočet směrodatné odchylky σ původního nenormalizovaného příznaku:

$$d_{ymin} = \frac{d_{xmin}}{\sigma}, \tag{4.12}$$

$$d_{ymin} = \frac{1}{\sigma}, \tag{4.13}$$

$$\sigma = \frac{1}{d_{ymin}}. \tag{4.14}$$

Dále budeme naivně předpokládat, že nejmenší možná hodnota příznaku — nula, se v původních datech objevila, a odpovídá tedy nejmenší hodnotě y_{min} v anonymizovaných hodnotách tohoto příznaku, tedy platí:

$$y_{min} = \frac{x_{min} - \mu}{\sigma}, \tag{4.15}$$

$$y_{min} = \frac{0 - \mu}{\sigma}, \tag{4.16}$$

$$y_{min} \cdot \sigma = -\mu, \tag{4.17}$$

$$\mu = -y_{min} \cdot \sigma. \tag{4.18}$$

Při znalosti obou parametrů provedené z-score normalizace (σ a μ), můžeme všechny normalizované hodnoty převést na původní nenormalizované hodnoty následovně:

$$x = y \cdot \sigma + \mu. \tag{4.19}$$

Poté, co jsem takto příznak převedl do jeho původního měřítka, provedl jsem ověření, zda můj počáteční předpoklad o přirozenosti původních čísel byl správný, a to empiricky vypsáním unikátních původních hodnot, kdy v případě, že byl počáteční předpoklad správný, jsou těmito unikátními hodnotami přirozená čísla (s malou odchylkou způsobenou nepřesností čísel s plovoucí desetinnou čárkou). Příklad denormalizovaných hodnot příznaku 7[4] lze vidět na obrázku 4.2.

```
[1.0000000000000002, 2.000000000000081, 3.0000000000001603, 4.000000000000241,
5.0000000000003215, 6.0000000000004015, 7.000000000000483, 8.000000000000565,
9.000000000000643, 10.000000000000725]
```

Obrázek 4.2: Příklad prvních deseti unikátních hodnot příznaku 7 (počet unikátních adres na vstupu) po denormalizaci.

Nakonec je třeba zmínit, že některé z deanonymizovaných příznaků neměly původní nejmenší hodnotu nula a na základě toho bylo potřeba upravit rovnici 4.18, pro získání správné původní hodnoty. Jednalo se například o příznak značící *počet vstupů dané transakce*, kdy nejmenší hodnotou bylo v rámci původní datové sady jedna, jelikož se zde nenachází žádné coinbase transakce, anebo se jednalo o případ *doby života adres na vstupu transakce*, kdy existují adresy se zápornou dobou života, způsobenou ne vždy chronologickým pořadím bloků v rámci blockchainu sítě Bitcoin.[5]

- *Příznak agregující přirozená čísla.* V případě, kdy hodnoty získané denormalizací neodpovídaly přirozeným číslům a původní předpoklad, že se jedná o přirozená čísla, byl tak neplatný, jsem využil dalšího naivního předpokladu, a to, že se jedná o agregaci přirozených čísel, konkrétně o průměr. Z popisu datové sady `Elliptic` vyplývá, že globální příznaky jsou agregací příznaků z okolních transakcí, tedy pokud jsem předpokládal, že v rámci příznaků existují příznaky, jejichž hodnoty odpovídají přirozeným číslům, je pravděpodobné, že v rámci agregovaných příznaků budou i příznaky, které agregují přirozená čísla. V takovém případě nejmenší rozdíl dvou různých hodnot

[4] Počet unikátních adres na vstupu.

[5] Jako příklad zde lze uvést bloky 429 508 a 429 509, kdy časová značka bloku 429 509 je o 124 sekund menší, než předcházejícího bloku 429 508.

již neodpovídá jedné, avšak zde jsem naivně předpokládal, že jedné bude odpovídat nejčastější rozdíl d_{ymodus} mezi dvěma po sobě následujícími unikátními hodnotami. Dosadíme-li do rovnice 4.10, dostaneme rovnici:

$$d_{ymodus} = \frac{d_{xmodus}}{\sigma}, \tag{4.20}$$

do které můžeme na základě předpokladu za d_{xmodus} dosadit 1, čímž získáme rovnici:

$$d_{ymodus} = \frac{1}{\sigma}. \tag{4.21}$$

Rovnici 4.21 můžeme upravit do následujícího tvaru pro výpočet původní směrodatné odchylky σ na základě nejčastějšího rozdílu mezi dvěma po sobě jdoucími normalizovanými hodnotami d_{ymodus}:

$$\sigma = \frac{1}{d_{ymodus}}. \tag{4.22}$$

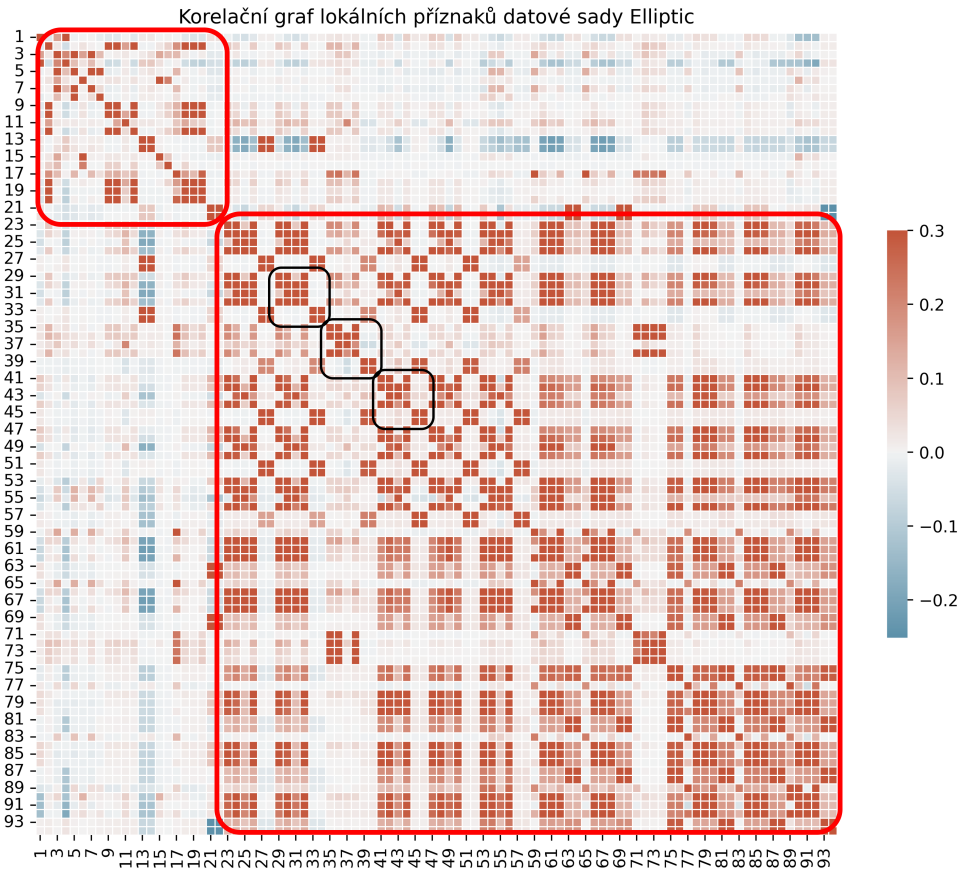
Tímto způsobem se mi podařilo denormalizovat hodnoty, které vznikly agregováním příznaků nabývajících hodnot přirozených čísel, jako je například *průměr výšek bloků*, ve kterých se nacházejí vstupní transakce.

- *Ostatní příznaky.* V případě příznaků, které se nepodařilo denormalizovat ani jedním z předchozích způsobů, jsem využil znalosti z již deanonymizovaných příznaků společně se vzory v korelačním grafu k jejich deanonymizaci, tedy krok denormalizace a krok odhalení významu zde byly spojeny do jednoho společného kroku.

  V již deanonymizovaných příznacích jsem nalezl vzor, kdy u příznaků agregujících příznaky zapojených adres se vždy nacházel vedle sebe příznak minima a příznak maxima, dále ob jedno následovaný příznakem průměru. Tento vzor lze vidět i v samotném korelačním grafu lokálních příznaků transakcí. V korelačním grafu (viz obrázek 4.3) můžeme identifikovat příznaky, které jsou počítány ze samotné transakce — prvních 22 příznaků, a příznaky, které jsou počítány jako agregace příznaků zapojených adres — příznaky 23 až 94. V rámci příznaků adres můžeme pozorovat vzor, kdy se vždy nacházejí čtyři navzájem korelované příznaky, následované dalšími dvěma navzájem korelovanými příznaky. Díky znalosti již deanonymizovaných příznaků se mi podařilo odhalit, že se jedná vždy o šest po sobě jdoucích příznaků, které spolu souvisí — agregují stejné atributy. Konkrétně se vždy jedná o zmíněné minimum, maximum, hodnota vyjadřující rozsah těchto hodnot, průměr a dva koeficienty vztahující se k samotné distribuci hodnot. Bohužel se mi v rámci této práce nepodařilo odhalit konkrétní způsob výpočtu těchto koeficientů a způsob výpočtu hodnoty vyjadřující rozsah.

Odhalení významu příznaku

Po úspěšné denormalizaci hodnot daného příznaku jsem se zaměřil na identifikaci významu tohoto příznaku, a to vždy následujícím způsobem. Unikátní denormalizované hodnoty jsem seřadil a vybral skupinu transakcí, jež mají nejmenší možnou hodnotu tohoto příznaku, stejně tak skupinu transakcí s největší možnou hodnotou, a dále jsem se pokusil najít takovou skupinu transakcí, jejichž hodnota přibližně odpovídá středu možného rozsahu, a která obsahuje alespoň dvě transakce. Následně jsem empiricky zkoumal, v jakých atributech se transakce v rámci těchto skupin shodují a v jakých atributech se liší mezi skupinami. Při



Obrázek 4.3: Korelační graf lokálních příznaků transakcí v datové sadě `Elliptic`. Na grafu lze pozorovat opakující se vzor, kdy je vždy čtveřice po sobě jdoucích korelovaných příznaků následovaná dvojicí spolu souvisejících příznaků. V rámci této práce se povedlo odhalit, že se jedná vždy o šestici příznaků, které jsou agregací stejných vstupních dat, jako je například počet prostředků na jednotlivých vstupních adresách.

tomto porovnávání mi jako vodítko sloužil rozsah denormalizovaných hodnot. V případě, kdy se největší možné hodnoty pohybovaly v miliardách, bylo pravděpodobné, že se bude jednat o příznak nějak spojený s časovými razítky transakcí či jejich objemem. V případě, kdy se většina hodnot pohybovala ve stovkách až tisících, hledal jsem v příznacích, jako je počet vstupů, počet výstupů nebo počty transakcí spojených s obsaženými adresami.

Z obou stran ohraničená distribuce hodnot

V případě, kdy se jedná o zdola i shora ohraničené rozdělení, přičemž většina hodnot se nabývá minima či maxima, se může pravděpodobně jednat o původní hodnoty v tradičním rozsahu $\langle 0, 1\rangle$, či $\langle -1, 1\rangle$.

Při denormalizaci takovýchto příznaků jsem využil naivního předpokladu, že se jedná o rozsah $\langle 0, 1\rangle$, tedy samotné rozpětí původních hodnot odpovídá jedné. Pro výpočet rozsahu normalizovaných hodnot d_{ymax} platí následující rovnice:

$$d_{ymax} = y_{max} - y_{min}, \tag{4.23}$$

kde y_{max} a y_{min} odpovídá maximální, respektive minimální, hodnotě v rámci normalizovaných hodnot. Stejným způsobem lze definovat i rozsah původních hodnot d_{xmax}:

$$d_{xmax} = x_{max} - x_{min}, \tag{4.24}$$

kde x_{max} a x_{min} odpovídá maximální, respektive minimální, hodnotě v rámci původních hodnot.

