



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

**KLASIFIKACE SÍŤOVÉHO PROVOZU POMOCÍ
CONTRASTIVE LEARNING**

NETWORK TRAFFIC CLASSIFICATION USING CONTRASTIVE LEARNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB ČOČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. KAMIL JEŘÁBEK, Ph.D.

BRNO 2025

Zadání bakalářské práce



Ústav: Ústav informačních systémů (UIFS) 163746
Student: **Čoček Jakub**
Program: Informační technologie
Název: **Klasifikace síťového provozu pomocí contrastive learning**
Kategorie: Počítačové sítě
Akademický rok: 2024/25

Zadání:

1. Seznamte se s problematikou analýzy síťové komunikace pomocí síťových toků. Nastudujte metody self-supervised learning a konkrétně se zaměřte na metody contrastive learning a práce doporučené vedoucím v oblasti počítačových sítí.
2. Navrhněte model a postup pro reprodukci experimentů provedených v doporučených pracech, nad poskytnutými datasety.
3. Po domluvě s vedoucím implementujte vybrané augmentace, které budou použité pro trénink modelu.
4. Experimentálně implementujte model s použitím knihovny PyTorch.
5. Zhodnotěte výsledky a porovnejte se s výsledky dosaženými ve vybraných pracech.
6. Ověřte postup a vyhodnotěte výsledky na datové sadě z reálné sítě.

Literatura:

- Chen, T., Kornblith, S., Norouzi, M. and Hinton, G., 2020, November. A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597-1607). PMLR.
- Wang, C., Finamore, A., Michiardi, P., Gallo, M. and Rossi, D., 2024, March. Data Augmentation for Traffic Classification. In *International Conference on Passive and Active Network Measurement* (pp. 159-186). Cham: Springer Nature Switzerland.
- Horowicz, E., Shapira, T. and Shavitt, Y., 2022, October. A few shots traffic classification with mini-flowpic augmentations. In *Proceedings of the 22nd ACM Internet Measurement Conference* (pp. 647-654).
- Finamore, A., Wang, C., Krolikowski, J., Navarro, J.M., Chen, F. and Rossi, D., 2023, October. Replication: Contrastive Learning and Data Augmentation in Traffic Classification Using a Flowpic Input Representation. In *Proceedings of the 2023 ACM on Internet Measurement Conference* (pp. 36-51).

Při obhajobě semestrální části projektu je požadováno:

Body zadání 1-3.

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

Vedoucí práce: **Jeřábek Kamil, Ing., Ph.D.**

Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.

Datum zadání: 1.11.2024

Termín pro odevzdání: 14.5.2025

Datum schválení: 22.10.2024

Abstrakt

Tato bakalářská práce se zaměřuje na klasifikaci síťového provozu pomocí kontrastivního učení pomocí metody SimCLR. Jelikož je tato metoda určená pro klasifikaci obrazových dat, síťové toky jsou převedeny do obrazové reprezentace nazývané FlowPic. Cílem je replikovat výsledky dosažené v referenční práci a rozšířit ji o další datové sady a obecně větší počet vzorků. Model je ověřen také na datové sadě z reálné sítě vytvořené sdružením CESNET.

Abstract

This bachelor's thesis focuses on network traffic classification using contrastive learning, specifically the SimCLR method. Since this method is designed for image classification tasks, network flows are transformed into an image-based representation known as FlowPic. The objective is to replicate the results achieved in the reference work and extend them by incorporating additional datasets and a generally larger number of samples. The model is also validated on a real-world dataset collected from a production network operated by the CESNET association.

Klíčová slova

klasifikace, síťový provoz, neuronové sítě, kontrastivní učení, SimCLR, self-supervised learning, Python, PyTorch, FlowMind, FlowPic

Keywords

classification, network traffic, neural networks, contrastive learning, SimCLR, self-supervised learning, Python, PyTorch, FlowMind, FlowPic

Citace

ČOČEK, Jakub. *Klasifikace síťového provozu pomocí contrastive learning*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Kamil Jeřábek, Ph.D.

Klasifikace síťového provozu pomocí contrastive learning

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Kamila Jeřábka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Čoček
11. května 2025

Poděkování

Tímto bych rád poděkoval vedoucímu mé práce, Ing. Kamilu Jeřábkovi, Ph.D., za odborné vedení, cenné rady a velkou podporu při řešení této práce.

Obsah

1	Úvod	5
2	Klasifikace síťového provozu	6
2.1	Toky síťového provozu	6
2.1.1	Architektura monitorování toků	6
2.1.2	NetFlow	7
2.1.3	IPFIX	7
2.2	Metody klasifikace síťového provozu	8
2.2.1	Port-based klasifikace	8
2.2.2	Payload-based klasifikace	8
2.2.3	Klasifikace pomocí strojového učení	8
2.3	Klasifikace šifrovaného síťového provozu	9
2.3.1	Feature-based metody	9
2.3.2	FlowPic	9
2.3.3	Mini FlowPics	10
3	Neuronové sítě	11
3.1	Různé typy neuronových sítí	11
3.1.1	Dopředné neuronové sítě	11
3.1.2	Konvoluční neuronové sítě	12
3.1.3	Rekurentní neuronové sítě	12
3.2	Struktura a fungování neuronových sítí	13
3.2.1	Základní pojmy	13
3.3	Trénování neuronových sítí	14
3.3.1	Backpropagation	15
3.4	Supervised learning	15
3.4.1	Self-supervised learning	15
3.4.2	Kontrastivní učení	16
3.4.3	SimCLR	16
3.4.4	Ztrátová funkce	17
3.5	Augmentace dat	17
3.5.1	Modifikace RTT	18
3.5.2	Modifikace IAT	18
3.5.3	Vynechávání paketů	19
3.5.4	FlowPic augmentace	19
4	Návrh řešení	21
4.1	Datové sady	21

4.1.1	Mirage19	21
4.1.2	Mirage22	22
4.1.3	Ucdavis-icdm19	24
4.1.4	UTMobileNetTraffic2021	25
4.1.5	CESNET	26
4.2	Model	27
4.2.1	Příprava dat	27
4.2.2	Extrakce reprezentací	27
4.2.3	Lineární klasifikátor	28
4.3	Návrh experimentů	28
5	Implementace a vyhodnocení dosažených výsledků	30
5.1	Použité knihovny	30
5.1.1	PyTorch	30
5.1.2	FlowMind	32
5.2	Postup trénování	32
5.3	Dosažené výsledky	33
5.3.1	Mirage19 a Mirage22	34
5.3.2	Ucdavis-icdm19	35
5.3.3	UTMobileNetTraffic2021	36
5.3.4	CESNET	37
5.3.5	FlowPic 64×64	38
6	Závěr	41
Literatura		42

Seznam obrázků

2.1	Architektura typické sestavy pro monitorování toků [14].	7
2.2	Příklady FlowPics pro několik video aplikací (pro ilustrační účely představují černé pixely libovolnou hodnotu v rozmezí 1 až 255, bílé pixely představují hodnotu 0) [28].	10
3.1	Základní architektura dopředných neuronových sítí [25].	12
3.2	Základní architektura konvolučních neuronových sítí [9].	12
3.3	Základní intuice paradigmatu kontrastního učení, přiblížit původní a augmentované obrázky a oddálit původní a negativní [16].	16
3.4	Ilustrace tréninkového procesu pomocí metody SimCLR, kde x je vstupní vzorek, \mathcal{T} je rodina validních augmentací, t je vybraná transformace, \tilde{x}_i a \tilde{x}_j jsou augmentované pohledy, $f(\cdot)$ je základní neuronová síť (<i>encoder</i>), h_i a h_j jsou reprezentace z augmentovaných vzorků, $g(\cdot)$ je projekční hlava, která mapuje reprezentace do latentního prostoru a z_i , z_j jsou latentní reprezentace [4].	17
3.5	Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).	18
3.6	Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).	19
3.7	Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).	19
3.8	Rozdíl mezi originální FlowPic reprezentací (vlevo) a rotovanou verzí (vpravo).	20
4.1	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage19 bez filtru.	22
4.2	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage19 s filtrem.	22
4.3	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage22 s filtrem $>10\text{pkts}$	23
4.4	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage22 s filtrem $>1\,000\text{pkts}$	23
4.5	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Ucdavis-icdm19.	24
4.6	Porovnání FlowPic reprezentací vytvořených z části <i>script</i> (vlevo) a <i>human</i> (vpravo).	24
4.7	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady UTMobileNetTraffic2021 bez filtru.	25
4.8	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady UTMobileNetTraffic2021 s filtrem.	26

4.9	FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady CESNET.	26
4.10	LeNet-5 architektura [19].	27
4.11	Ilustrace architektury pro 32×32 Mini FlowPic.	28
4.12	Ilustrace procesu trénování. Sněhová vločka značí zmražení <i>encoderu</i>	29

Kapitola 1

Úvod

S rostoucí digitalizací a závislostí společnosti na sítové infrastrukturu se problematika efektivní klasifikace sítového provozu stává stále důležitější. Sítový provoz zahrnuje různé typy datových přenosů, jejichž přesné rozlišení hraje klíčovou roli v oblastech, jako je správa sítí, zajištění bezpečnosti, identifikace anomalií či optimalizace výkonu. Tradiční metody klasifikace, jako jsou *port-based* nebo *payload-based* přístupy, narážejí na limity způsobené šifrováním provozu a rostoucí složitostí sítových aplikací. Proto se stále více prosazuje využití pokročilých technik strojového učení, které dokážou zpracovávat složitější datové struktury a přizpůsobovat se dynamickým podmínkám. Zejména metody hlubokého učení nabízejí možnost extrahat komplexní charakteristiky sítového provozu, které tradiční algoritmy nedokážou efektivně identifikovat. Tyto techniky umožňují analyzovat šifrovaný provoz na základě vnějších vlastností, díky čemuž se strojové učení stává klíčovým nástrojem pro moderní analýzu a správu sítí.

Tato práce se zaměřuje na aplikaci kontrastivního učení (*contrastive learning*), což je moderní přístup v oblasti *self-supervised learning*, který umožňuje efektivní učení reprezentací dat bez nutnosti rozsáhlého ručního označování. Hlavní myšlenkou této metody je využití pozitivních a negativních párů vzorků k naučení modelu rozlišovat mezi podobnými a odlišnými charakteristikami dat. V kontextu klasifikace sítového provozu kontrastivní učení nabízí možnost lepšího porozumění datovým tokům a zlepšení přesnosti klasifikace zejména u šifrovaného provozu.

V rámci kontrastivního učení se v této práci využívá framework SimCLR (*A Simple Framework for Contrastive Learning of Visual Representations*), který patří mezi moderní a vysoce efektivní přístupy v oblasti *self-supervised learning*. Přestože je tento framework primárně navržený pro visuální reprezentace, lze ho využít i pro sítový provoz.

V této práci toho bude docíleno pomocí metody FlowPic, respektive její menší varianty Mini FlowPic. Tento přístup převádí základní data o sítových tokích do vizuální reprezentace v podobě histogramu, který se označuje jako FlowPic. Díky tomu je možné využít výhod frameworku SimCLR a efektivně aplikovat kontrastivní učení.

Dále si tato práce klade za úkol experimentálně ověřit efektivitu vybraných augmentací sítových toků na různých datových sadách zachycující jak uměle vygenerovaný provoz, tak reálný provoz generovaný lidmi.