Využijeme-li dále předpokladu, že jsou čísla z-score normalizována a dosadíme do rovnice 4.23, dostaneme rovnici:

$$d_{ymax} = \frac{x_{max} - \mu}{\sigma} - \frac{x_{min} - \mu}{\sigma}. \tag{4.25}$$

Poté zlomky na pravé straně rovnice převedeme na společný jmenovatel a dostaneme rovnici:

$$d_{ymax} = \frac{(x_{max} - \mu) - (x_{min} - \mu)}{\sigma}, \text{respektive} \tag{4.26}$$

$$d_{ymax} = \frac{x_{max} - x_{min}}{\sigma}, \tag{4.27}$$

přičemž $x_{max} - x_{min}$ dle rovnice 4.24 odpovídá d_xmax, rovnici 4.27 lze tedy převést do tvaru

$$d_{ymax} = \frac{d_{xmax}}{\sigma}. \tag{4.28}$$

Převedeme-li σ na levou stranu a d_{ymax} na pravou stranu, získáme rovnici pro výpočet původní směrodatné odchylky:

$$\sigma = \frac{d_{xmax}}{d_{ymax}}. \tag{4.29}$$

Poté dle naivního předpokladu o původním rozsahu hodnot za d_{xmax} dosadíme 1, čímž získáme rovnici:

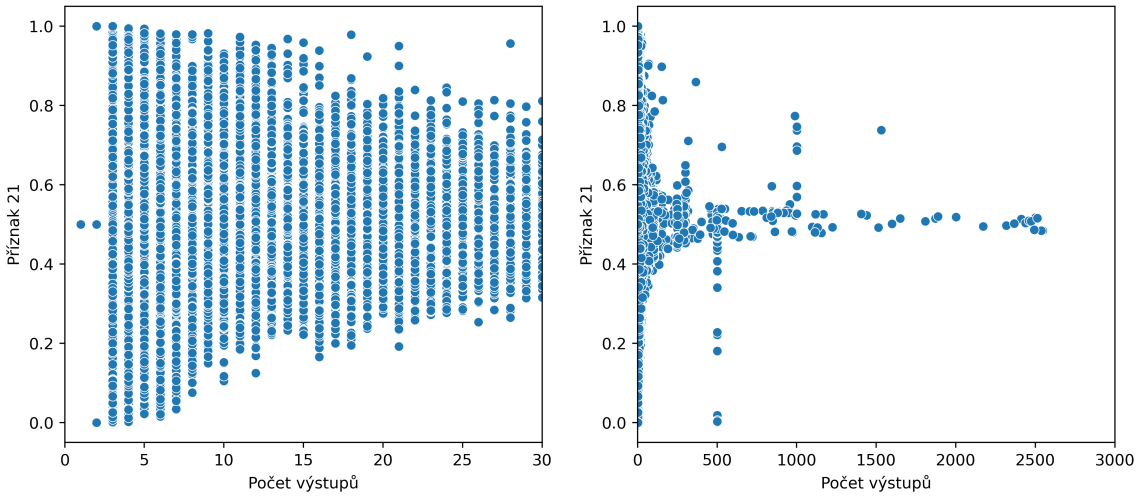
$$\sigma = \frac{1}{d_{ymax}}, \tag{4.30}$$

která bude sloužit pro výpočet původní směrodatné odchylky pouze z hodnoty d_{ymax}, kterou lze vypočítat dosazením maximální a minimální hodnoty v rámci normalizovaných čísel do rovnice 4.23.

Pro střední hodnotu μ pak platí rovnice 4.18, jelikož stejně jako v minulém případě je nejnižší možnou hodnotou původního rozpětí nula. V rámci této práce se bohužel nepodařilo plně deanonymizovat žádný z takovýchto příznaků. U příznaku 21 bylo identifikováno, že tento příznak je závislý na distribuci hodnot na výstupu dané transakce, avšak nepovedlo se mi identifikovat konkrétní způsob výpočtu. Na obrázku 4.4 lze pozorovat, že v případě pouze jedné výstupní hodnoty odpovídá hodnota tohoto příznaku 0,5. V případě, kdy má transakce dva výstupy, pak je tento příznak 0 ve chvíli, kdy je první výstup větší než ten druhý, 1 v opačném případě a 0,5 v případě, kdy jsou hodnoty na obou výstupech shodné. Pro tři a více výstupů hodnota tohoto příznaku již nikdy nenabývá hodnotu 0 nebo 1, a nabývá hodnotu 0,5 v případě, kdy jsou hodnoty všech výstupů shodné.

Výsledky reverzního inženýrství datové sady `Elliptic`

Prostřednictvím reverzního inženýrství datové sady `Elliptic` se podařilo identifikovat původní významy 48 lokálních a 30 globálních příznaků. To tvoří 51 % původních lokálních příznaků, respektive 41 % globálních příznaků (konkrétní význam všech deanonymizovaných příznaků se nachází v příloze A). Dále jsem zjistil, že lokální příznaky jednotlivých



Obrázek 4.4: Graf vztahu mezi příznakem 21 a počtem výstupů dané transakce. Na grafu lze pozorovat, že pro 1 výstup nabývá příznak 21 vždy hodnoty 0,5, a pro 2 výstupy nabývá hodnot 0, 0,5 a 1. Pro 3 a více vstupů nabývá příznak 21 hodnot v rozsahu $(0, 1)$.

transakcí nejsou tvořeny pouze informacemi obsaženými v rámci dané transakce, ale tvoří je i informace o adresách zapojených do těchto transakcí, jako je doba jejich života, počet příchozích a odchozích transakcí, volné prostředky v době tvorby datové sady. Identifikoval jsem konkrétní blok (blok 575 059), ke kterému jsou vztaženy jak globální příznaky (existující transakce na výstupech), tak tyto globální příznaky adres. V neposlední řadě jsem zjistil, že každý z příznaků byl pravděpodobně normalizován za pomoci z-score normalizace, přičemž k výpočtu parametrů σ a μ bylo využito celé datové sady.

4.2 Nedostatky datové sady `Elliptic`

V rámci této práce byly identifikovány nejzávažnější nedostatky datové sady `Elliptic`. Jedná se o problém špatné izolace testovací datové sady, problém s chybějícím okolím klasifikovaných transakcí a o samotný problém s anonymností datové sady, přičemž většina těchto nedostatků přechází i na datové sady z této sady odvozených, tj. například `Elliptic++`.

Špatně izolovaná testovací datová sada

Nejzávažnějším problémem datové sady `Elliptic` je nedostatečná izolace testovací datové sady. Jak bylo zmíněno v kapitole 4.1, jednotlivé příznaky transakcí obsahují také agregované příznaky jednotlivých adres, přičemž tyto příznaky nejsou vztaženy k časovému období trénovací datové sady, avšak jedná se o stav adres k době sestavení samotné datové sady, tedy chvíle, kdy příznaky adres obsahují informace i o transakcích obsažených v testovací sadě, čímž efektivně dochází k pronikání informací z testovací datové sady do trénovací.

Závažnost tohoto problému je dále ještě prohloubena faktem, že více jak 25 % transakcí v testovací datové sadě má alespoň jednu adresu společnou s nějakou transakcí v sadě trénovací, přičemž příznaky této adresy jsou z důvodu volby jednoho momentu pro jejich výpočet stejné. Dále Song a spol. [23] ukazují, že v případě, kdy místo transakcí jsou klasifikovány adresy a následně jsou z predikovaných tříd odvozeny třídy samotných transakcí dle pravidla, kdy každá transakce zahrnující nezákonnou adresu je klasifikována jako *nezákonná*,

dosahuje tato klasifikace state-of-the-art výsledků. Elmougy a Liu [6] v rámci jejich práce identifikovali příznaky, které nejvíce ovlivňují klasifikaci transakcí natrénovaného Random Forest klasifikátoru, přičemž se převážně jedná o příznaky adres než příznaky samotných transakcí.[6] Z toho lze usoudit, že klasifikátory natrénované na datové sadě `Elliptic` s největší pravděpodobností hojně využívají příznaky samotných adres, přitom pro zmíněných 25 % transakcí odpovídají tyto příznaky příznakům obsaženým v trénovací datové sadě, díky čemuž může v rámci testování modelů docházet ke klasifikaci stejných adres, na kterých byl model trénován.

Chybějící okolí klasifikovaných transakcí

Dalším problémem této datové sady je chybějící okolí klasifikovaných transakcí, což je fakt, který pramení ze samotného návrhu této datové sady, kdy je každý časový krok plně odděleným podgrafem a obsahuje pouze transakce, které se objevily během tří hodin sběru tohoto časového kroku. Tento způsob sběru dat byl zvolen za účelem poskytnutí v čase se vyvíjejícího grafu, aby byla umožněna jeho časová analýza. Tato datová sada se tak dá velice dobře využít pro evaluaci klasifikátorů a grafových neuronových sítí pracujících s touto časovou informací, například pro grafovou neuronovou síť EvolveGCN [21].

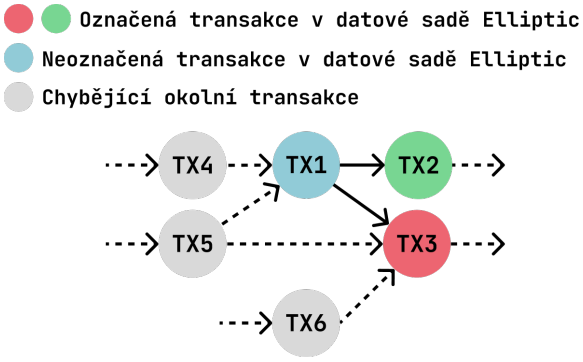
Na druhou stranu, v případě, kdy tyto časové informace trénovaný klasifikátor nijak nevyužívá, což odpovídá většině prací využívajících tuto datovou sadu, a je využita grafová neuronová síť za účelem využití kontextu klasifikovaných transakcí, způsobuje tato metoda sběru dat chybějící okolní transakce, které by grafová neuronová síť mohla využít. Dostupné vstupní transakce jednotlivých transakcí tedy nejsou všechny vstupní transakce, avšak pouze ty, které se nacházely ve zvoleném časovém úseku pro daný časový krok. Výsledkem jsou situace, kdy klasifikovaná transakce nemá žádný vstup, přestože se nejedná o coinbase transakci. Příklad takovéto situace lze pozorovat na obrázku 4.5. Transakce TX3 je označenou, a tedy i klasifikovanou, transakcí v datové sadě `Elliptic`. V případě použití jednovrstvé grafové neuronové sítě, která pracuje s 1-okolím dané transakce, budou v agregovaných příznacích chybět informace z transakce TX5 a TX6, které jsou vstupem klasifikované transakce TX3, neboť se nenachází ve zvoleném časovém okně pro tento časový krok.

Anonymizované příznaky

V neposlední řadě je jedním z problémů datové sady `Elliptic` samotná anonymita příznaků v této datové sadě a jejich nedostatečný popis z důvodů ochrany duševního vlastnictví společnosti `Elliptic Enterprises Limited`. Jak se podařilo zjistit, lokální příznaky transakcí se skládají i z agregovaných příznaků jednotlivých zapojených adres. Nezveřejnění této skutečnosti však vedlo k špatné interpretaci lokálních příznaků transakcí, jak lze vidět například v práci Elmougy a Liu [6], kteří tvrdí, že se při klasifikaci transakcí z datové sady `Elliptic` neberou v potaz informace o adresách, což, jak bylo dokázáno v této práci, není pravdivé tvrzení.

Významnějším problémem, který anonymnost příznaků způsobuje, je fakt, že bez znalosti způsobu výpočtu každého z příznaků není možné vytvořit takovýto vektor příznaků z libovolné bitcoinové transakce. To nejen znemožňuje rozšíření této datové sady o nové

[6] Elmougy a Liu hovoří v rámci jejich práce pouze o konkrétních příznacích v anonymizované formě a až deanonymizace těchto příznaků dovoluje zmíněnou interpretaci.



Obrázek 4.5: Vizualizace znázorňuje problém chybějícího okolí klasifikovaných transakcí v datové sadě `Elliptic`. Na obrázku lze pozorovat, že datová sada obsahuje transakce TX1, TX2 a TX3, avšak chybí transakce, jejichž výstupy jsou využívány v rámci vstupů transakce TX1 a označené transakce TX3. V takovém případě agregované příznaky generované grafovou neuronovou sítí nemohou obsahovat informace o těchto chybějících transakcích, což je v rozporu s původním záměrem využívat kontext okolních transakcí pro přesnější klasifikaci.

transakce, ale znemožňuje to i samotné využití této datové sady k natrénování modelu použitelného ve forenzní praxi, jak již bylo zmíněno v kapitole 2.3.

4.3 Identifikace transakcí obsažených v datové sadě `Elliptic`

Znalost významu a hodnot většiny příznaků datové sady `Elliptic` umožnila identifikaci všech původních transakcí obsažených v této datové sadě. Proces identifikace byl následující:

1. Nejprve byla z již identifikovaných transakcí extrahována čísla bloků pro jednotlivé časové kroky. Tyto rozsahy bloků byly následně rozšířeny o 10 bloků na každou stranu, aby byl pokryt celý původní časový interval sběru transakcí.

2. Poté pro každý časový krok byly z aplikačního rozhraní aplikace Blockbook získány všechny transakce, které se nacházely v příslušných blocích. Pro každou z těchto transakcí byly vypočteny hodnoty deanonymizovaných příznaků.

3. Neidentifikované transakce byly seskupeny na základě shody všech příznaků (i těch nedeanonymizovaných) a shody v hranách obsažených v datové sadě `Elliptic`.

4. Následně pro každou skupinu byla vytvořena množina transakcí, které se shodují ve všech deanonymizovaných příznacích s příznaky této skupiny. Tuto množinu budeme dále považovat za množinu shodujících se transakcí.

5. Pokud kardinalita množiny shodujících se transakcí odpovídala kardinalitě samotné skupiny, byly všechny transakce této skupiny označeny jako identifikované a byly jim přiřazeny haše transakcí z množiny shodujících se transakcí. Tyto skupiny již dále nebyly zpracovávány.

6. Pro skupiny, jejichž množina shodujících se transakcí obsahovala více transakcí než obsahovala tato skupina, byl zkoumán vztah k již identifikovaným transakcím. Z množiny shodujících se transakcí byly odstraněny transakce, jež neměly vztah s již identifikovanými transakcemi, se kterými však dle hran původní datové sady vztah mít

měly. V případě, kdy se kardinalita této skupiny srovnala s kardinalitou množiny shodujících se transakcí, byla skupina zpracována dle bodu 5.

7. K identifikaci transakcí ve zbylých skupinách bylo využito empirického zpracování, kdy byly porovnávány nedeanonymizované příznaky k identifikaci konkrétní transakce.

4.4 Tvorba otevřené datové sady

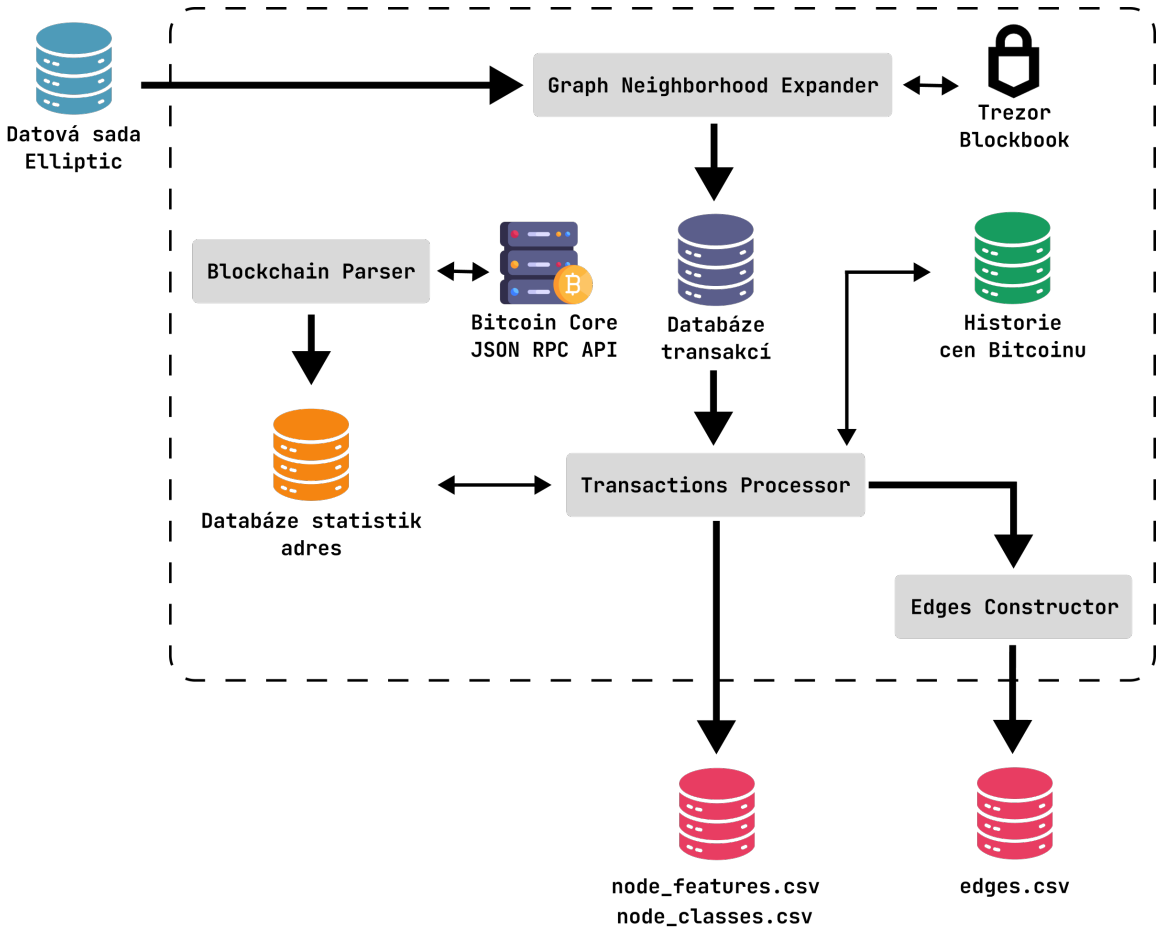
K vytvoření nové otevřené datové sady bylo potřeba navrhnout systém, jehož úkolem je sesbírat potřebná data a následně ze sesbíraných dat tuto datovou sadu sestavit. Tato kapitola se zabývá popisem navrženého systému k sestavení datové sady `OpenElliptic`. Vstupem tohoto systému je datová sada `Elliptic` a výstupem tak soubory datové sady `OpenElliptic`. Navržený systém se skládá z několika modulů. Modul Graph Neighborhood Expander, jehož úkolem je získání dat o jednotlivých transakcích a rozšíření původního transakčního grafu o chybějící okolní transakce. Modul Blockchain Parser postupnou analýzou bloků blockchainu vytváří databázi statistik adres, která je následně využita jako zdroj dat pro modul Transactions Processor, který pro každou transakci konstruuje výsledný vektor příznaků této transakce. Tento modul předává informace o existujících transakcích a jejich vstupech a výstupech modulu Edges Constructor, jenž má za úkol sestavit množinu hran mezi jednotlivými transakcemi obsaženými ve výsledné datové sadě. Schéma celého navrženého systému lze vidět na obrázku 4.6.

4.4.1 Modul Graph Neighborhood Expander

Prvním modulem využitým při konstrukci datové sady je *Graph Neighborhood Expander* (GNE). Tento modul má 2 klíčové úlohy. První úlohou je získání dat pro každou z transakcí. Toho je docíleno za využití aplikačního rozhraní aplikace Trezor Blockbook,[7] které pro každou z transakcí vrátí její obsah v JSON formátu společně i se vstupními adresami, které v samotné transakci v blockchainu sítě Bitcoin nejsou uloženy, a dále společně i s haši transakcí, které spotřebovávají některý z výstupů této transakce, čehož je využito v rámci rozšiřování transakčního grafu. Druhou klíčovou úlohou tohoto modulu je tedy samotné rozšíření transakčního grafu o relevantní okolní transakce tak, aby bylo zajištěno, že se ve vytvořené datové sadě budou nacházet všechny transakce, které se nacházejí v n-hopovém sousedství uzlu označené transakce, přičemž n je parametrem tohoto modulu. Tímto způsobem je řešen nedostatek původní datové sady `Elliptic`, ve které toto širší okolí u některých označených transakcí chybělo. Vstupem tohoto modulu je původní datová sada `Elliptic` společně se seznamem překladů původních identifikátorů transakcí na haše těchto transakcí, jehož tvorbou se zabývá kapitola 4.3. Výstupem tohoto modulu je pak databáze transakcí, která se skládá z uspořádaných pětic $(txId, hash, class, data, distance)$, přičemž samotný proces tvorby této databáze probíhá následovně:

1. *Inicializace.* Do databáze jsou vloženy transakce z datové sady `Elliptic` tak, že každá z transakcí je vložena jako pětice $(next\_int(), tx\_hash, tx\_class, null, d)$, kde $next\_int()$ značí funkci vracející vždy unikátní celé číslo, tx_hash značí haš právě vkládané transakce, tx_class značí třídu právě vkládané transakce, přičemž 0 značí třídu neoznačených transakcí, 1 značí třídu ilegálních transakcí a 2 značí třídu legálních transakcí, a $d = 0$ pokud $tx\_class = 1 \vee 2$, jinak $d = null$.

[7] https://trezor.io/learn/a/trezor-blockbook-explorer



Obrázek 4.6: Schéma systému určeného pro tvorbu datové sady `OpenElliptic`.