Kapitola 2

Klasifikace síťového provozu

Na provoz v datové síti lze pohlížet jako na tok procházející síťovými prvky. Pro administrativní nebo jiné účely je často zajímavé, užitečné nebo dokonce nezbytné mít přístup k informacím o těchto tocích [2]. Klasifikace provozu je prvním krokem k identifikaci a klasifikaci neznámých tříd síťového provozu. Klasifikace síťového provozu hraje klíčovou roli v bezpečnosti a správě sítě, jako je detekce průniků, detekci anomalií, optimalizaci výkonu, ochranu proti škodlivým aktivitám nebo zajištění kvality služby (QoS) [26]. Díky této technice mohou např. správci sítí přijímat opatření, jako je blokování některých toků, řízení zdrojů či monitorování síťových aplikací.

2.1 Toky síťového provozu

V dnešních vysokorychlostních sítích s přenosovou rychlosťí až 100 Gbps vyžaduje pasivní monitorování, jako je zachytávání paketů, nákladný hardware a rozsáhlou infrastrukturu pro ukládání a analýzu dat. Efektivnějším a škálovatelnějším přístupem k pasivnímu monitorování ve vysokorychlostních sítích je export toků, kdy jsou pakety agregovány do toků a následně exportovány k uložení a analýze [14].

RFC 7011 definuje tok jako soubor IP paketů, které procházejí pozorovacím bodem v síti během určitého časového intervalu, přičemž všechny pakety v rámci daného toku sdílejí určité společné vlastnosti [2].

V datasetech využívaných v této práci jsou ony společné vlastnosti ID toku, zdrojová a cílová IP adresa, zdrojový a cílový port, protokol, čas prvního a posledního paketu, počet paketů, jejich směr, velikost a značka (*label*).

2.1.1 Architektura monitorování toků

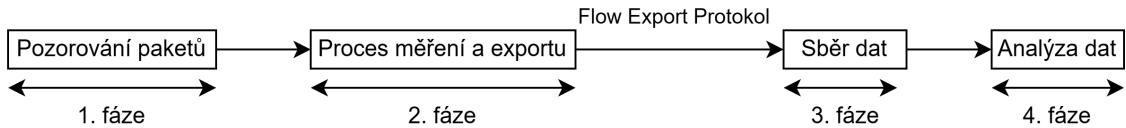
Architektura typických systémů pro monitorování toků se skládá z několika fází.

První fáze, nazvaná pozorování paketů, zahrnuje zachycení paketů z pozorovacího bodu a jejich následné předzpracování. Pozorovacími body mohou být například rozhraní síťových zařízení [14].

Druhá fáze se skládá z procesu měření a procesu exportu. V rámci procesu měření jsou pakety seskupovány do toků. Jakmile je tok považován za ukončený, je záznam o toku exportován prostřednictvím procesu exportu, přičemž záznam je vložen do datagramu podle protokolu pro export toků [14].

Třetí fáze je nazvaná sběr dat. Jejím hlavním úkolem je přijímání, ukládání a předzpracování dat toků generovaných v předchozí fázi. Mezi běžné operace předzpracování patří agregace, filtrování, komprese dat a vytváření souhrnů [14].

Poslední fází je analýza dat. V rámci výzkumných nasazení má analýza dat často průzkumný charakter (tj. manuální analýza), zatímco v operačních prostředích jsou analytické funkce obvykle integrovány do fáze sběru dat (tj. kombinace manuální a automatizované analýzy). Mezi běžné analytické činnosti patří korelace a agregace, profilování, klasifikace a charakterizace provozu, detekce anomalií a identifikace průniků [14].



Obrázek 2.1: Architektura typické sestavy pro monitorování toků [14].

Dva hlavní protokoly pro export toků jsou NetFlow [5] a Internet Protocol Flow Information Export (IPFIX) [2].

2.1.2 NetFlow

V devadesátých letech dvacátého století začala společnost Cisco pracovat na vlastní technologii exportu toků, kterou pojmenovala NetFlow. Tuto technologii firma patentovala v roce 1996. Zpočátku se jednalo o proprietární protokol, který byl kompatibilní pouze se zařízeními Cisco. První verze, která získala široké rozšíření, byla NetFlow v5, která byla veřejnosti zpřístupněna kolem roku 2002 a popsána v RFC 3954 [5]. Nejnovější verzí NetFlow je NetFlow v9, která byla představena v roce 2004.

2.1.3 IPFIX

V devadesátých letech dvacátého století standardizovala *Internet Engineering Task Force* (IETF) protokol *Realtime Traffic Flow Management* (RTFM), sloužící k měření síťových toků na základě SNMP¹. V roce 2001 zahájila IETF iniciativu zaměřenou na vytvoření interoperabilního protokolu pro export síťových toků v rámci pracovní skupiny IPFIX. Výsledkem tohoto úsilí byla definice požadavků (RFC 3917 [33]) a následné zhodnocení kandidátních protokolů (RFC 3955 [20]) vhodných pro měření síťových toků, jak bylo detailně popsáno v prohlášení o použitelnosti IPFIX (RFC 5472 [32]). Pracovní skupina, s cílem vyházet při návrhu budoucího standardu co nejvíce z existujících de facto standardů, zvolila jako základ protokol Cisco NetFlow v9 [5]. Tento protokol, nástupce široce používaného NetFlow v5, byl vytvořen s ohledem na specifické požadavky stanovené IPFIX [30].

IPFIX definuje architekturu (RFC 5470 [24]), která představuje zobecnění existujících architektur používaných v měřicích infrastrukturách. Tato architektura poskytuje jednotnou terminologii a formuluje předpoklady na architektonické úrovni, které protokol předpokládá.

Architektura rozlišuje tři základní typy procesů: měřicí procesy (MPs), jež generují toku na základě pozorovaných paketů, exportní procesy (EPs), které zajišťují přenos toků prostřednictvím protokolu IPFIX do sběrných procesů (CPs). Všechny vztahy mezi těmito procesy mají charakter jednoho vůči mnoha, například exportní proces může exportovat

¹<https://doi.org/10.17487/RFC1157>

toky z více měřicích procesů do několika sběrných procesů, přičemž každý sběrný proces může přijímat IPFIX zprávy od více exportních procesů a naopak.

Snahy o škálování IPFIX pro rozsáhlá a vícevrstvá nasazení vedly k zavedení intermediárních procesů (ImPs), které provádějí úpravy toků, jako je jejich agregace, korelace či anonymizace. Zařízení, které kombinuje sběrný proces, jeden nebo více intermediárních procesů a exportní proces, se označuje jako mediátor [30].

2.2 Metody klasifikace síťového provozu

Klasifikace síťového provozu je proces identifikace síťových aplikací nebo protokolů, které se vyskytují v síti. Existuje mnoho různých metod pro klasifikaci síťových aplikací, nicméně tato práce se zaměří na tři hlavní přístupy: klasifikace založená na portech (tzv. *Port-based Classification*), klasifikace založená na datech přenášených v paketech (tzv. *Payload-Based Classification*) a nakonec klasifikace pomocí strojového učení (ML) [26].

2.2.1 Port-based klasifikace

Klasifikace síťových aplikací se provádí pomocí tzv. *well-known* čísel portů. Provoz je identifikován na základě portů registrovaných u *Internet Assigned Numbers Authority* (IANA²). Například e-mailové aplikace používají port 25 (SMTP³) pro odesílání e-mailů a port 110 (POP3⁴) pro jejich přijímání. Dalším známým příkladem jsou webové aplikace, které využívají port 80. Tento přístup však neposkytuje příliš přesné výsledky klasifikace [21]. Tato technika navíc selhává kvůli používání dynamických čísel portů nových aplikací, aby se vyhnula odhalení [26].

2.2.2 Payload-based klasifikace

Tato metoda, známá jako technika hluboké analýzy paketů (*Deep Packet Inspection*, DPI), spočívá v prozkoumání obsahu paketů a hledání charakteristických signatur síťových aplikací v síťovém provozu. Je to první alternativa k metodě založené na portech zmíněné v 2.2.1. Tato technika je zvláště určena pro aplikace typu *Peer to Peer* (P2P), které používají dynamická čísla portů [17].

Tato technika má však několik nedostatků. Prvním problémem je, že vyžaduje nákladný hardware pro vyhledávání vzorců v datech paketů. Druhým problémem je, že ji nelze použít u šifrovaného síťového provozu. Kromě toho tento přístup vyžaduje neustálé aktualizace signaturních vzorců pro nové aplikace [26].

2.2.3 Klasifikace pomocí strojového učení

Technika strojového učení (ML) spočívá v analýze a učení vzorců z již zachyceného síťového provozu. V rámci této metody se klasifikátor strojového učení trénuje na základě tréninkových vzorků, a poté se pomocí tohoto modelu klasifikují neznámé třídy [26]. V této práci se právě tato metoda klasifikace síťového provozu využívá. Konkrétně se jedná o kontrastní učení, které je popsáno v sekci 3.4.2.

²<https://www.iana.org/>

³<https://doi.org/10.17487/RFC5321>

⁴<https://doi.org/10.17487/RFC1939>

2.3 Klasifikace šifrovaného síťového provozu

S rychlým růstem objemu internetového provozu se bezpečnost síťových spojení stává klíčovým aspektem, jelikož prostřednictvím internetu dochází k přenosu velkého množství citlivých uživatelských informací, jako jsou například bankovní údaje a záznamy o platbách. Pro zajištění důvěrnosti a integrity dat se široce implementují moderní šifrovací technologie jako je *Secure Socket Layer/Transport Layer Security (SSL/TLS)*⁵, které poskytují nezbytnou ochranu síťových spojení. Avšak neustálý nárůst šifrovaného provozu přináší nové výzvy pro analýzu síťového provozu. Tradiční metody, které se při analýze spoléhaly na přístup k obsahu nešifrovaných paketů (*plaintext payload*), v případě šifrovaného provozu ztrácejí svou efektivitu, neboť tento obsah již není dostupný [29].

2.3.1 Feature-based metody

Jedním z přístupů ke klasifikaci šifrovaného síťového provozu jsou metody založené na vlastnostech síťového provozu (*feature-based*). Tyto metody se nezaměřují na obsah datových částí paketů, ale na vnější charakteristiky síťového provozu, jako jsou velikost paketů, časové intervaly mezi jejich přenosy, počet paketů v obou směrech komunikace a doba trvání síťového spojení. Tyto charakteristiky zůstávají do určité míry stabilní i v případě šifrované komunikace, což umožňuje efektivní analýzu a klasifikaci provozu [31].

Některé dodatečné informace je možné získat také během inicializační fáze šifrovaných protokolů. Typickým příkladem je *TLS handshake*, který často obsahuje nešifrované části umožňující analýzu. Tato fáze může poskytnout cenné informace o šifrovacích algoritmech a dalších parametrech použitého protokolu, což je užitečné pro identifikaci typu šifrované komunikace.