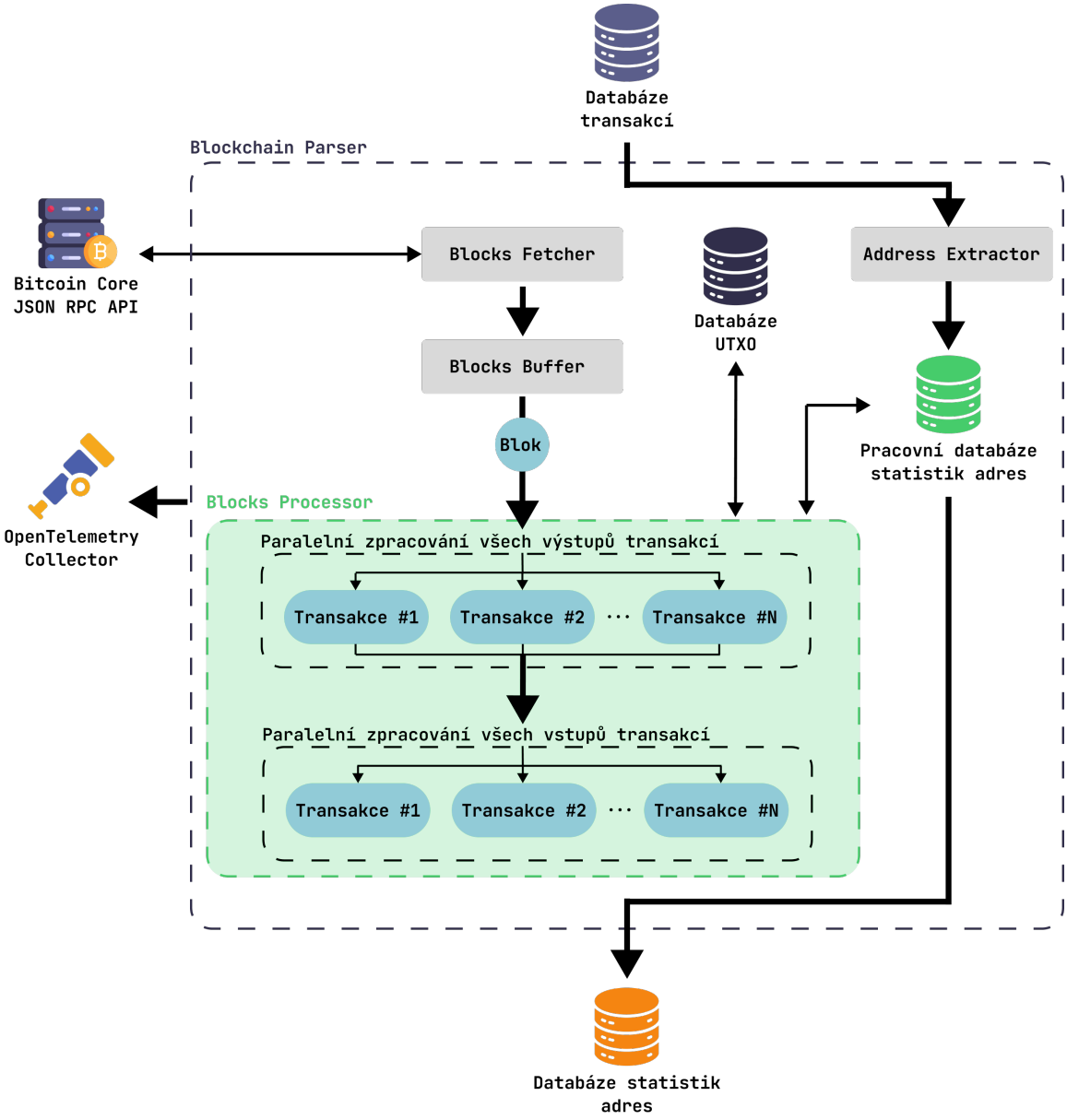
2. *Získání obsahu transakcí.* Pro každou transakci v databázi, kde $data = null$, je zavoláním aplikačního rozhraní aplikace Blockbook získán obsah dané transakce a ten je následně uložen do databáze transakcí.

3. *Expanze.* Pro každou transakci, pro kterou platí $distance < n$, kde n je parametr GNE modulu, extrahujeme z uložených dat její sousední transakce. Pokud se extrahovaná transakce nenachází v databázi transakcí, je do databáze vložena jako pětice $(next\_int(), tx\_hash, 0, null, d + 1)$, kde tx_hash značí haš extrahované transakce a d odpovídá hodnotě $distance$ transakce, z jejichž dat byla vkládaná transakce extrahována. V opačném případě, kdy se extrahovaná transakce v databázi již nachází, je $distance$ tohoto záznamu nastaven na $distance = MIN(distance, d+1)$, kde $distance$ je atribut existujícího záznamu a d odpovídá hodnotě $distance$ transakce, z jejichž dat byla transakce extrahována.

4. Pokud se v databázi nachází transakce, pro kterou platí $data = null$, potom se skočí na bod 2, jinak je proces tvorby databáze transakcí ukončen.

4.4.2 Modul Blockchain Parser

Po expanzi grafu následuje proces získání statistik jednotlivých adres použitých v příslušných transakcích expandovaného grafu. Tento proces je vykonáván modulem *Blockchain Parser*. Jeho úkolem je sesbírat statistiky adres použitých v příslušných transakcích, a to vždy v době vykonání dané transakce. Architektura tohoto modulu je schematicky vyobrazena na obrázku 4.7. Modul je implementován v jazyce C# s využitím aplikačního rámce .NET 8, přičemž tyto technologie byly zvoleny se záměrem zajištění lepší paralelizace zpracování jednotlivých transakcí v rámci zpracovávaných bloků.



Obrázek 4.7: Schéma Blockchain Parser modulu.

Vstupem tohoto modulu je databáze transakcí, ze kterých jsou v rámci prvního kroku vyextrahovány jednotlivé zapojené adresy, čímž je vytvořena pracovní databáze statistik adres. Výstupem tohoto modulu je pak vytvořená databáze statistik adres.

Po inicializaci databáze statistik adres je inicializována také databáze UTXO (databáze nespotřebovaných výstupů transakcí), jejíž obsah je při spuštění modulu prázdný.

Modul se skládá ze dvou hlavních komponent, které jsou zapojeny v modelu *producent-konzument*. Producentem bloků je komponenta *Blocks Fetcher*, jejíž úkolem je komunikace s aplikačním rozhraním aplikace Bitcoin Core.[8] Konzumentem je pak komponenta *Blocks Processor*, jejíž úlohou je postupně *analyzovat jednotlivé bloky* a dle jejich obsahu aktualizovat jednotlivé záznamy v databázi statistik adres a v databázi UTXO.

Komponenta Blocks Fetcher

Komponenta Blocks Fetcher naplňuje a udržuje *vyrovnávací paměť bloků*, které jsou poté v pořadí zpracovávány. Jednotlivé bloky jsou získávány *paralelně* z aplikačního rozhraní Bitcoin Core API s využitím *mechanismu posuvného okna* (angl. sliding window). Posuvné okno zajišťuje, že počet souběžných požadavků na Bitcoin Core API nepřesáhne mez, kdy by aplikace Bitcoin Core nestíhala požadavky odbavovat. Kromě toho mechanismus posuvného okna stanovuje horní hranici počtu bloků ve vyrovnávací paměti, což chrání komponentu před vyčerpáním veškeré dostupné paměti. Mechanismus posuvného okna je implementován následovně. Nechť k je velikost posuvného okna a n odpovídá číslu posledního bloku vybranému konzumentem z vyrovnávací paměti. Potom Blocks Fetcher získává obsah bloků ve výšce $n+1$, $n+2$ až $n+k$. Každý takto získaný obsah je uložen do vyrovnávací paměti.

Komponenta Blocks Processor

Úlohou komponenty Blocks Processor je samotná analýza jednotlivých bloků blockchainu sítě Bitcoin. Tato komponenta v pořadí po jednom vybírá bloky z vyrovnávací paměti bloků a zpracovává je ve dvou krocích následovně:

1. *Zpracování výstupů transakcí.* V rámci prvního kroku jsou všechny transakce obsažené ve zpracovávaném bloku rozhozeny napříč všemi vlákny a následně jsou paralelně zpracovávány jejich výstupy. V případě, kdy se shoduje adresa na výstupu s nějakou adresou v databázi statistik adres, jsou všechny záznamy této adresy, jejichž cílená výška je větší nebo rovna výšce aktuálně zpracovávaného bloku, aktualizovány úměrně k objemu na tomto výstupu. V takovém případě je tento výstup také přidán do databáze UTXO.

2. *Zpracování vstupů transakcí.* Všechny transakce obsažené ve zpracovávaném bloku jsou znovu rozhozeny napříč všemi vlákny a následně jsou paralelně zpracovávány jejich vstupy. Pokud se použitý výstup nenachází v databázi UTXO, je přeskočen. V případě, kdy se použitý výstup nachází v databázi UTXO, je z tohoto záznamu získána hodnota daného výstupu a použitá adresa, a tento nevyužitý výstup je z databáze UTXO odstraněn. Poté jsou aktualizovány všechny záznamy této adresy, jejichž cílená výška je větší nebo rovna výšce aktuálního zpracovávaného bloku.

Proces zpracovávání transakcí je rozdělen do dvou kroků z důvodu možné závislosti některých transakcí na výstupech transakcí ve stejném bloku. Transakce mohou využívat výstupy, které se v rámci bloku nacházejí před touto transakcí, tedy při sekvenčním zpracování transakcí jsou tyto využité výstupy již známy. Pro zajištění znalosti všech využitých nespotřebovaných výstupů jsou tedy v rámci paralelního zpracování nejprve zpracovány

[8] https://bitcoincore.org/

všechny výstupy, a až pak následně všechny vstupy všech transakcí obsažených ve zpracovávaném bloku.

4.4.3 Moduly Transactions Processor a Edges Constructor

Posledními dvěma moduly jsou modul Transactions Processor a modul Edges Constructor. Na rozdíl od dvou předchozích modulů, jejichž úkolem bylo získat data pro sestavení datové sady `OpenElliptic`, úlohou modulu Transactions Processor je sestavit samotné vektory příznaků transakcí a úkolem modulu Edges Constructor je sestavit finální množinu hran mezi těmito transakcemi.

4.5 Popis vytvořené datové sady `OpenElliptic`

Datová sada `OpenElliptic` se skládá z jednoho grafu transakcí, přičemž uzly tohoto grafu reprezentují transakce a hrany značí tok prostředků mezi těmito transakcemi. Tento graf se skládá z 5 635 022 uzlů a 28 079 403 hran mezi nimi, přičemž 46 564 uzlů z nich je označených, konkrétně se jedná o 42 019 legálních transakcí a 4 545 ilegálních transakcí. Jedná se tak o velmi nevyváženou datovou sadu. Pro každý označený uzel obsahuje tato datová sada všechny sousední transakce do vzdálenosti 2.

Ke každému uzlu grafu je přiřazen vektor 122 příznaků, přičemž příznaky se primárně skládají ze dvou částí — 52 příznaků je extrahováno z obsahu samotné transakce a zbylých 70 příznaků je získáno agregací příznaků adres použitých v této transakci. Seznam všech příznaků je uveden v příloze B. Mezi lokální příznaky patří celkový objem transakce, zaplacený poplatek, počet vstupů a počet výstupů, množství satoshi na jednotlivých vstupech a výstupech a další informace, které lze napřímo získat z obsahu této transakce a obsahu vstupních transakcí (objemy a adresy na spotřebovaných UTXO). Po vzoru práce Schnoering a spol. [22] se příznaky objemu a zaplaceného poplatku nacházejí ve vektoru příznaků transakce dvakrát — jednou v počtu satoshi a ve druhém případě v amerických dolarech dle hodinového kurzu v době provedení transakce.

Příznaky použitých adres nejsou vztaženy přímo ke konkrétní transakci, avšak jedná se o dostupné informace o dané adrese v čase této konkrétní transakce. Konkrétně jde o počet transakcí spojených s danou adresou, počet transakcí používajících adresu na vstupu, počet transakcí používajících adresu na výstupu, objem přijatých a odeslaných prostředků, počet aktuálně dostupných prostředků na této adrese a doba života této adresy (tj. doba od jejího prvního využití až do doby využití v rámci transakce, pro kterou jsou tyto příznaky počítány). Tyto příznaky adres jsou následně agregovány za pomoci minima, maxima, průměru, mediánu a rozpětí hodnot, a to zvlášť pro adresy použité na vstupu a na výstupu transakce. Postup výpočtu těchto příznaků adres je detailně popsán v kapitole 4.4.2.

4.5.1 Rozdělení datové sady

Před samotným využitím této datové sady pro trénování modelů strojového učení je zapotřebí tuto datovou sadu rozdělit na sadu trénovací, validační a testovací. Vytvořená sada se skládá z jednoho propojeného grafu — grafu pozadí, přičemž jednotlivé označené transakce nesou informaci o původním časovém kroku, ve kterém se nacházely.

Aby byla zajištěna porovnatelnost výsledků s výsledky na datové sadě `Elliptic`, převzal jsem její rozdělení datové sady. Označené transakce jsou rozděleny na základě jim přiřazených časových kroků, kdy trénovací datová sada obsahuje označené transakce z časového

kroku 1 až 29, validační sada obsahuje transakce z časového kroku 30–34 a testovací sada obsahuje transakce z časového kroku 35 až 49. K těmto označeným transakcím jsou poté přidány transakce, jejichž časová značka předchází začátku následující sady, tedy trénovací datová sada obsahuje pouze transakce, jež se na blockchainu objevily do konce časového kroku 29, validační datová sada poté transakce, jež se objevily do konce kroku 34, a testovací datová sada obsahuje všechny transakce z vytvořeného grafu pozadí. Tímto je zajištěno, že trénovaný model strojového učení nemá v průběhu trénování přístup k žádným informacím, které by nebyly známy v odpovídající době trénovací datové sady.

4.5.2 Předzpracování

Samotné příznaky vytvořené datové sady jsou v původní formě a nejsou nijak předzpracovány pro trénování modelů strojového učení. Před takovým využitím je vhodné provést předzpracování jednotlivých příznaků — konkrétně je standardní praxí numerické atributy normalizovat před jejich použitím ke trénování neuronových sítí a kategorické atributy převést na numerické například pomocí one-hot kódování. Sada `OpenElliptic` však žádné kategorické příznaky neobsahuje. Při normalizaci jednotlivých příznaků je vhodné spočítat normalizační parametry z trénovací datové sady, což může vést k horším výsledkům, ale zabrání to úniku informací ze sady validační a testovací. V rámci experimentů provedených na datové sadě `OpenElliptic` byly všechny numerické atributy normalizovány za pomoci *z-score* normalizace (viz kapitola 5).

Kapitola 5

Experimenty

V rámci této práce jsem provedl experimentální vyhodnocení modelů strojového učení zaměřených na detekci transakcí spojených s ilegálními aktivitami. Pro tyto experimenty jsem využil datovou sadu `Elliptic`, `Elliptic++` a vytvořenou datovou sadu `OpenElliptic`, které obsahují částečně označené transakční grafy sítě Bitcoin a o jejichž obsahu pojednává podrobněji kapitola 2.3 a kapitola 4.5.

Nejprve jsem provedl experimenty s tradičními přístupy strojového učení, jako je algoritmus Random Forest, XGBoost a vícevrstvý perceptron (MLP). Následně jsem se zaměřil na základní modely grafových neuronových sítí, konkrétně na konvoluční neuronové sítě a architekturu Graph Attention Network. Rovněž jsem provedl experimenty s jednou ze state-of-the-art architektur grafových neuronových sítí — architekturou DGA-GNN, která má otevřený zdrojový kód a díky tomu ji bylo možné snadno zařadit do mých experimentů a vyhodnotit její výkon.

Cílem těchto experimentů je nabídnout jednotné a korektní vyhodnocení výkonu jednotlivých metod, a to jak na původní anonymizované datové sadě `Elliptic` (resp. `Elliptic++`), tak primárně na nově vytvořené otevřené datové sadě `OpenElliptic`, která adresuje zásadní nedostatky datové sady `Elliptic` (resp. `Elliptic++`), jako je především špatná izolace testovací sady, která kompromituje experimentální výsledky jednotlivých modelů na této datové sadě, jak je podrobněji popsáno v kapitole 4.2. Většina dosavadních publikací má v této oblasti několik dalších nedostatků, které brání použití jimi poskytnutých výsledků pro porovnání s jinou metodou.

1. Publikace používají různé hodnoticí metriky pro vyhodnocení navržených modelů. Většina publikací na detekci ilegálních transakcí využívá jednu z F1 metrik, ale například Duan a spol. [5] a Chai a spol. [4] využívají k porovnávání výkonnosti modelů plochu pod ROC křivkou ROC-AUC a průměrnou přesnost AP, tedy plochu pod precision-recall křivkou, což v principu není špatně, avšak tato metrika není s F1 metrikou porovnatelná.

2. Některé publikace nevyužívají původního rozdělení datové sady `Elliptic` na trénovací, testovací a validační datovou sadu. Například Wang a spol. [26] využili náhodné rozdělení této datové sady, čímž efektivně znemožnili porovnání svých výsledků s výsledky ostatními, ale také nerespektovali chronologické rozdělení dat, jehož účelem bylo ověření výkonnosti modelu v induktivním zapojení a simulování situace, kdy model chceme využít na data, která se v budoucnu objeví.

3. Některé z publikací (Chai a spol. [4], Duan a spol. [5]) odstraňují z grafu transakcí neoznačené uzly, čímž však vzniká nová zcela nereálná situace a naučené vzorce na takto upravené datové sadě se v praxi bohužel nedají použít.

5.1 Nastavení prostředí experimentů

Datová sada `Elliptic` a její rozšířená verze `Elliptic++` byly pro experimenty rozděleny na trénovací, validační a testovací datovou sadu na základě pokynů z článku Weber a spol. [28]. Datová sada je rozdělena na základě časové osy, konkrétně uzly z 1–29 časového úseku jsou použity pro trénovací datovou sadu. Uzly z 30–34 časového úseku jsou použity pro validaci a uzly z časového úseku 35–49 jsou použity pro vytvoření testovací datové sady využité pro výsledné vyhodnocení výkonu natrénovaných modelů. Označená část datové sady `OpenElliptic` je pak totožně rozdělena na sadu testovací, validační a trénovací, přičemž neoznačená část této datové sady, kde jednotlivé transakce nejsou přímo zařazeny do konkrétního časového úseku, byla rozdělena na základě postupu popsaného v kapitole 4.5.1.

Modely XGBoost, Random Forest a vícevrstvý perceptron byly vyhodnoceny jak pouze na lokálních příznacích jednotlivých transakcí, tak na všech poskytnutých příznacích z datové sady `Elliptic`. Modely grafových neuronových sítí byly vyhodnoceny pouze na lokálních příznacích uzlů, jelikož úkolem těchto modelů je se naučit agregovat okolí uzlu do jeho výsledného vnoření, a tím eliminovat závislost na expertem vytvořených příznacích.

Hyperparametry jednotlivých modelů byly vybrány na základě experimentů s těmito modely. Testované hyperparametry byly vybrány na základě literatury Weber a spol. [28], Wang a spol. [26] a Elmougy a Liu [6], přičemž byl následně vybrán model s nejlepší F1 metrikou třídy transakcí spojených s ilegální aktivitou na validační datové sadě.

U algoritmu Random Forest bylo použito 25, 50, 60, 75, 90, 100 nebo 200 estimátorů a váženého gini indexu nebo vážené křížové entropie (angl. weighted cross entropy loss function) pro výběr nejlepšího rozdělení. U algoritmu XGBoost byl koeficient učení (angl. learning rate) nastaven na 0,05 a 0,1, maximální hloubka byla nastavena na 8, 10 a 12 a jako ztrátová funkce byla zvolena funkce vážené křížové entropie. Stejná ztrátová funkce byla využita i pro trénování všech neuronových sítí. Protože se ve všech třech případech jedná o nevyváženou datovou sadu, byly váhy jednotlivých tříd nastaveny v inverzním poměru zastoupení dané třídy v datové sadě na základě rovnice:

$$w_c = \frac{|C_1| + |C_2|}{2|C_c|}, \tag{5.1}$$

kde w_c značí váhu třídy c ($c = 1$ pro třídu transakcí spojených s ilegální aktivitou, $c = 2$ pro třídu legálních transakcí) a C_c značí množinu všech označených transakcí třídy c. Váha třídy legálních transakcí tedy byla nastavena na 4,59 a váha třídy transakcí spojených s ilegální aktivitou na 0,56. Vícevrstvý perceptron měl jednu skrytou vrstvu o velikosti 5, 10, 15, 20, 40 nebo 50 neuronů. Grafová neuronová síť i grafová attention síť měly dvě vrstvy, síť DGA-GNN byla jednovrstvá, což odpovídalo nastavení autorů této sítě pro klasifikaci datové sady `Elliptic`.

Jak před grafovou konvoluční síť, tak i před grafovou attention síť byl vložen vícevrstvý perceptron s dvěma skrytými vrstvami, následovanými batch normalizací a ReLU aktivační funkcí. Stejně tak byl vícevrstvý perceptron použit jako predikční hlava, která klasifikovala výstupní vnoření uzlu grafové neuronové sítě. Tato úprava vedla ke zlepšení výkonu obou sítí.

Pro úplnost, při experimentech s modelem Random Forest byla využita jeho implementace v knihovně scikit-learn,[1] pro experimenty s modelem XGBoost byla využita původní implementace tohoto algoritmu,[2] a k implementaci modelů neuronových sítí bylo využito knihovny PyTorch 2.4.0[3] v kombinaci s knihovnou Deep Graph Library (DGL).[4] Pro evaluaci modelu DGA-GNN byla využita její open-source implementace dostupná na stránce GitHub.[5] Průběh a výsledky jednotlivých experimentů byly zaznamenávány pomocí služby Weights and Biases.[6]

5.2 Metriky hodnocení jednotlivých metod

Datové sady `Elliptic`, `Elliptic++` a `OpenElliptic` jsou velmi nevyvážené, pozitivní případy (tj. transakce spojené s ilegální činností) tvoří pouze 9,8 % označené části datové sady. V rámci detekce transakcí spojených s ilegální činností je pro nás důležité, jak detekovat co nejvíce takových transakcí, tak i co nejmenší počet falešných detekcí. Pro sledování těchto dvou cílů slouží metrika přesnosti (angl. precision, rovnice 5.2), která odpovídá požadavku na co nejmenší počet falešných detekcí, a metrika recall (rovnice 5.3), která odpovídá druhému požadavku na detekci co nejvíce pozitivních případů. Abychom mohli modely navzájem porovnat v obou těchto oblastech, využívá se metriky $F1$ (rovnice 5.4), která kombinuje metriku přesnosti s metrikou recall. Tato metrika je v literatuře hojně využívána pro hodnocení modelů spojených s detekcí transakcí spojených s ilegální činností [15, 26, 28] a z toho důvodu jsem ji zvolil jako hlavní metriku pro porovnání testovaných modelů.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.4}$$

Mezi další metriky používané pro evaluaci modelů na takto nevyvážené datové sadě patří metrika ROC-AUC, která odpovídá ploše pod ROC křivkou, a metrika průměrné přesnosti (APS), která odpovídá ploše pod precision-recall křivkou.

5.3 Výsledky experimentů

Výsledky provedených experimentů na jednotlivých datových sadách `Elliptic`, `Elliptic++` a `OpenElliptic` jsou uvedeny v tabulkách 5.1, 5.2 a 5.3.

Z výsledků experimentů na původní datové sadě `Elliptic` a sadě `Elliptic++` vyplývá, že vícevrstvý perceptron je nejhůře se chovajícím modelem z testovacího vzorku modelů a čistě jeho využití pro identifikaci transakcí spojených s ilegálními aktivitami není možné. Jeho přesnost se u obou datových sad pohybovala pod 50 %, tedy bylo více falešných detekcí než těch správných. Nejvýkonnějším modelem z pohledu F1 metriky je v obou případech

[1] https://scikit-learn.org/stable/
[2] https://xgboost.readthedocs.io
[3] https://pytorch.org/
[4] https://www.dgl.ai/
[5] https://github.com/AtwoodDuan/DGA-GNN
[6] https://wandb.ai/site/

model Random Forest, avšak pouze v případě, kdy pracuje i s poskytnutými expertem vytvořenými agregovanými příznaky ze sousedů dané transakce. Přičemž podobné zlepšení při použití i agregovaných příznaků lze pozorovat také u modelu XGBoost. To potvrzuje prvotní hypotézu (viz kapitola 1), že pro identifikaci transakcí spojených s ilegální činností je důležité brát v potaz i kontext dané transakce, a ne pouze transakci samotnou. Nejvýkonnějším modelem z modelů grafových neuronových sítí byla z pohledu F1 metriky dvouvrstvá grafová attention síť s residuálním spojením, která se výkonností přibližuje modelu Random Forest. Skok výkonnosti při přidání residuálního spojení může značit, že agregace vlastností sousedů společně s vlastnostmi konkrétní zkoumané transakce způsobuje významnou ztrátu informací, které jsou pro klasifikaci dané transakce důležité.