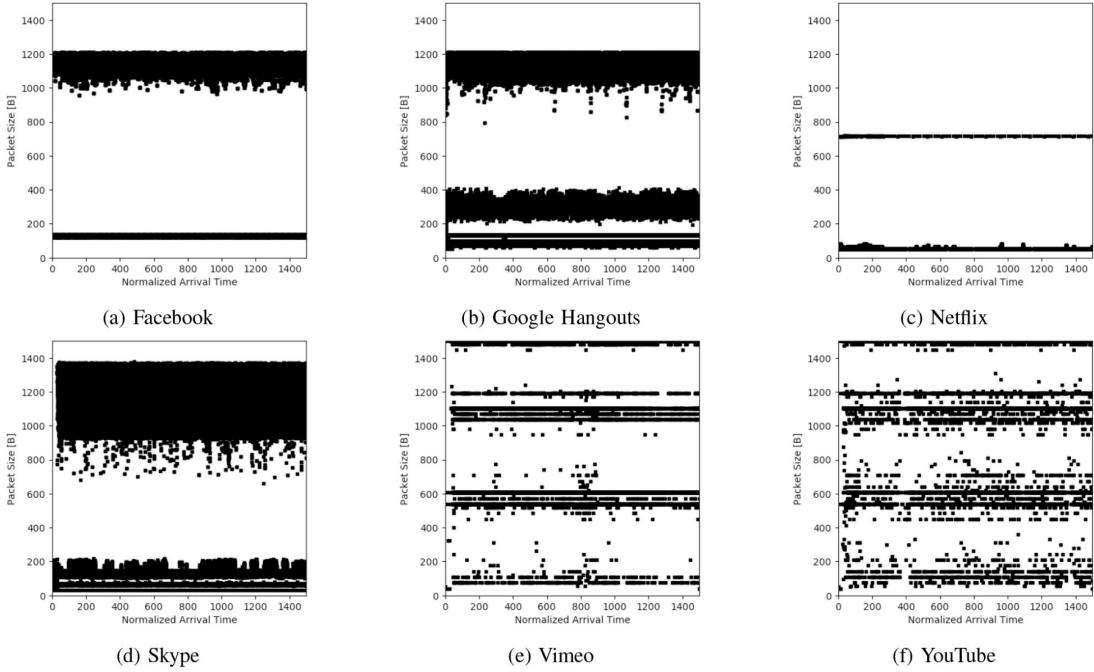
2.3.2 FlowPic

Další metodou pro klasifikaci šifrovaného internetového provozu a identifikaci aplikací je FlowPic. Tento přístup převádí základní data o síťových tocích, které jsou popsány v sekci 2.1, do vizuální reprezentace v podobě obrazu, označovaného jako FlowPic, a následně aplikuje pokročilé techniky hlubokého učení, konkrétně konvoluční neuronové sítě (CNNs), které jsou popsány v sekci 3.1.2, k určení kategorie síťového toku (např. prohlížení webu, chat, streamování videa) a identifikace konkrétní aplikace, která daný tok generuje [28].

FlowPic vytváří obraz založený na dvourozměrném histogramu síťového toku. Tento obraz lze chápat jako pole distribucí velikostí datových částí paketů (*Payload Size Distributions*, PSD), kde každá PSD odpovídá specifickému časovému intervalu v rámci jednosměrného síťového toku. Vzhledem k tomu, že naprostá většina velikostí paketů nepřesahuje 1 500 bajtů (hodnota maximální přenosové jednotky (MTU) pro Ethernet), pakety s velikostí větší než 1 500 bajtů jsou ignorovány a osa Y je omezena na rozsah 1 až 1 500 bajtů [27].

Následně jsou všechny normalizované páry vloženy do dvourozměrného histogramu, kde každá buňka obsahuje počet paketů, které dorazily v daném časovém intervalu a mají odpovídající velikost. Každý histogram o rozměrech $1\ 500 \times 1\ 500$ je uložen jako maticová reprezentace obrazu, která se nazývá FlowPic [27].

⁵<https://doi.org/10.17487/RFC8446>



Obrázek 2.2: Příklady FlowPics pro několik video aplikací (pro ilustrační účely představují černé pixely libovolnou hodnotu v rozmezí 1 až 255, bílé pixely představují hodnotu 0) [28].

2.3.3 Mini FlowPics

Jak je uvedeno výše v sekci 2.3.2, původní FlowPic obrázky mají rozlišení 1500×1500 pixelů. Hodnota 1500 na ose Y je odůvodněna schopností reprezentovat velikost každého paketu mezi 1 a hodnotou MTU, zatímco hodnota 1500 na ose X umožňuje vytvoření jednoduchého čtvercového obrázku.

Mini FlowPic využívají stejný konstrukční princip jako FlowPic, ale používají výrazně menší obrázky. Rozměr na ose Y je zmenšen generováním histogramů, které počítají počet paketů v určitých rozsazích délek. Rozměr na ose X je zmenšen rozdělením časového intervalu na menší počet časových slotů. Mini FlowPics jsou levnější na konstrukci, trénování a predikci, což představuje významnou výhodu [27].

Práce od Shapira *et al.* [28] testovala časové bloky o délce 120, 60, 30 a 15 sekund a zjistila, že se rozdíl v průměrné přesnosti zvedl o 1,25 %.

Kapitola 3

Neuronové sítě

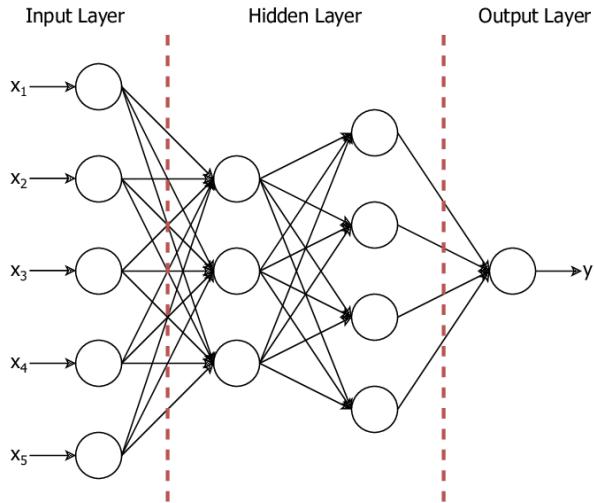
Neuronové sítě jsou komplexní a adaptivní statistické modely, které se inspirovaly strukturou lidského mozku. Jsou součástí širšího rámce umělé inteligence a strojového učení, přičemž jejich hlavní charakteristikou je schopnost učit se a adaptovat na základě zkušeností. Tento proces učení se podobá způsobu, jakým mozek zpracovává informace a vyvozuje závěry na základě vjemů a předchozích zkušeností [13]. Neuronových sítí je zároveň více druhů, v této práci bude využívána dopředná neuronová síť, neboli *feedforward neural network* (FFNN) a také konvoluční neuronová síť (CNN).

3.1 Různé typy neuronových sítí

Neuronové sítě lze rozdělit do několika typů, z nichž každá je vhodná pro různé úkoly. Mezi základní patří dopředné neuronové sítě (FFNN), které tvoří nejjednodušší architekturu a slouží jako výchozí model pro řadu klasifikačních a regresních úloh, konvoluční neuronové sítě (CNN), které se uplatňují zejména při analýze obrazových dat a rekurentní neuronové sítě, neboli *recurrent neural networks* (RNN), které jsou vhodné pro práci s časovými řadami nebo sekvenčními daty.

3.1.1 Dopředné neuronové sítě

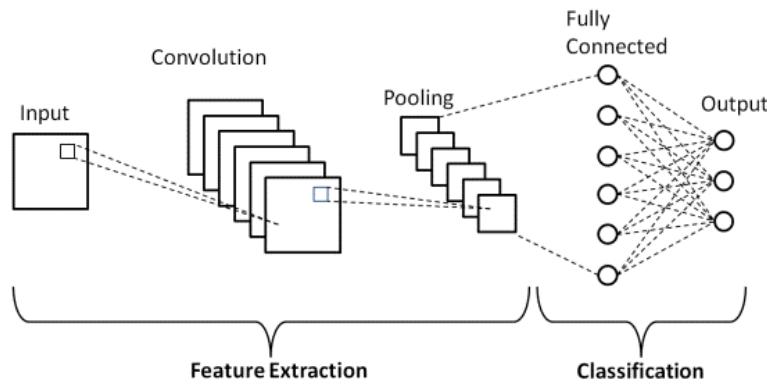
Dopředné neuronové sítě, někdy označovány jako vícevrstvé perceptrony, neboli *Multilayer perceptrons* (MLP), jsou klíčovými modely hlubokého učení. Tyto modely se nazývají dopředné, protože informace proudí skrze funkci, která je vyhodnocována od vstupu x , přes mezi-výpočty potřebné k definici funkce f , až po výstup y . Neexistují zde žádné zpětné vazby, kdy by výstupy modelu byly zpětně vráceny do něho samotného. Tento typ sítě má typicky tři druhy vrstev, a to první vrstvu, která se označuje jako vstupní, pak jednu nebo více skrytých vrstev, a poslední vrstvu, která se označuje jako výstupní. Tyto sítě mají zásadní význam pro odborníky na strojové učení. Tvoří základ mnoha důležitých komerčních aplikací, například konvoluční sítě používané pro rozpoznávání objektů na fotografiích jsou specializovaným druhem dopředných sítí, jsou také konceptuálním krokem na cestě k rekurentním sítím, které pohánějí mnoho aplikací z oblasti zpracování přirozeného jazyka [10].



Obrázek 3.1: Základní architektura dopředných neuronových sítí [25].

3.1.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou specializovaný typ neuronových sítí určený pro zpracování dat, která mají známou mřížkovou topologii. Příklady zahrnují časové řady, které lze považovat za jednorozměrnou mřížku odebírající vzorky v pravidelných časových intervalech, a obrazová data, která lze považovat za dvourozměrnou mřížku pixelů. Konvoluční sítě dosáhly obrovského úspěchu v praktických aplikacích. Její název naznačuje, že síť používá matematickou operaci nazvanou konvoluce. Konvoluční sítě jsou jednoduše neuronové sítě, které používají konvoluci místo obvyklého maticového násobení v alespoň jedné ze svých vrstev [10].



Obrázek 3.2: Základní architektura konvolučních neuronových sítí [9].

3.1.3 Rekurentní neuronové sítě

Rekurentní neuronové sítě jsou rodinou neuronových sítí určených pro zpracování sekvenčních dat $x^{(1)}, \dots, x^{\tau}$. Rekurentní sítě dokážou efektivně zpracovávat mnohem delší sekvence, než by bylo možné u sítí, které na to nejsou specializovány. Většina rekurentních sítí dokáže zpracovávat sekvence proměnlivé délky. Sdílení parametrů umožňuje rozšířit a aplikovat

model na příklady různého typu (v tomto případě různé délky sekvencí) a zobecnit je. Kdybychom měli samostatné parametry pro každou hodnotu časového indexu, nemohli bychom generalizovat na sekvence, jejichž délka nebyla během trénování viděna, ani sdílet statistickou sílu napříč různými délkami sekvencí a pozicemi v čase. Takové sdílení je obzvláště důležité, když se může konkrétní informace objevit na více pozicích v rámci sekvence [10].

3.2 Struktura a fungování neuronových sítí

Neuronová síť se skládá ze vstupní vrstvy, kde se přijímají vstupy problému, skrytých vrstev, kde se určuje vztah mezi vstupy a výstupy, a kde jsou reprezentovány synaptickými váhami, a výstupní vrstvy, která vysílá výstupy problému. Jednotlivé vrstvy se skládají z několika uzlů, kterým se říká neurony, jež jsou mezi jednotlivými vrstvami propojené váženými spoji. Výstup každého uzlu závisí pouze na informacích, které jsou v uzlu lokálně k dispozici, ať už jsou uloženy interně, nebo přicházejí prostřednictvím vážených spojení. Každá jednotka přijímá vstupy z mnoha jiných uzlů předává svůj výstup dalším uzlům [7]. Na konci je spočítána chyba (tzv. *loss funkce*), které je definována jako rozdíl mezi požadovaným výstupem a výstupem systému. Tato informace o chybě se vrací zpět do systému a systematicky upravuje parametry. Tento proces se opakuje, dokud není výkonnost přijetelná.

3.2.1 Základní pojmy

Neuronové sítě představují komplexní systémy, které se skládají z několika klíčových komponent. Pro usnadnění orientace v této práci je nezbytné definovat základní pojmy, které se v souvislosti s neuronovými sítěmi často používají.

Neuron

Základní stavební jednotka neuronové sítě, inspirovaná biologickými neurony. Ve své podstatě je neuron optimalizován tak, aby přijímal informace od jiných neuronů, zpracovával je jedinečným způsobem a posílal jejich výsledek dalším buňkám. Stejně jako u biologických neuronů, umělý neuron přijímá určitý počet vstupů x_1, x_2, \dots, x_n , z nichž každý je vynásoben určitou váhou w_1, w_2, \dots, w_n . Tyto vážené vstupy se sečtou dohromady, aby vznikl *logit* neuronu $z = \sum_{i=0}^n w_i x_i$. Ve většině případech je součástí *logitu* také *bias* [3].

Vrstva

Neurony v lidském mozku jsou uspořádány do vrstev. Informace proudí z jedné vrstvy do druhé, dokud se smyslový vjem nepromění v pojmové porozumění. Podobně vzniká neuronová síť, a to když začneme propojovat neurony mezi sebou v různých vrstvách. První vrstva je označována jako vstupní a její úkol je načíst vstupní data. Poslední vrstva neuronů (výstupní vrstva) vypočítává konečnou odpověď. Prostřední vrstva (vrstvy) neuronů se nazývají skryté vrstvy, přičemž $w^{(k)}$ je váha spojení mezi neuronem ve vrstvě k s neuronem ve vrstvě $k + 1$. Neurony ve stejné vrstvě mezi sebou nejsou propojeny. Skryté vrstvy mají často méně neuronů než vstupní vrstva, aby se síť naučila komprimovanou reprezentaci původního vstupu, který je zároveň i s výstupy vektorizovanou reprezentací [3].

Váha

Váhy jsou parametry modelu a určují sílu spojení mezi neurony. Každá váha w řídí, jak velký vliv má jeden neuron na další. Během trénování jsou tyto váhy upravovány tak, aby minimalizovaly chybu mezi výstupem sítě a požadovaným výstupem [10].

Bias

Bias (někdy označován také jako posun) lze chápat jako klíčový parametr, který ve spojení s váhami ovlivňuje výstup každého uzlu [22]. Bias umožňuje neuronům posunout aktivaci výstupu, což znamená, že neuron může aktivovat (nebo neaktivovat) svůj výstup i tehdy, když jsou všechny vstupy nulové nebo když mají určité hodnoty, které by jinak vedly k neaktivnímu stavu. Podobně jako váha je také bias při trénování upravován, aby se minimalizovala celková chyba mezi výstupem sítě a požadovaným výstupem.

Aktivační funkce

Aktivační funkce jsou klíčovým prvkem, který umožňuje neuronům vykazovat nelineární chování, což je zásadní pro schopnost neuronových sítí modelovat složité vzory a vztahy mezi daty. Jsou tři základní typy, které se v praxi využívají. Podle typu aktivační funkce lze pojmenovávat i samotné neurony. První z nich je *sigmoid neuron*, který používá funkci:

$$f(z) = \frac{1}{1 + e^{-z}}.$$

V praxi to znamená, že když je logit velmi malý, výstup neuronu je velmi blízko 0, a když je velmi velký, výstup je blízko 1 [3].

Dalším z používaných je *tanh neuron*, který využívá podobný druh nelinearity jako *sigmoid neuron*, ale namísto rozsahu 0 až 1 se výstup *tanh neuronů* pohybuje v rozmezí -1 až 1 [3]. Funkce, kterou neuron používá je, jak již název napovídá, následující:

$$f(z) = \tanh(z).$$

Poslední typ, který poslední dobou získává více a více popularity a je hojně využívaný, je *restricted linear unit (ReLU) neuron*, který závádí odlišný typ nelinearity než předchozí dvě aktivační funkce [3]. V praxi to znamená, že když je logit záporný, výstup je nulový, a když je logit kladný, jeho hodnota se nemění. Matematicky lze funkci vyjádřit následovně:

$$f(z) = \max(0, z).$$

3.3 Trénování neuronových sítí

Trénink modelu znamená iterativní úpravu jeho parametrů, jako jsou váhy a biasy, s cílem minimalizovat rozdíl mezi výstupy modelu a očekávanými výsledky. Tento rozdíl je měřen pomocí ztrátové funkce, která kvantifikuje výkon modelu a slouží k optimalizaci. Fáze trénování zahrnuje dva hlavní kroky: dopředný a zpětný průchod sítí. Nejprve jsou vstupní data poslána dopředu, kde model na základě svých aktuálních parametrů vytvoří předpovědi. Následně se spočítá chyba mezi těmito předpověďmi a skutečnými výstupy. Tato chyba se následně využije ve zpětném průchodu, kde se pomocí výpočtu gradientů ztrátové funkce aktualizují parametry modelu. Tento proces se opakuje v mnoha iteracích, dokud model nenajde optimální hodnoty parametrů, které minimalizují chybu a zajišťují

co nejlepší výkon [3]. Jedním z nejvíce používaných algoritmů je zpětné propagace, neboli *backpropagation*.

3.3.1 Backpropagation

Pokud používáme neuronovou síť typu feedforward, která přijímá vstup x a generuje výstup \hat{y} , informace procházejí sítí směrem vpřed. Vstup x poskytuje počáteční data, která se postupně šíří skrz skryté jednotky v jednotlivých vrstvách a nakonec vedou k výstupu \hat{y} . Tento proces je známý jako dopředná propagace, neboli *forward propagation* [10].

Během trénování může dopředná propagace pokračovat, dokud nevytvoří skalární hodnotu ztrátové funkce $J(\theta)$. Algoritmus zpětné propagace, často jednoduše nazývaný *back-prop*, umožňuje, aby informace o ztrátě následně proudily zpětně sítí, aby bylo možné vypočítat [10].

Termín zpětná propagace je často mylně chápán jako celý učící algoritmus pro vícevrstvé neuronové sítě. Ve skutečnosti zpětná propagace označuje pouze metodu pro výpočet gradientu, zatímco jiný algoritmus, například stochastický gradientní sestup, je použit k provádění učení na základě tohoto gradientu. Dále je zpětná propagace často mylně považována za specifickou pro vícevrstvé neuronové sítě, ale v principu může vypočítat derivace jakékoli funkce (u některých funkcí je správnou odpovědí, že derivace funkce je nedefinovaná) [10].

3.4 Supervised learning

Učení s učitelem, neboli *supervised learning*, představuje velkou část výzkumné činnosti v oblasti strojového učení. Definující charakteristikou učení s učitelem je dostupnost anotovaných trénovacích dat, na rozdíl od učení bez dohledu, neboli *unsupervised learning*, kdy algoritmus musí odvodit vzory a struktury pouze z neoznačených dat. Název naznačuje představu „učitele“, který instruuje učící se systém, jaké štítky přiřadit k trénovacím příkladům. Tyto štítky jsou obvykle třídy v úlohách klasifikace. Učení spočívá v naučení se mapování mezi sadou vstupních proměnných X a výstupní proměnnou Y a následném použití tohoto mapování k predikci výstupu pro dosud neviděná data [6].

Tradiční přístupy učení s učitelem jsou silně závislé na množství dostupných anotovaných trénovacích dat. I když je k dispozici obrovské množství dat, nedostatek anotací vedl výzkumníky k hledání alternativních přístupů, které je dokázou využít. V tomto kontextu hrají *self-supervised* metody klíčovou roli v podpoře pokroku hlubokého učení bez nutnosti nákladných anotací [16].

3.4.1 Self-supervised learning

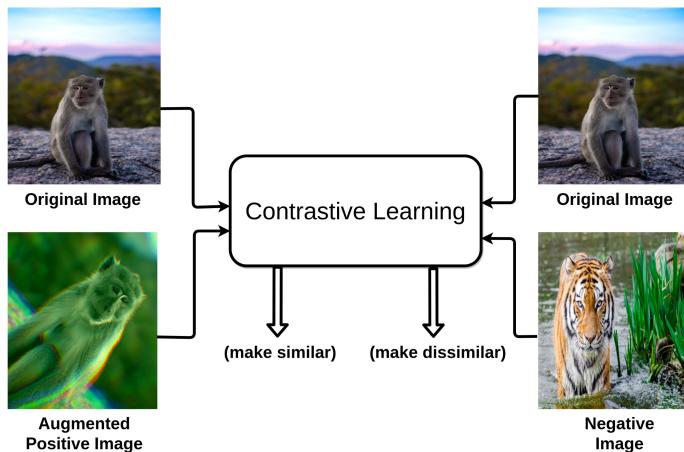
Self-supervised learning (SSL) představuje způsob, jak využít neoznačená data k učení základních reprezentací, čímž se eliminuje závislost na ručně anotovaných trénovacích datech. Místo explicitních anotací si model generuje vlastní pseudoznačky, což umožňuje učení přímo z dat bez nutnosti lidského zásahu.

V poslední době se v metodách SSL kombinují jak generativní, tak kontrastivní přístupy. Generativní metody se zaměřují na modelování dat tak, aby model mohl generovat nebo rekonstruovat jejich části, zatímco kontrastivní metody se soustředí na učení reprezentací tím, že rozlišují mezi podobnými a odlišnými vzory. Mezi oblíbené techniky patří navrhování různých pretextových úloh, které pomáhají modelům učit se reprezentace dat pomocí pseudoznaček. Příklady těchto pretextových úloh zahrnují malování obrázků, kolorování čer-

nobílých obrázků, skládání obrázků z rozbitých částí (tzv. *jigsaw puzzles*), superrozlišení, predikci následujících snímků ve videu nebo audiovizuální korespondence. Tyto úlohy se ukázaly jako efektivní pro učení silných reprezentací, které lze následně využít pro širokou škálu *downstream* úkolů, jako je klasifikace nebo detekce objektů [16].