Model	Precision	Recall	Illicit F1	APS	ROC-AUC
XGBoost$_{LF}$	0,783	0,679	0,727	0,765	0,920
XGBoost$_{AF}$	0,788	**0,733**	0,760	0,793	0,931
Random Forest$_{LF}$	0,810	0,718	0,761	0,777	0,892
Random Forest$_{AF}$	0,878	0,724	**0,794**	0,792	0,933
Vícevrstvý perceptron$_{LF}$	0,497	0,719	0,588	0,389	0,892
Vícevrstvý perceptron$_{AF}$	0,371	0,651	0,473	0,290	0,863
GCN$_{LF}$	0,830	0,612	0,705	0,727	0,914
GAT$_{LF}$	0,831	0,613	0,706	0,725	0,911
GAT-Residual$_{LF}$	**0,924**	0,662	0,771	0,772	0,913
DGA-GNN$_{LF}$	0,737	0,727	0,732	**0,797**	**0,938**

Tabulka 5.1: Výsledky experimentů provedených na datové sadě `Elliptic` [28]. *LF* značí využití pouze lokálních příznaků každého uzlu, *AF* značí využití všech příznaků. Nejlepší výsledek v dané metrice je zvýrazněn tučně.

	Precision	Recall	Illicit F1	APS	ROC-AUC
XGBoost$_{LF}$	0,718	0,692	0,705	0,766	0,916
XGBoost$_{AF}$	0,825	**0,738**	0,779	**0,797**	0,928
Random Forest$_{LF}$	0,793	0,725	0,757	0,773	0,889
Random Forest$_{AF}$	0,877	0,722	**0,792**	0,785	0,910
Vícevrstvý perceptron$_{LF}$	0,445	0,712	0,548	0,386	0,880
Vícevrstvý perceptron$_{AF}$	0,418	0,524	0,465	0,285	0,860
GCN$_{LF}$	0,765	0,628	0,690	0,727	0,915
GAT$_{LF}$	0,842	0,618	0,712	0,731	0,910
GAT-Residual$_{LF}$	**0,909**	0,662	0,766	0,773	0,921
DGA-GNN$_{LF}$	0,785	0,726	0,754	0,795	**0,933**

Tabulka 5.2: Výsledky experimentů provedených na datové sadě `Elliptic++` [6]. *LF* značí využití pouze lokálních příznaků každého uzlu, *AF* značí využití všech příznaků. Nejlepší výsledek v dané metrice je zvýrazněn tučně.

Ve výsledcích experimentů na nově vytvořené datové sadě `OpenElliptic` však lze pozorovat výraznější rozdíl mezi modely založenými na rozhodovacích stromech (Random Forest, XGBoost) a modely neuronových sítí. Model Random Forest i u této datové sady je nejlépe chovajícím se modelem a jeho výkon nebyl prakticky vůbec poznamenán opra-

vou příznaků spojených s použitými adresami, stejně tak nelze pozorovat žádný pokles výkonu na modelu XGBoost, který se z pohledu F1 metriky umístil na druhém místě. Na obrázku 5.1a a na obrázku 5.1b lze pozorovat rozdíl mezi významem jednotlivých příznaků pro model Random Forest natrénovaný na původní datové sadě `Elliptic` a model Random Forest natrénovaný na sadě `OpenElliptic`. Pozornost si zaslouží skutečnost, že model natrénovaný na sadě `Elliptic` přiřadil velkou významnost příznakům spojeným s počtem odeslaných a přijatých satoshi na použitých adresách, přičemž u modelu natrénovaném na datové sadě `OpenElliptic` se tyto příznaky vůbec neumístily v 20 nejvýznamnějších příznacích. V těchto nejvýznamnějších příznacích se ani u jednoho modelu neumístil samotný objem klasifikované transakce. To může správně odpovídat faktu, že nelegální činnost může být provozována jak v malých částkách, tak ve velkých, a tudíž objem transakce by neměl být významným faktorem v rozhodnutí, zda se jedná o legální či ilegální transakci. Změnu ve významnosti počtu přijatých a odeslaných satoshi použitých adres lze interpretovat podobně — samotný objem transakcí spojených s použitou adresou by neměl být významným identifikátorem pro klasifikaci transakce a původní model spíše takto rozpoznával příznaky adres, které se nacházely zároveň v trénovací a testovací datové sadě, což potvrzuje špatný návrh datové sady `Elliptic`.

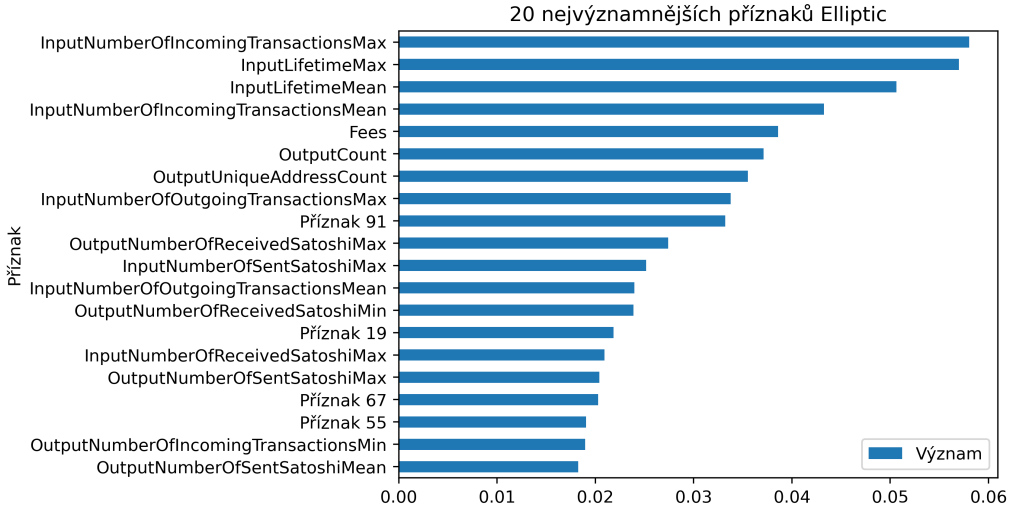
Grafové neuronové sítě v případě experimentů na sadě `OpenElliptic` dopadly významně hůře. Konvoluční grafová neuronová síť (GCN) a grafová attention síť (GAT) dosáhly menší přesnosti jak 30 % a ze všech modelů dosáhly nejmenšího F1 skóre. Na druhou stranu, grafová attention síť s residuálními spojeními (GAT-Residual) dosáhla F1 skóre 0,66, což je výrazně horší než skóre, které získal tento model na datové sadě `Elliptic`, avšak nejedná se o nepoužitelný model jako v případě grafové konvoluční sítě a grafové attention sítě bez reziduálních spojení. Zde je vhodné podotknout, že na validační datové sadě tento model dosáhl F1 skóre 0,88, což překonalo i výkon modelu Random Forest.

Příčinou špatného chování modelů GCN a GAT může být over-smoothing problém (viz kapitola 3.2.4), jenž bude způsoben přidáním chybějícího okolí klasifikovaných transakcí a který adresuje využití reziduálních spojení. Přidání informací v podobě chybějícího okolí pravděpodobně způsobuje i pokles výkonu u modelu GAT-Residual, konkrétně menší generalizaci natrénovaného modelu, kdy se natrénované vzorce nacházejí ve validační datové sadě, jak ukazuje dobrý výkon na této sadě, avšak již se nenachází v sadě testovací.

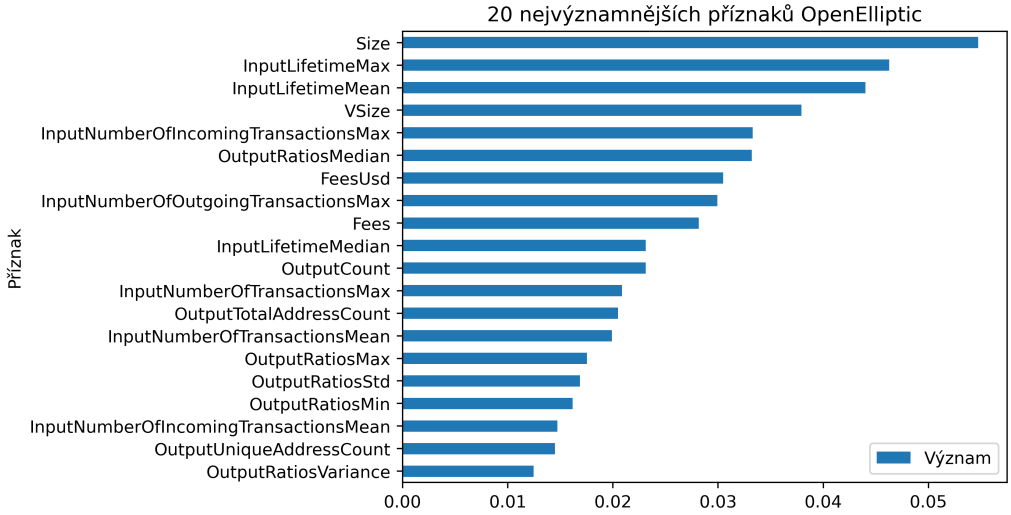
Ve výsledku se dá konstatovat, že se v rámci této práce nepodařilo prokázat lepší výkon grafových neuronových sítí vůči tradičním modelům strojového učení, konkrétně modelu Random Forest a XGBoost, na úloze klasifikace transakcí spojených s ilegální aktivitou. Model Random Forest prokázal schopnost lepší generalizace bez potřeby velkého množství trénovacích vzorků. Dále lze konstatovat, že využití grafových neuronových sítí ke klasifikaci transakcí kryptoměny Bitcoin trpí over-smoothing problémem, který lze adresovat použitím reziduálních spojení, jak ukazují výsledky modelu GAT-Residual. V rámci experimentů na datové sadě `OpenElliptic` se však tato grafová síť přeučila (angl. overfitted) na trénovací sadě. Tento problém se dá adresovat rozšířením původní datové sady o další označené transakce, což nově vytvořená datová sada `OpenElliptic` umožňuje.

	Precision	Recall	Illicit F1	APS	ROC-AUC
XGBoost	**0,892**	0,618	0,730	0,717	0,896
Random Forest	0,853	0,707	**0,773**	**0,780**	0,917
Vícevrstvý perceptron	0,595	0,525	0,558	0,378	0,880
GCN	0,291	**0,763**	0,421	0,410	0,898
GAT	0,278	0,702	0,399	0,428	0,885
GAT-Residual	0,671	0,656	0,663	0,707	**0,922**
DGA-GNN	0.526	0,695	0,599	0,436	0,881

Tabulka 5.3: Výsledky experimentů provedených na vytvořené datové sadě `OpenElliptic`. Hodnoty v závorce značí výsledek odpovídajícího modelu na validační datové sadě. Nejlepší výsledek v dané metrice je zvýrazněn tučně.



(a)



(b)

Obrázek 5.1: Graf 20 nejvýznamnějších příznaků pro Random Forest klasifikátor trénovaný na datové sadě (a) `Elliptic`, (b) `OpenElliptic`.

Kapitola 6

Závěr

Tato práce si kladla za cíl představit možnosti využití metod hlubokého učení, konkrétně grafových neuronových sítí (GNN), pro forenzní analýzu blockchainu sítě Bitcoin. Nejprve byly shrnuty základní principy kryptoměny Bitcoin, včetně popisu jeho klíčových kryptografických primitiv, struktury blockchainu a transakcí. Následně byly představeny hlavní postupy, které se v literatuře objevují v rámci kontextu analýzy blockchainu sítě Bitcoin, včetně dostupných datových sad (`Elliptic`, `Elliptic++` a `BitcoinTemporalGraph`). Byly identifikovány případy užití neuronových sítí pro analýzu sítě Bitcoin, kterými jsou *detekce transakcí spojených s ilegální činností*, *detekce praní špinavých peněz* pomocí klasifikace podgrafů, *deanonymizace transakcí* a *identifikace služeb mixérů*.