3.4.2 Kontrastivní učení

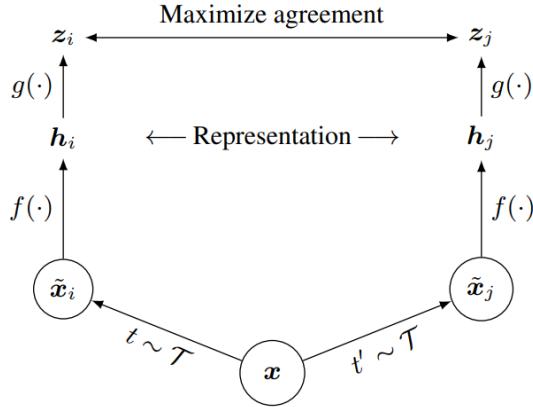
Kontrastivní učení, neboli *contrastive learning* (CL), je diskriminační přístup, který se zaměřuje na seskupování podobných vzorků blíže k sobě a odlišných vzorků dál od sebe, jak je znázorněno na obrázku 3.3. K dosažení tohoto cíle se používá metrika podobnosti, která měří, jak blízko jsou dva embeddingy, neboli číselné reprezentace dat ve vektorovém prostoru. Zejména pro úkoly počítacového vidění se hodnotí kontrastní ztráta na základě reprezentací vlastností obrázků, které jsou získány z enkodéru. Například je vybrán jeden vzorek z trénovacího datasetu a jeho transformovaná verze je získána aplikací vhodných technik augmentace dat. Během trénování je augmentovaná verze původního vzorku považována za pozitivní vzorek, a zbytek vzorků v dávce (tzv. *batch*)/datasetu (záleží na použité metodě) je považován za negativní vzorky. Model je následně trénován tak, aby se naučil odlišovat pozitivní vzorky od negativních [16].



Obrázek 3.3: Základní intuice paradigmatu kontrastního učení, přiblížit původní a augmentované obrázky a oddálit původní a negativní [16].

3.4.3 SimCLR

SimCLR je framework pro kontrastivní učení vizuálních reprezentací. *SimCLR* se učí reprezentace maximalizováním shody mezi různě augmentovanými pohledy na stejný datový vzorek pomocí kontrastní ztráty v latentním prostoru. Metoda náhodně vybere N vzorků (tzv. *minibatch*) a definuje kontrastní predikční úlohu pro páry augmentovaných vzorků, které jsou z této skupiny vzorků odvozeny, což vede k $2N$ datovým bodům. Negativní vzorky nejsou explicitně vybírány. Místo toho, pokud je dán pozitivní páru, ostatní $2(N - 1)$ augmentované vzorky ve zmíněné skupině vzorků (*minibatch*) jsou považovány za negativní vzorky [4].



Obrázek 3.4: Ilustrace tréninkového procesu pomocí metody SimCLR, kde x je vstupní vzorek, \mathcal{T} je rodina validních augmentací, t je vybraná transformace, \tilde{x}_i a \tilde{x}_j jsou augmentované pohledy, $f(\cdot)$ je základní neuronová síť (*encoder*), h_i a h_j jsou reprezentace z augmentovaných vzorků, $g(\cdot)$ je projekční hlava, která mapuje reprezentace do latentního prostoru a z_i , z_j jsou latentní reprezentace [4].

3.4.4 Ztrátová funkce

Ztrátová funkce, neboli *loss function* (také označována jako *cost function*), je funkce, která měří rozdíl mezi výstupem sítě a očekávaným výsledkem. Ztrátová funkce slouží jako optimalizační kritérium během trénování modelu, přičemž usměrňuje úpravy jeho parametrů s cílem minimalizovat chybu. Volba vhodné ztrátové funkce hraje zásadní roli, jelikož výrazně ovlivňuje schopnost modelu efektivně se učit a zobecňovat získané poznatky [22].

Ztrátová funkce využívaná metodou *SimCLR*, která je definována v sekci 3.4.3, je označována jako *NT-Xent* (*Normalized Temperature-scaled Cross Entropy Loss*) a definována následovně [4]:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)},$$

kde:

- $\text{sim}(z_i, z_k)$ je kosinová podobnost mezi reprezentacemi z_i a z_j ,
- τ je teplotní parametr,
- $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ je indikátor, že funkce vyhodnocuje 1, pokud $k \neq i$.

3.5 Augmentace dat

Jednou z možností, jak zlepšit generalizaci modelu a eliminovat přeúčení (*overfitting*), je trénovat ho na větším množství dat. V praxi ho však máme jen omezené množství. U některých úloh strojového učení je celkem snadné vytvořit nová, umělá data. Tento přístup je nejjednodušší u klasifikace. Klasifikátor musí vzít složitý, vysoce dimenzionální vstup x a shrnout ho do jediné kategorie y . To znamená, že hlavním úkolem klasifikátoru je být invariantní vůči široké škále transformací. Nové páry (x, y) můžeme snadno generovat jednoduše tím, že transformujeme vstupy x v trénovacím datasetu [10].

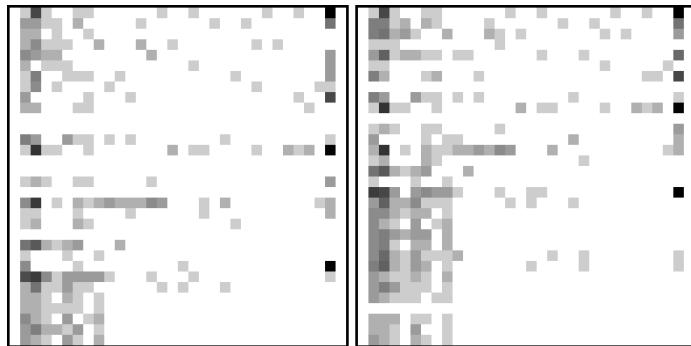
Jelikož se tato práce zabývá klasifikací síťového provozu, augmentace relevantní pro tuto práci se týkají modifikace síťových toků. Ty lze augmentovat několika způsoby, mezi které patří časový posun v paketech (*inter-arrival time*, IAT), změna obousměrného zpoždění (označováno jako *round-trip time*, RTT), změna směru toku a v neposlední řadě vynechání paketů v toku [15].

V metodě SimCLR, využívané v této práci, jsou augmentace dat klíčovou součástí procesu. Pomocí vybraných augmentací budou vytvářeny pozitivní páry, které jsou důležité pro ztrátovou funkci popsanou v sekci 3.4.4.

3.5.1 Modifikace RTT

Obousměrné zpoždění (RTT) v internetových sítích se pohybuje v rozmezí od několika milisekund až po malé zlomky sekundy. Vzhledem k mechanismům řízení přetížení v protokolu TCP a řízení toku na aplikační vrstvě se změny v RTT téměř lineárně promítají do rozložení paketů podél časové osy. Kromě těchto faktorů může změna RTT nastat také v důsledku dalších situací, častými příčinami změn jsou také přetížení síťových zařízení a ztráta paketů [15].

Pro simulaci změny RTT násobíme čas příchodu každého paketu faktorem α . Faktor α je vybíráno rovnoměrně v intervalu $[\alpha_{min}, \alpha_{max}]$, tedy $RTT_{new} = \alpha * RTT_{old}$ [15].

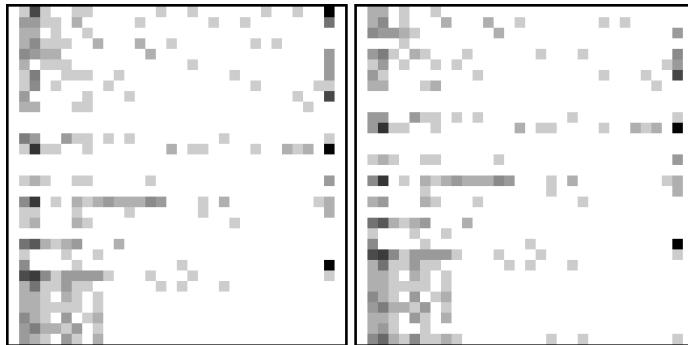


Obrázek 3.5: Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).

3.5.2 Modifikace IAT

Častými příčinami změny IAT jsou síťová přetížení a nárazové zátěže. Při síťovém přetížení může být generování paketů zpomalené, což vede k delším časovým intervalům mezi příjmy paketů, čímž dochází k prodloužení IAT. Naopak, v případě nárazových zátěží, kdy síť najednou obdrží vysoký objem dat může být čas mezi příjmy paketů zkrácen [15].

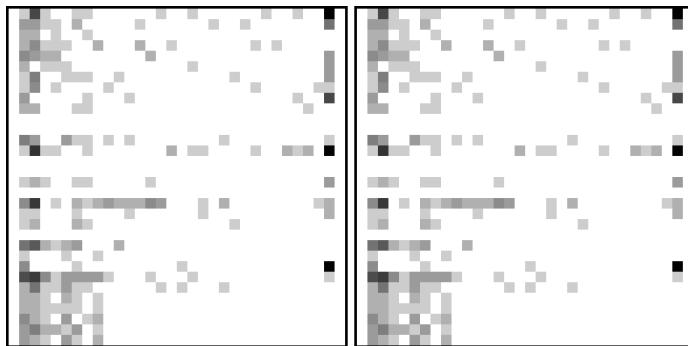
Posun v čase lze simulovat přidáním konstanty $b \in [b_{min}, b_{max}]$ k času příchodu každého paketu. Konstantní hodnota b je rovnoměrně vzorkována z intervalu $U[b_{min}, b_{max}]$, což znamená, že $t_{new} = t_{old} + b$ [4].



Obrázek 3.6: Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).

3.5.3 Vynechání paketů

Ztráty (vynechání) paketů, jsou častým jevem v síťích. Existuje několik metod simulace ztráty paketů, které mohou zohlednit komplexní chování, jako je přetížení sítových zařízení, špatná kvalita připojení nebo řízení přetížení v protokolu TCP. Simulaci lze provést na jednoduchém procesu ztráty, při němž jsou všechny pakety odstraněny v časovém intervalu $[t - \Delta t, t + \Delta t]$. Hodnota t je náhodně vybrána v rámci časového intervalu relace a Δt představuje délku tohoto intervalu [15].



Obrázek 3.7: Rozdíl mezi originální FlowPic reprezentací (vlevo) a augmentovanou verzí (vpravo).

Na první pohled vypadají obě reprezentace velmi podobně. Vzhledem k náhodné délce vynechaného intervalu může být rozdíl mezi nimi minimální. Tato změna se ve FlowPic reprezentaci projeví pouze drobnou odchylkou v intenzitě některých pixelů, což nemusí být vizuálně patrné.

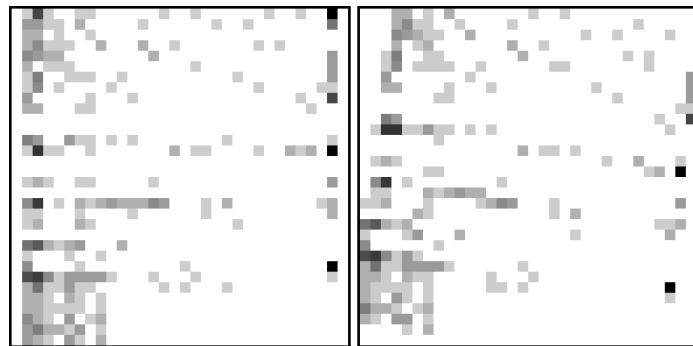
3.5.4 FlowPic augmentace

V oblasti počítačového vidění existuje široká škála známých transformací obrazů, mezi které patří například převrácení, úprava barev, ořez, rotace, přidání šumu nebo náhodné vymazání částí obrazu. Při aplikaci těchto transformací na FlowPic je však třeba zohlednit, že mohou zásadně změnit významné charakteristiky obrazu. V důsledku toho nelze jednoznačně určit užitečnost konkrétní transformace pro augmentaci pouze na základě vizuálního posouzení výsledného obrazu. Například populární transformace rotace může u FlowPicu

výrazně změnit rozložení velikosti paketů v čase, čímž dochází k deformaci významných rysů a narušení informací nezbytných pro proces učení [15].

Některé augmentace využívané v počítačovém vidění i přes to použít lze, například horizontální otočení je akceptovatelné pro periodické signály jako jsou video a VoIP, kde jsou relace dlouhé a obraz se podél horizontální osy významně nemění. Experimentovat lze také s malými rotacemi v rámci několika jednotek stupňů, jak je ukázáno na porovnání 3.8.

Přesto se ukazuje jako efektivnější aplikovat augmentace přímo na síťový tok ještě před jeho transformací do FlowPic reprezentace, protože manipulace s původním tokem umožnuje zachovat a lépe reflektovat jeho klíčové vlastnosti a statistiky [15].



Obrázek 3.8: Rozdíl mezi originální FlowPic reprezentací (vlevo) a rotovanou verzí (vpravo).

Kapitola 4

Návrh řešení

Tato kapitola popisuje datové sady, které budou využity pro trénování modelu a následné porovnání s výsledky dosažených v referenční práci [15]. Zároveň je zde popsán navrhovaný model obsahující jak *encoder* $f(\cdot)$, tak projekční hlavu $g(\cdot)$. Definován je zde také lineární model sloužící k výsledné klasifikaci, který v podstatě při tzv. *fine tuningu* v konečné fázi nahradí vícevrstvý perceptron $g(\cdot)$. Součástí návrhu řešení je také popis samotného trénování a aplikace augmentací.

4.1 Datové sady

K tomu, aby bylo možné efektivně trénovat neuronovou síť, je nezbytné mít k dispozici dostatečně velké množství dat. Tato práce bude pracovat se čtyřmi různými datasety: Mirage19 [1], Mirage22 [11], Ucdavis-icdm19 [23] a UTMobileNetTraffic2021 [12]. Každý z těchto datasetů je rozdělen na trénovací část, která obsahuje největší množství dat a slouží k učení modelu, validační část, která se využívá ke sledování výkonnosti modelu během trénování a k optimalizaci jeho hyperparametrů a testovací část, která je určena pro finální vyhodnocení výkonu modelu na dosud neviděných datech, aby bylo možné objektivně posoudit jeho generalizaci.

4.1.1 Mirage19

Dataset mirage19 zahrnuje síťový provoz generovaný 40 aplikacemi pro Android, které spadají do 16 různých kategorií dle klasifikace portálu pro distribuci aplikací Google Play. Celkem bylo v rámci datasetu shromážděno 4 606 PCAP záznamů [1].

Společné vlastnosti, které dataset popisuje jsou ID toku, zdrojová a cílová IP adresa, zdrojový a cílový port, protokol, čas prvního a posledního paketu, směr paketů, délka paketů, flagy, informace o označení (*label*), celkový počet paketů a Android aplikace, ze které byl síťový provoz zachycen.

Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
Mirage19	train	žádný	20	51 798	28,62 s	30,33
	val			5 776	28,40 s	30,43
	test			6 418	26,66 s	30,33

Tabulka 4.1: Analýza datové sady Mirage19 bez filtru.

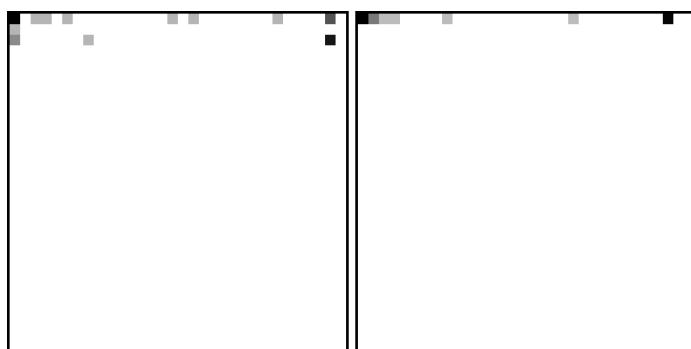
Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
Mirage19	train	>30pkts	20	38 959	20,52 s	31,89
	val			4 402	28,40 s	31,89
	test			4 818	26,66 s	31,89

Tabulka 4.2: Analýza datové sady Mirage19 s filtrem >30pkts.

Pro datovou sadu byl zvolen filtr, který zpracovává pouze toky s více než 30 pakety. Na rozdíl od referenčních prací [8], které volily filtr s hranicí 10 paketů, byla tato vyšší hodnota zvolena z důvodu, že použitá datová sada neobsahuje toky kratší než 10 paketů a zároveň obsahuje pouze malé množství toků kratších 20 paketů, konkrétně pouze 1,09 %.



Obrázek 4.1: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage19 bez filtru.



Obrázek 4.2: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage19 s filtrem.

Jak lze pozorovat na FlowPic reprezentacích, tato datová sada není pro vytváření FlowPiců příliš vhodná, což se projeví na finální přesnosti modelu.

4.1.2 Mirage22

Tento dataset obsahuje data zachycená z provozu devíti mobilních aplikací, které za doby pandemie COVID-19 zaznamenaly velký růst. Ony aplikace jsou Discord, GotoMeeting, Meet, Messenger, Skype, Slack, Teams, Webex a Zoom. Trénovací část obsahuje celkem 21 685 vzorků, validační 2 410 a trénovací 2 678. Celkový počet tříd činí 9 [11].

Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
Mirage22	train			21 685	333,28 s	6 708,65
	val	>10pkts	9	2 410	316,36 s	6 370,42
	test			2 678	309,81 s	7 258,81

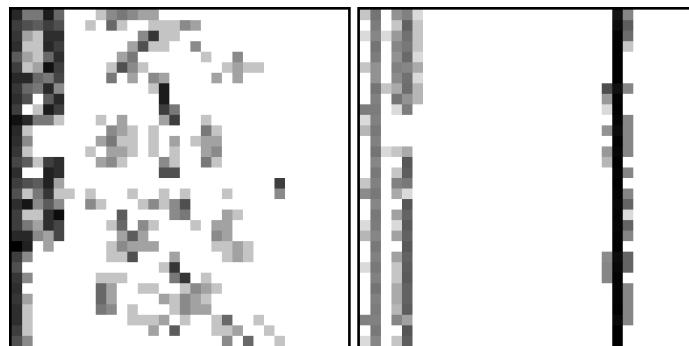
Tabulka 4.3: Analýza datové sady Mirage22 s filtrem >10pkts.

Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
Mirage22	train			3 700	614,51 s	39 466,81
	val	>1 000pkts	9	412	617,08 s	37 571,55
	test			457	588,74 s	35 279,41

Tabulka 4.4: Analýza datové sady Mirage22 s filtrem >1 000pkts.



Obrázek 4.3: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage22 s filtrem >10pkts.



Obrázek 4.4: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Mirage22 s filtrem >1 000pkts.

Oproti datové sadě Mirage19 lze pozorovat, že výsledné FlowPic reprezentace jsou mnohem plnější a bohatší. To umožní dosáhnout výraznějších rozdílů mezi augmentovanými vzorky, což povede k lepší generalizaci modelu.

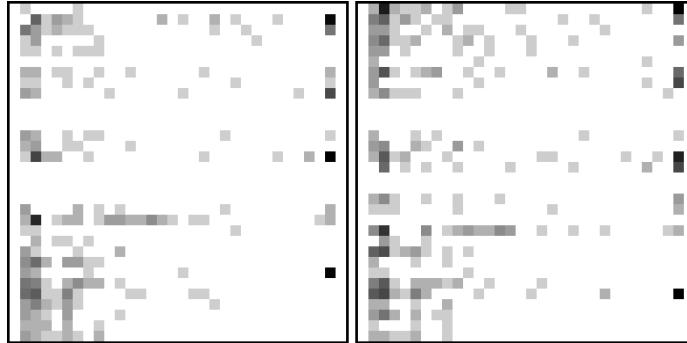
4.1.3 Ucdavis-icdm19

Dataset ucdavid-icdm19 obsahuje zachycený provoz z pěti služeb společnosti Google. Konkrétně se jedná o služby Google Drive, Youtube, Google Docs, Google Search a Google Music. Dataset obsahuje 6 672 vzorků. Celkový počet tříd činí 5 [23].

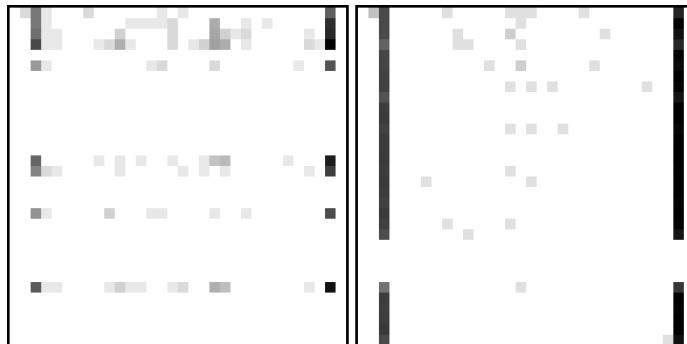
Tato datová sada se od ostatních odlišuje tím, že pro testování modelu nabízí dvě testovací části, *human* a *script*, které se liší od té využívané pro trénování. Část *script* je generována skripty a automatizovanými nástroji, zatímco *human* zachycuje provoz reálných uživatelů.

Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
Ucdavis-icdm19	pretraining			6 439	42,92 s	6 652,93
	val			1 288	43,50 s	6 811,09
	human	žádný	5	83	31,33 s	7 666,37
	script			150	42,94 s	7 130,64

Tabulka 4.5: Analýza datové sady Ucdavis-icdm19.



Obrázek 4.5: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady Ucdavis-icdm19.



Obrázek 4.6: Porovnání FlowPic reprezentací vytvořených z části *script* (vlevo) a *human* (vpravo).

Na vytvořených FlowPic reprezentacích můžeme pozorovat velice dobrou distribuci paketů napříč FlowPicem. Augmentace aplikované na tyto reprezentace tak budou mnohem výraznější než u prázdnějších FlowPiců, což může vést k lepší generalizaci modelu.

4.1.4 UTMobileNetTraffic2021

Dataset obsahuje více než 29 hodin síťového provozu generovaného jak automatizovanou platformou, tak lidskými uživateli. Zachycený provoz pochází z 16 nejoblíbenějších mobilních aplikací, jako jsou Reddit, Spotify, YouTube, Instagram, Hulu a další. Celkem tento dataset obsahuje 9 460 vzorků a celkový počet tříd činí 14 [12].

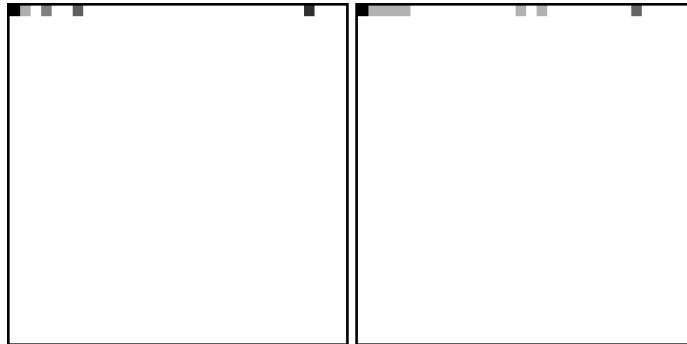
Název	Rozdelení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
UTMobileNetTraffic2021	train			7 568	13,47 s	2 537,27
	val	žádný	14	946	67,40 s	1 554,43
	test			946	1,59 s	1 810,65

Tabulka 4.6: Analýza datové sady UTMobileNetTraffic2021 bez filtru.