Z hlediska metod hlubokého učení byla popsána koncepce grafových neuronových sítí postavených na principu *message passing* a byly představeny klíčové architektury, jako jsou grafové konvoluční sítě, GraphSAGE a grafové attention sítě. V praktické části pak byla analyzována nejpoužívanější anonymizovaná datová sada `Elliptic`. Za pomoci reverzního inženýrství se podařilo deanonymizovat 47 % původních příznaků a identifikovat všechny původní transakce obsažené v této datové sadě. To vedlo ke zjištění, že datová sada `Elliptic` má několik nedostatků, které kompromitují naměřené výsledky na této datové sadě a představují tak určitá omezení z hlediska reálného nasazení natrénovaných modelů. Mezi hlavní nedostatky lze zařadit špatnou izolaci trénovací a testovací datové sady a chybějící okolí klasifikovaných transakcí. V rámci této práce byla proto navržena nová otevřená datová sada `OpenElliptic`, která vychází z původní datové sady `Elliptic`, avšak opravuje všechny identifikované nedostatky.

Dále byly provedeny experimenty s tradičními klasifikátory (Random Forest, XGBoost, vícevrstvý perceptron) i vybranými modely grafových neuronových sítí. Experimenty byly realizovány jak na původní datové sadě `Elliptic`, tak i na nově vytvořené datové sadě `OpenElliptic` s cílem zajistit korektní a jednotné porovnání jednotlivých metod strojového učení. Z výsledků experimentů vyplývá, že v obou případech je nejlépe chovajícím se modelem model Random Forest, který dokázal nejlépe generalizovat. Z výsledků experimentů na datové sadě `Elliptic` lze konstatovat, že přidání globálních příznaků má v případě modelů založených na rozhodovacích stromech pozitivní dopad na výsledné F1 skóre a lze tedy potvrdit první výchozí hypotézu této práce o pozitivním efektu využití informací z okolí klasifikované transakce. Grafová attention síť s reziduálními spojeními se v případě datové sady `Elliptic` umístila na druhém místě hned za modelem Random Forest, avšak v případě sady `OpenElliptic` bylo možné pozorovat výrazný propad ve výsledném F1 skóre tohoto modelu, který byl pravděpodobně způsoben přeučením této sítě na trénovací datové sadě. To by mělo být řešitelné rozšířením trénovací a validační datové sady, což otevřená datová

sada `OpenElliptic` na rozdíl od původní nejpoužívanější datové sady `Elliptic` dovoluje. Výsledky experimentálního ověření proto nejsou v případě prvotní hypotézy o superioritě grafových neuronových sítí dostatečně průkazné pro její potvrzení či zamítnutí.

Přínosy této práce

V rámci této diplomové práce se mi podařilo:

- zanalyzovat klasické i moderní přístupy (včetně GNN) k forenzní analýze blockchainu sítě Bitcoin,
- identifikovat případy užití neuronových sítí pro analýzu blockchainu sítě Bitcoin,
- provést sjednocené experimenty, které mohou sloužit jako základní linie pro porovnání dalších přístupů,
- identifikovat problémy dostupných datových sad způsobené anonymizací příznaků,
- deanonymizovat nejpoužívanější datovou sadu `Elliptic` a odhalit její významné nedostatky,
- stanovit hlavní výzvy pro zlepšení použitelnosti pokročilých metod v reálné forenzní praxi,
- vytvořit novou otevřenou datovou sadu `OpenElliptic`, která může sloužit jako základní kámen pro využití grafových neuronových sítí a dalších metod strojového učení v reálných vyšetřovacích nástrojích, které budou moci bezpečnostní složky využít v boji s finanční kriminalitou.

Literatura

[1] Alarab, I. a Prakoonwit, S. Graph-Based LSTM for Anti-money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data. *Neural Processing Letters*, 2022, sv. 55, s. 1–19. ISSN 1370-4621.

[2] Antonopoulos, A. M. a Harding, D. A. *Mastering Bitcoin: Programming the Open Blockchain*. 3rd ed. O'Reilly Media, Inc., listopad 2023. ISBN 978-1-09-815009-9.

[3] Bellei, C.; Xu, M.; Phillips, R.; Robinson, T.; Weber, M. et al. *The Shape of Money Laundering: Subgraph Representation Learning on the Blockchain with the Elliptic2 Dataset*. arXiv, 2024. Dostupné z: http://arxiv.org/abs/2404.19109.

[4] Chai, Z.; You, S.; Yang, Y.; Pu, S.; Xu, J. et al. Can Abnormality be Detected by Graph Neural Networks? In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, červenec 2022, s. 1945–1951. ISBN 978-1-956792-00-3.

[5] Duan, M.; Zheng, T.; Gao, Y.; Wang, G.; Feng, Z. et al. DGA-GNN: Dynamic Grouping Aggregation GNN for Fraud Detection. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Březen 2024, sv. 38, č. 10, s. 11820–11828. ISBN 978-1-57735-887-9.

[6] Elmougy, Y. a Liu, L. Demystifying Fraudulent Transactions and Illicit Nodes in the Bitcoin Network for Financial Forensics. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, Srpen 2023, s. 3979–3990. KDD '23. ISBN 979-8-4007-0103-0.

[7] Eric Lombrozo, P. W. *Segregated Witness (Consensus layer)* online. 2015. Dostupné z: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki. [cit. 2025-01-12].

[8] Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O. a Dahl, G. E. Neural Message Passing for Quantum Chemistry. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 06–11 srpen 2017, sv. 70, s. 1263–1272. Proceedings of Machine Learning Research. ISBN 978-1-5108-5514-4.

[9] Hamilton, W.; Ying, Z. a Leskovec, J. Inductive Representation Learning on Large Graphs. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, sv. 30. ISBN 978-1-5108-6096-4.

[10] Hamilton, W. L. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan and Claypool, sv. 14, č. 3, s. 1–159. ISSN 1939-4608.

[11] He, K.; Zhang, X.; Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. ISBN 978-1-4673-8852-8.

[12] Hu, X.; Gui, M.; Cheng, G.; Li, R. a Wu, H. Multi-class Bitcoin mixing service identification based on graph classification. *Digital Communications and Networks*, 2024, sv. 10, č. 6, s. 1881–1893. ISSN 2352-8648.

[13] Kipf, T. N. a Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In: *International Conference on Learning Representations*. 2017. ISBN 978-1-7138-7271-9.

[14] Lee, C.; Maharjan, S.; Ko, K. a Hong, J. W.-K. Toward Detecting Illegal Transactions on Bitcoin Using Machine-Learning Methods. In: Zheng, Z.; Dai, H.-N.; Tang, M. a Chen, X., ed. *Blockchain and Trustworthy Systems*. Singapore: Springer Singapore, 2020, s. 520–533. ISBN 978-981-15-2777-7.

[15] Li, Y.; Cai, Y.; Tian, H.; Xue, G. a Zheng, Z. Identifying Illicit Addresses in Bitcoin Network. In: Zheng, Z.; Dai, H.-N.; Fu, X. a Chen, B., ed. *Blockchain and Trustworthy Systems*. Singapore: Springer, 2020, s. 99–111. ISBN 978-981-15-9213-3.

[16] Li, Z. a He, E. Graph Neural Network-Based Bitcoin Transaction Tracking Model. *IEEE Access*, 2023, sv. 11, s. 62109–62120. ISSN 2169-3536.

[17] Lo, W. W.; Kulatilleke, G. K.; Sarhan, M.; Layeghy, S. a Portmann, M. Inspection-L: self-supervised GNN node embeddings for money laundering detection in bitcoin. *Applied Intelligence*, srpen 2023, sv. 53, č. 16, s. 19406–19417. ISSN 1573-7497.

[18] Merkle, R. C. A Digital Signature Based on a Conventional Encryption Function. In: Pomerance, C., ed. *Advances in Cryptology — CRYPTO '87*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, s. 369–378. ISBN 978-3-540-48184-3.

[19] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. online, srpen 2008. Dostupné z: https://bitcoin.org/bitcoin.pdf. [cit. 2025-01-11].

[20] Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y. et al. *Graph2vec: Learning Distributed Representations of Graphs*. 2017. Dostupné z: https://arxiv.org/abs/1707.05005.

[21] Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T. et al. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: *Proceedings of the AAAI conference on artificial intelligence*. 2020, sv. 34, č. 04, s. 5363–5370. ISBN 978-1-7138-2732-0.

[22] Schnoering, H. a Vazirgiannis, M. Bitcoin research with a transaction graph dataset. *Scientific Data*, 2025, sv. 12, č. 1, s. 404. ISSN 2052-4463.

[23] Song, J.; Zhang, S.; Zhang, P.; Park, J.; Gu, Y. et al. Illicit Social Accounts? Anti-Money Laundering for Transactional Blockchains. *IEEE Transactions on Information Forensics and Security*, 2025, sv. 20, s. 391–404. ISSN 1556-6021.

[24] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P. et al. Graph Attention Networks. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings*. Curran Associates, 2018. ISBN 978-1-7138-7272-6.

[25] Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y. et al. Deep Graph Infomax. In: *International Conference on Learning Representations (ICLR 2019)*. Curran Associates, Inc., 2019. ISBN 978-1-7138-7273-3.

[26] Wang, Q.; Tsai, W.-T. a Du, B. RMGANets: reinforcement learning-enhanced multi-relational attention graph-aware network for anti-money laundering detection. *Complex & Intelligent Systems*, listopad 2024, sv. 11, č. 1, s. 5. ISSN 2198-6053.

[27] Wang, X. a Zhang, M. GLASS: GNN with Labeling Tricks for Subgraph Representation Learning. In: *International Conference on Learning Representations*. 2022. ISBN 979-8-3313-0009-8.

[28] Weber, M.; Domeniconi, G.; Chen, J.; Weidele, D. K. I.; Bellei, C. et al. *Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics*. arXiv, červenec 2019. Dostupné z: http://arxiv.org/abs/1908.02591.

[29] Wu, L.; Cui, P.; Pei, J.; Zhao, L. a Guo, X. Graph Neural Networks: Foundation, Frontiers and Applications. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2023, s. 5831–5832. KDD '23. ISBN 979-8-4007-0103-0.

[30] Xu, K.; Hu, W.; Leskovec, J. a Jegelka, S. How Powerful are Graph Neural Networks? In: *International Conference on Learning Representations (ICLR 2019)*. Curran Associates, Inc., 2019. ISBN 978-1-7138-7273-3.

[31] Yi, X.; Yang, X.; Kelarev, A.; Lam, K. Y. a Tari, Z. Cryptographic Primitives. In: *Blockchain Foundations and Applications*. Cham: Springer International Publishing, 2022, s. 1–24. ISBN 978-3-031-09670-9.

Příloha A

Význam příznaků transakcí v datové sadě `Elliptic`

Příznak	Význam
Příznak 1	Časový krok
Příznak 2	Objem transakce
Příznak 3	Poplatek
Příznak 5	Počet vstupů
Příznak 6	Počet výstupů
Příznak 7	Počet unikátních adres na vstupu
Příznak 9	Minimální vstupní objem
Příznak 10	Maximální vstupní objem
Příznak 12	Průměrný vstupní objem
Příznak 15	Počet unikátních adres na výstupu
Příznak 17	Minimální výstupní objem
Příznak 18	Maximální výstupní objem
Příznak 20	Průměrný výstupní objem

Tabulka A.1: Deanonymizované lokální příznaky transakce vypočtené z obsahu samotné transakce.