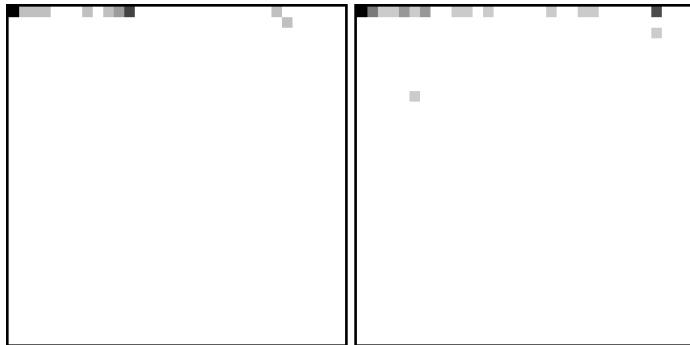
Název	Rozdelení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
UTMobileNetTraffic2021	train			5 028	18,23 s	3 808,52
	val	>30pkts	14	656	93,25 s	2 232,41
	test			648	1,05 s	2 633,73

Tabulka 4.7: Analýza datové sady UTMobileNetTraffic2021 s filtrem >30pkts.

Podobně jako u datové sady Mirage19 je i zde zvolen větší filtr než v referenční práci [8] z důvodu absence toků kratších než 10 paketů.



Obrázek 4.7: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady UTMobileNetTraffic2021 bez filtru.



Obrázek 4.8: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady UTMobileNetTraffic2021 s filtrem.

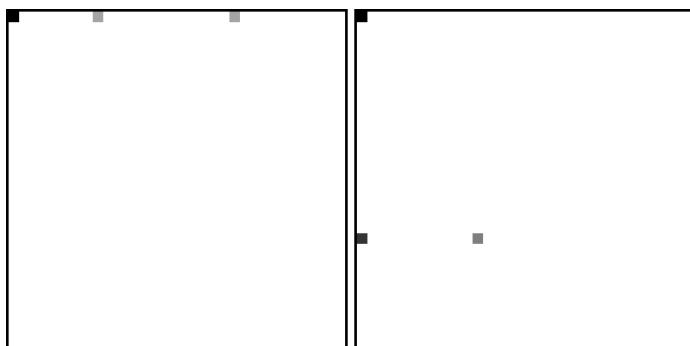
Podobně jako u datové sady Mirage19 můžeme pozorovat velmi špatnou distribuci paketů. Tedy i tato datová sada nemůže dosahovat dobrých výsledků.

4.1.5 CESNET

Poslední datová sada byla vytvořena sdružením CESNET¹. Jedná se o největší dataset ze všech použitých, a tedy obsahuje i nejvyšší počet síťových toků. Jeho celková velikost přesahuje 7,1 GB. Vzhledem k tomu, že cílem je využít tuto sadu výhradně pro tvorbu *embeddingů*, tedy pro kontrastivní učení, bude nasazena v kombinaci s ostatními datovými sadami.

Název	Rozdělení	Filtры	Třídy	Počet toků	Průměrná délka toku	Průměrný počet paketů
CESNET	žádné	žádné	15	590 962	70,00 s	519,20

Tabulka 4.8: Analýza datové sady CESNET.



Obrázek 4.9: FlowPic reprezentace vytvořené z dvou náhodně vybraných toků z datové sady CESNET.

Vytvořené FlowPic reprezentace se výrazně podobají těm z datových sad Mirage19, Mirage22 a UTMobileNetTraffic21. Aplikace časového filtru, který by odstraňoval toky kratší než nějaká zvolená délka, v tomto případě nebude efektivní, neboť značná část paketů

¹<https://www.cesnet.cz/>

přichází již během první sekundy komunikace. Výhodou této nové datové sady však může být její rozsah, vysoký počet toků totiž může do určité míry kompenzovat nízkou kvalitu FlowPic reprezentací.

4.2 Model

Tato práce si klade za cíl replikovat výsledky dosažené v [15]. Model použitý v této práci je proto odvozen od modelu prezentovaného v práci referenční.

4.2.1 Příprava dat

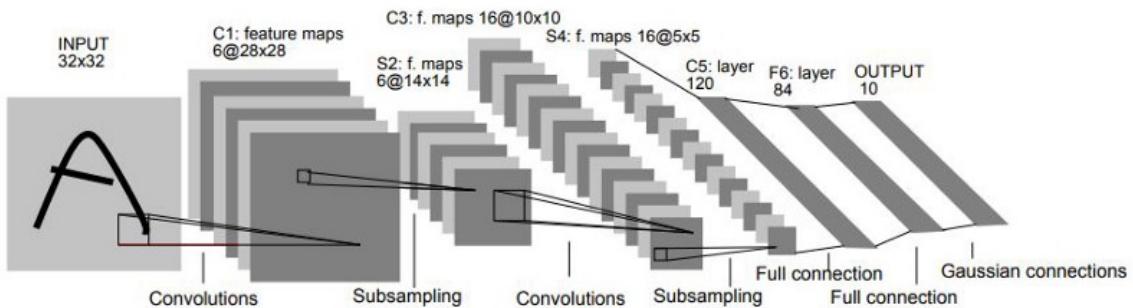
Framework *SimCLR* generuje dva augmentované pohledy \tilde{x}_i a \tilde{x}_j pomocí různých náhodných augmentací, které jsou popsány v sekci 3.5, jež budou tvořit pozitivní páry. Tyto pohledy jsou následně použity k vytvoření FlowPic reprezentací.

Vzhledem k výpočetní náročnosti standardně velkých FlowPic reprezentací o rozměrech 1500×1500 , popsaných v sekci 2.3.2, bude v této práci použita jejich zmenšená verze nazvaná mini-FlowPic, která je detailně popsána v sekci 2.3.3. Konkrétně bude použita velikost 32×32 . Standardní FlowPic mapuje velikost paketu v poměru 1:1 na ose Y (do maximální MTU), zatímco menší verze používá transformaci, která je podrobněji popsána v sekci 2.3.3. Osa X v obou případech reprezentuje čas.

4.2.2 Extrakce reprezentací

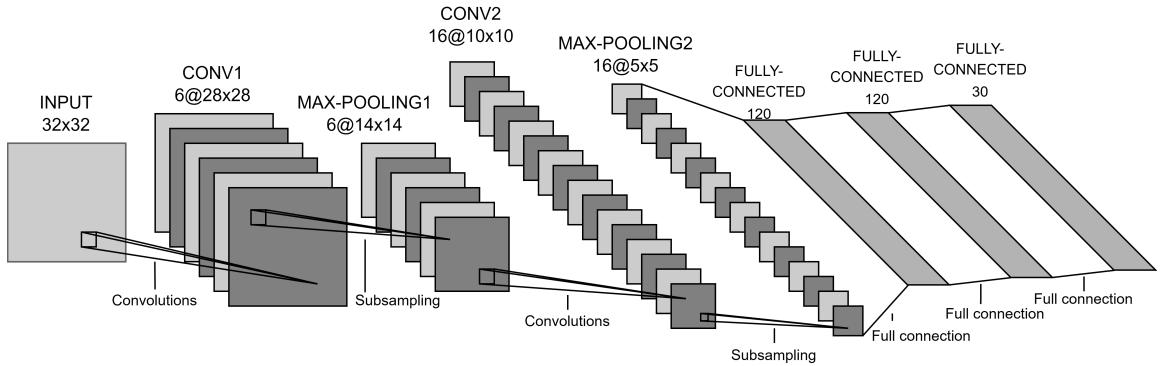
Extrakce reprezentací je část, kde se bude využívat kontrastivní učení společně s *SimCLR* frameworkm.

Klíčovou komponentou v této fázi je CNN, která je v sekci 3.1.2 teoreticky popsána, a v rámci *SimCLR* je označována jako $f(\cdot)$. Podobně jako v pracích [15] a [28], i zde vychází základní architektura z modelu LeNet-5 [19].



Obrázek 4.10: LeNet-5 architektura [19].

Původní LeNet-5 architekturu, popsanou na obrázku 4.10, je potřeba pro účely této fáze upravit. V původním modelu jsou vrstvy S2 a S4 implementovány jako *pooling* vrstvy využívající průměrování, tedy tzv. *average pooling*, ovšem dle referenční práce [15] je vhodnější využít výběr maxima (tzv. *max pooling*). Další modifikací je nahrazení posledních dvou vrstev (F6 a OUTPUT) dvěma lineárními vrstvami o velikosti 120 a 30. Tímto získáme 120 dimenzionální vektor $h = f(FlowPic)$ a 30 dimenzionální vektor podobnosti $z = g(h)$. Pro trénink bude využita ztrátová funkce, která je popsána v sekci 3.4.4.



Obrázek 4.11: Ilustrace architektury pro 32×32 Mini FlowPic.

4.2.3 Lineární klasifikátor

Jako možnost zhodnocení naučených reprezentací je zvolen lineární klasifikátor, který je trénován na klasifikaci vektorů reprezentací. Extrakce, popsaná v sekci 4.2.2, byla prováděna jako učení bez učitele (tzv. *unsupervised learning*), nicméně tento lineární klasifikátor bude trénován s učitelem (tzv. *supervised learning*), tj. s pomocí označených vzorků. V této fázi se zmrazí váhy využívané při trénování $f(\cdot)$, což znamená že extraktor se stane fixním extraktorem reprezentací, aby bylo možné lineární klasifikátor efektivně trénovat.

Lineární klasifikátor je jednoduchá, jednovrstevná neuronová síť, která může být popsaná vzorcem $y = xA^T + b$, kde A, b jsou klasifikační parametry, x je vstupní vektor a y je výstupem. Jak popisuje sekce 4.2.2, vstupní vektor x je 120 dimenzionální.