Příznak	Význam
Příznak 23	Minimum z počtu přijatých satoshi adres na vstupu
Příznak 24	Maximum z počtu přijatých satoshi adres na vstupu
Příznak 26	Průměr z počtu přijatých satoshi adres na vstupu
Příznak 29	Minimum z počtu odeslaných satoshi adres na vstupu
Příznak 30	Maximum z počtu odeslaných satoshi adres na vstupu
Příznak 32	Průměr z počtu odeslaných satoshi adres na vstupu
Příznak 35	Minimum z počtu nevyčerpaných satoshi na adresách na vstupu
Příznak 36	Maximum z počtu nevyčerpaných satoshi na adresách na vstupu
Příznak 38	Průměr z počtu nevyčerpaných satoshi na adresách na vstupu
Příznak 41	Minimum z počtu příchozích transakcí adres na vstupu
Příznak 42	Maximum z počtu příchozích transakcí adres na vstupu
Příznak 44	Průměr z počtu příchozích transakcí adres na vstupu
Příznak 47	Minimum z počtu odchozích transakcí adres na vstupu
Příznak 48	Maximum z počtu odchozích transakcí adres na vstupu
Příznak 50	Průměr z počtu odchozích transakcí adres na vstupu
Příznak 53	Minimum z dob života adres na vstupu
Příznak 54	Maximum z dob života adres na vstupu
Příznak 56	Průměr z dob života adres na vstupu

Tabulka A.2: Deanonymizované lokální příznaky transakce agregující příznaky použitých adres na vstupu transakce.

Příznak	Význam
Příznak 59	Minimum z počtu přijatých satoshi adres na výstupu
Příznak 60	Maximum z počtu přijatých satoshi adres na výstupu
Příznak 62	Průměr z počtu přijatých satoshi adres na výstupu
Příznak 65	Minimum z počtu odeslaných satoshi adres na výstupu
Příznak 66	Maximum z počtu odeslaných satoshi adres na výstupu
Příznak 68	Průměr z počtu odeslaných satoshi adres na výstupu
Příznak 71	Minimum z počtu nevyčerpaných satoshi na adresách na výstupu
Příznak 72	Maximum z počtu nevyčerpaných satoshi na adresách na výstupu
Příznak 74	Průměr z počtu nevyčerpaných satoshi na adresách na výstupu
Příznak 77	Minimum z počtu příchozích transakcí adres na výstupu
Příznak 78	Maximum z počtu příchozích transakcí adres na výstupu
Příznak 80	Průměr z počtu příchozích transakcí adres na výstupu
Příznak 83	Minimum z počtu odchozích transakcí adres na výstupu
Příznak 84	Maximum z počtu odchozích transakcí adres na výstupu
Příznak 86	Průměr z počtu odchozích transakcí adres na výstupu
Příznak 89	Minimum z dob života adres na výstupu
Příznak 90	Maximum z dob života adres na výstupu
Příznak 92	Průměr z dob života adres na výstupu

Tabulka A.3: Deanonymizované lokální příznaky transakce agregující příznaky použitých adres na výstupu transakce.

Příloha B

Formát datové sady `OpenElliptic`

Název	Význam
TimeStep	Časový krok transakce
BlockHeight	Výška bloku, ve kterém se transakce nachází
Value	Objem transakce v satoshi
ValueUsd	Objem transakce v USD
Fees	Poplatek transakce v satoshi
FeesUsd	Poplatek transakce v USD
Size	Velikost transakce
VSize	Virtuální velikost transakce
FeeValueRatio	Poměr poplatku k objemu transakce
VinVoutRatio	Poměr počtu vstupů k počtu výstupů
CoinbaseTransaction	1 pokud se jedná o coinbase transakci, jinak 0
TxInVoutRatio	Počet vstupních transakcí vůči počtu výstupů
NumberOfInputTransactions	Počet vstupních transakcí
NumberOfSameAddresses	Počet adres, které jsou využity jak na vstupu tak na výstupu
ForeignAddressesToAllRatio	Poměr počtu adres, které jsou využity pouze na výstupu k celkovému počtu použitých adres
ForeignValueToAllRatio	Poměr hodnoty odeslané na adresy nepoužité na vstupu k celému objemu transakce

Tabulka B.1: Seznam prvních 16 lokálních příznaků transakce v datové sadě `OpenElliptic`. Každá transakce má vedle vektoru příznaků také svůj unikátní identifikátor, původní haš a třídu, do které transakce spadá. Pokud je hodnota jakéhokoliv příznaku transakce nedefinována, je pole v tabulce transakcí prázdné.

Název	Význam
InputCount	Počet vstupů transakce
InputMin	Nejmenší hodnota vstupu v satoshi
InputMinUsd	Nejmenší hodnota vstupu v USD
InputMax	Největší hodnota vstupu v satoshi
InputMaxUsd	Největší hodnota vstupu v USD
InputStd	Směrodatná odchylka hodnot vstupů v satoshi
InputMean	Průměr hodnot vstupů v satoshi
InputMeanUsd	Průměr hodnot vstupů v USD
InputMedian	Medián hodnot vstupů v satoshi
InputMedianUsd	Medián hodnot vstupů v USD
InputRatiosMin	Nejmenší poměr hodnoty vstupu k objemu transakce
InputRatiosMax	Největší poměr hodnoty vstupu k objemu transakce
InputRatiosStd	Směrodatná odchylka poměrů hodnot vstupů k objemu transakce
InputRatiosVariance	Rozptyl poměrů hodnot vstupů k objemu transakce
InputRatiosMedian	Medián poměrů hodnot vstupů k objemu transakce
InputTotalAddressCount	Počet adres na vstupu
InputUniqueAddressCount	Počet unikátních adres na vstupu
InputUniqueToTotalAddressRatio	Poměr unikátních adres na vstupu k celkovému počtu adres na vstupu

Tabulka B.2: Seznam lokálních příznaků transakce v datové sadě `OpenElliptic`, které jsou založeny na informacích ze vstupů této transakce. V případě, kdy je počet vstupů transakce nulový, jsou všechny tyto příznaky nulové.

Název	Význam
OutputCount	Počet výstupů transakce
OutputMin	Nejmenší hodnota výstupu v satoshi
OutputMinUsd	Nejmenší phodnota výstupu v USD
OutputMax	Největší hodnota výstupu v satoshi
OutputMaxUsd	Největší hodnota výstupu v USD
OutputStd	Směrodatná odchylka hodnot výstupů v satoshi
OutputMean	Průměr hodnot výstupů v satoshi
OutputMeanUsd	Průměr hodnot výstupů v USD
OutputMedian	Medián hodnot výstupů v satoshi
OutputMedianUsd	Medián hodnot výstupů v USD
OutputRatiosMin	Nejmenší poměr hodnoty výstupu k objemu transakce
OutputRatiosMax	Největší poměr hodnoty výstupu k objemu transakce
OutputRatiosStd	Směrodatná odchylka poměrů hodnot výstupů k objemu transakce
OutputRatiosVariance	Rozptyl poměrů hodnot výstupů k objemu transakce
OutputRatiosMedian	Medián poměrů hodnot výstupů k objemu transakce
OutputTotalAddressCount	Počet adres na výstupu
OutputUniqueAddressCount	Počet unikátních adres na výstupu
OutputUniqueToTotalAddressRatio	Poměr unikátních adres na výstupu k celkovému počtu adres na výstupu

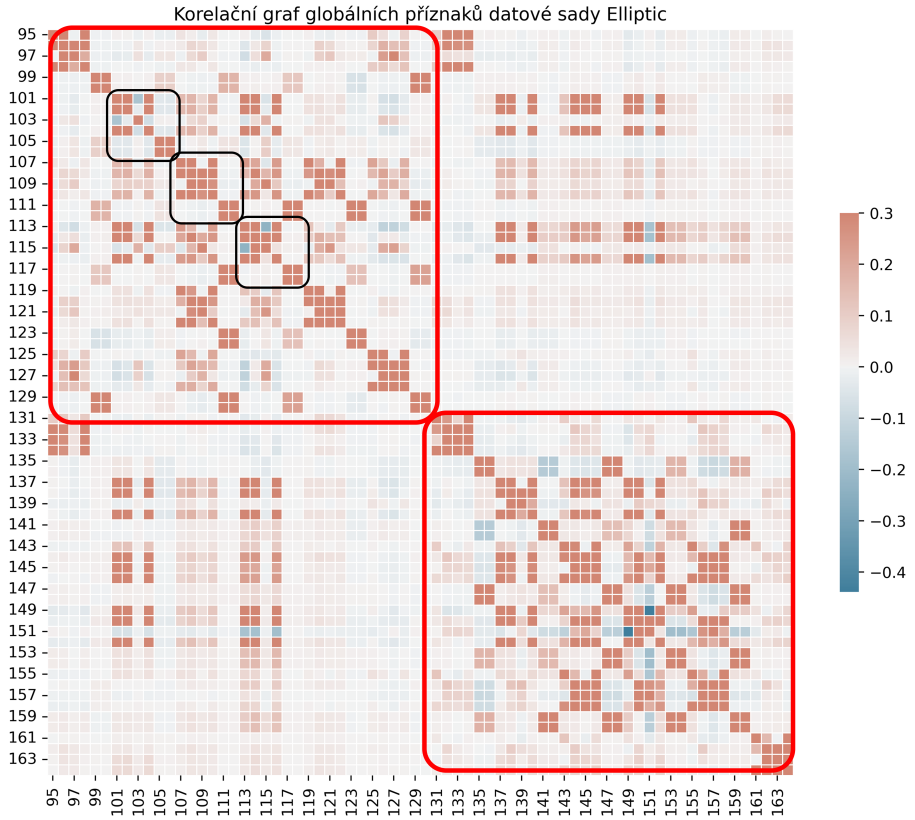
Tabulka B.3: Seznam lokálních příznaků transakce v datové sadě `OpenElliptic`, které jsou založeny na informacích z výstupů této transakce.

Název	Význam
NumberOfTransactions	Počet transakcí
NumberOfOutgoingTransactions	Počet odchozích transakcí
NumberOfIncomingTransactions	Počet příchozích transakcí
NumberOfSatoshis	Počet nevyčerpaných satoshi
NumberOfReceivedSatoshis	Počet přijatých satoshi
NumberOfSentSatoshis	Počet odeslaných satoshi
Lifetime	Doba života adresy

Tabulka B.4: Seznam příznaků adres, ze kterých jsou vypočteny agregované příznaky adres použitých na vstupu (Input) a výstupu (Output). Pro každý z těchto příznaků je ve výsledném vektoru příznaků transakce agregovaný příznak minimum (Min), maximum (Max), průměr (Mean), medián (Median) a rozsah hodnot (Range). Název průměrného počtu transakcí na adresách na vstupu pak je InputNumberOfTransactionsMean. V případě, kdy transakce nemá žádné vstupní, respektive výstupní adresy, jsou všechny tyto agregované příznaky nulové.

Příloha C

Korelační graf globálních příznaků transakcí v datové sadě `Elliptic`



Obrázek C.1: Korelační graf globálních příznaků transakcí v datové sadě `Elliptic`. Na grafu lze pozorovat opakující se vzor, kdy je vždy čtveřice po sobě jdoucích korelovaných příznaků následovaná dvojicí spolu souvisejících příznaků. V rámci této práce se povedlo odhalit, že se jedná vždy o šestici příznaků, které jsou agregací stejných vstupních dat, jako je například počet prostředků na jednotlivých vstupních adresách. Na obrázku lze dále pozorovat, že první polovinu globálních příznaků tvoří příznaky ze vstupních transakcí a druhou polovinu tvoří ty stejné příznaky z výstupních transakcí.

Příloha D

Obsah externí přílohy

Adresářová struktura

```
├── src - Adresář obsahující zdrojové soubory programů
│   ├── collector - Zdrojové soubory aplikace na tvorbu datové sady
│   ├── notebooks - Zdrojové soubory Jupyter sešitů
│   ├── experiments - Zdrojové soubory experimentů
│   └── ...
├── latex - Adresář obsahující zdrojové soubory pro text práce
├── datasets - Adresář obsahující využité datové sady
├── README.md - Návod k použití
├── master_thesis.pdf - Text práce
├── master_thesis_print.pdf - Text práce pro tisk
├── environment.yml - Conda prostředí pro operační systém Linux
├── requirements.txt - Seznam potřebných knihoven mimo Conda prostředí
└── ...
```