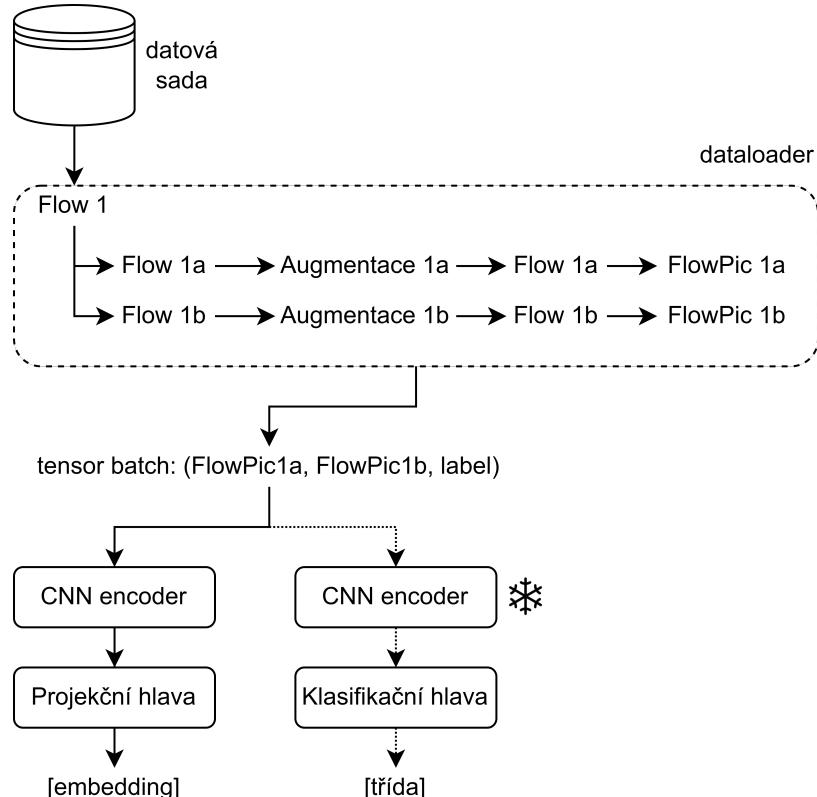
4.3 Návrh experimentů

Toky z vybraných datových sad budou načítány pomocí dataloaderů z knihovny FlowMind. V souladu s frameworkm SimCLR, popsaným v sekci 3.4.3, bude každý tok duplikován, aby na něj mohla být následně aplikována augmentace. Zvolené augmentační techniky, uvedené v sekci 3.5, budou, s výjimkou rotace FlowPic reprezentace, aplikovány přímo na samotný tok před konstrukcí výsledného FlowPicu. Rotace jako jediná proběhne až po vytvoření reprezentace. Pro každý pozitivní pár bude použita tatáž augmentační technika, lišící se pouze v konkrétních hodnotách parametrů, které budou náhodně vybírány z předem definovaných intervalů. Tyto intervaly jsou navrženy tak, aby umožnily srovnání dosažených výsledků s referenčními pracemi [15] a [8]. Konkrétně bude hodnota α pro modifikaci RTT vybírána z intervalu $[0,5; 1,5]$, zatímco hodnota b pro časový posun při modifikaci IAT bude náhodně zvolena z rozsahu $[-1,0; 1,0]$. Augmentace vynechání paketů bude pracovat s $\Delta t = 0,1$ a náhodně určeným časovým bodem t v intervalu $[0; \text{délka toku}]$. Poslední augmentací je rotace FlowPic reprezentace, která se bude pohybovat v rozmezí $[-10^\circ; 10^\circ]$.

Při učení bez učitele jsou výsledné FlowPic reprezentace využívány k trénování jak CNN encoderu $f(\cdot)$, tak projekční hlavy $g(\cdot)$. Cílem je získat kvalitní *embeddings*, které mohou být předány ztrátové funkci. Na základě zpětné propagace se následně aktualizují váhy obou těchto částí modelu. Po ukončení tohoto trénovacího procesu následuje *fine-tuning* (fáze učení s učitelem), při kterém je *encoder* zmrazen, tedy jeho váhy zůstávají beze změny. Původní projekční hlava $g(\cdot)$ je odstraněna a nahrazena novou klasifikační hlavou (lineárním klasifikátorem), která je jako jediná část modelu aktivně trénována a optimalizována na základě dat, která zatím nebyla pro trénování použita. V kontextu této práce se jedná

o testovací části datových sad. Rozdílem mezi projekční hlavou a klasifikační hlavou je také jejich výstup. Zatímco projekční hlava produkuje třicetidimenzionální vektor podobnosti z , klasifikační hlava se snaží výsledky mapovat na jednotlivé třídy.

Celý proces trénování, zahrnující jak učení bez učitele, tak učení s učitelem, bude zopakován celkem pětkrát a výsledky budou průměrovány. Pro širší kontext bude také uvedena směrodatná odchylka mezi výsledky. Důvodem je skutečnost, že hodnoty jednotlivých augmentací jsou vybírány náhodně a výsledky se tak mohou mírně lišit.



Obrázek 4.12: Ilustrace procesu trénování. Sněhová vločka značí zmražení *encoderu*.

Kapitola 5

Implementace a vyhodnocení dosažených výsledků

Tato kapitola popisuje zvolenou implementaci, výběr hyperparametrů a postup trénování neuronové sítě. Z vybraných datových sad, které jsou v této práci pro trénink modelu zvoleny, využívá referenční práce [14] pouze jednu, konkrétně Ucdavis-icdm19 [23], popsanou v sekci 4.1.3. Z tohoto důvodu budou výsledky na ostatních datových sadách porovnávány s výsledky dosažených v práci [8], která práci referenční rozšiřuje o další datové sady.

Pro verzování kódu a dokumentace byl využit GitHub repozitář¹.

5.1 Použité knihovny

Tato sekce se zaměřuje na knihovny, které byly využity při implementaci neuronové sítě a všech souvisejících komponent. Obě zmíněné knihovny jsou pro tuto práci esenciální. Kromě jejich popisu tato sekce zároveň demonstруje implementaci jednotlivých částí neuronové sítě.

5.1.1 PyTorch

Pro implementaci konvoluční neuronové sítě i lineárního klasifikátoru byla využita knihovna PyTorch², což je velmi bohatá knihovna, která poskytuje četné funkce umožňující vytváření stavebních bloků pro hluboké učení. Knihovna poskytuje základní datovou strukturu *tensor*, což není nic jiného než vícerozměrné pole objektů stejného typu. Lze říci, že na nižší úrovni abstrakce jsou veškeré výpočty v PyTorch založeny na tensorech a operacích nad nimi [18].

Konkrétněji je *tensor* zobecněný způsob reprezentace skalárů, vektorů a matic, který lze definovat jako n-rozměrnou matici. 0-rozměrný *tensor* (tj. jedno číslo) se nazývá skalár, 1-rozměrný se nazývá vektor, 2-rozměrný se nazývá matice a 3-rozměrný se také označuje jako krychle [18].

V této práci je pro ukládání dat využíván 4D (tedy čtyřrozměrný) *tensor* ve formátu [B, C, H, W], kde B označuje velikost batche (tzv. *batch size*), C označuje počet kanálů, H označuje výšku FlowPicu a W jeho šířku. Ve všech případech trénování (pokud není uvedeno jinak) je využito [32, 1, 32, 32], tedy *batch size* 32, 1 kanál a velikost FlowPicu 32x32.

¹<https://github.com/Simkuba/IBT>

²<https://pytorch.org/>

Implementace CNN

PyTorch nabízí předpřipravené třídy pro implementaci konvolučních neuronových sítí, což výrazně zjednodušuje jejich vytváření. Vzhledem k rozdílu mezi tensorům používaným v této práci je vhodné využívat 2d varianty těchto tříd, které jsou navrženy pro práci se 4D tensoři. Konkrétně jsou využity třídy `nn.Conv2d()`, `nn.BatchNorm2d()` a `nn.MaxPool2d()`. Výsledný *encoder* $f(\cdot)$ poté ještě obsahuje jednu lineární vrstvu, která je implementována pomocí třídy `nn.Linear()`. Aktivační *ReLU* funkce je implementována pomocí třídy `nn.ReLU()`. Všechny tyto vrstvy lze obalit třídou `nn.Sequential()`, což je kontejnerová třída, která umožňuje řetězit více vrstev dohromady v přesném pořadí, v jakém jsou zapsané.

Implementace výsledného *encoderu* $f(\cdot)$ tedy vypadá následovně:

```
1 nn.Sequential(
2     nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5),
3     nn.BatchNorm2d(6),
4     nn.ReLU(),
5     nn.MaxPool2d(kernel_size=2, stride=2),
6
7     nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5),
8     nn.BatchNorm2d(16),
9     nn.ReLU(),
10    nn.MaxPool2d(kernel_size=2, stride=2),
11
12    nn.Flatten(),
13    nn.Linear(400, 120),
14    nn.ReLU(),
15 )
```

Výpis 5.1: Implementace *encoderu* $f(\cdot)$.

Implementace *encoderu* tedy vychází z referenční práce [15]. Navíc jsou ovšem přidané normalizační vrstvy `nn.BatchNorm2d`, jelikož bylo v průběhu trénování pozorováno, že mají pozitivní vliv na výsledné reprezentace a tedy i na konečnou přesnost modelu.

Implementace projekční hlavy

V rámci frameworku SimCLR navazuje na *encoder* $f(\cdot)$ projekční hlava $g(\cdot)$, která se využívá k získání třicetidimenzionálního vektoru podobnosti $z = g(h)$. Projekční hlava obsahuje pouze lineární vrstvy implementované pomocí třídy `nn.Linear()`. Nově se zde využívá také *Tanh* aktivační funkce implementovaná pomocí třídy `nn.Tanh()`.

```
1 nn.Sequential(
2     nn.Linear(120,120),
3     nn.ReLU(),
4     nn.Linear(120,30),
5     nn.Tanh()
6 )
```

Výpis 5.2: Implementace projekční hlavy $g(\cdot)$.

Implementace lineárního klasifikátoru

Jak je popsáno v sekci 4.2.3, lineární klasifikátor používaný pro tzv. *fine-tuning* je jednoduchá jednovrstevná neuronová síť. Z toho důvodu ji ani není potřeba obalovat kontejnerovou vrstvou `nn.Sequential()` a lze ji implementovat jednou třídou `nn.Linear()`. Vstupem je

120-dimenziona lní vektor a výstupem je vektor o velikosti počtu tříd, který se liší podle použité datové sady.

5.1.2 FlowMind

K načítání dat z datových sad a jejich převodu do FlowPic reprezentací byla využita knihovna FlowMind, vyvýjená na Fakultě informačních technologií Vysokého učení technického v Brně. Pro tzv. *dataloader*, který zajišťuje načtení a následné zpracování dat, byla použita funkce `create_flowpic_dataloader()`. Tato funkce byla pro účely kontrastivního učení mírně upravena tak, aby místo jediné FlowPic reprezentace vracela tensor obsahující dvě. Za tímto účelem byla knihovna rozšířena o funkci `build_two_augmented_flows()`, která toto chování implementuje.

Proces zpracování dat pomocí funkce `create_flowpic_dataloader()` je poměrně přímočarý. Funkce, kromě dalších argumentů, přijímá cestu k datové sadě, jejíž jednotlivé toky postupně zpracovává. Toky jsou nejprve převedeny z formátu `.csv` do Python datové struktury *dictionary*, následně jsou filtrovány na základě počtu paketů či délky toku, a poté je každý tok duplikován a předán již zmíněné funkci `build_two_augmented_flows()`. Ta následně pomocí dalších tříd definovaných v knihovně FlowMind vytvoří dvě augmentované FlowPic reprezentace.

Pomocí *dataloaderu* jsou na jednotlivé toky také aplikovány vybrané augmentace, které jsou podrobně popsány v sekcích 3.5 a 4.3. Všechny z nich, s výjimkou rotace FlowPicu, se aplikují přímo na síťový tok ještě před samotnou konstrukcí FlowPic reprezentace. K provádění těchto úprav slouží parametry `flow_transform1` a `flow_transform2`, jenž se dále předávají až ke třídě odpovědné za konstrukci samotného toku, tedy objektu `Flow`. Naproti tomu augmentace rotace FlowPicu se neprovádí prostřednictvím *dataloaderu*, ale aplikuje se až na vytvořenou FlowPic reprezentaci v rámci tréninkové smyčky.

Pro korektní načtení dat a sestavení FlowPic reprezentace je u *dataloaderu* nutné definovat několik klíčových parametrů. Mezi nimi je klíčové slovo `meta_key`, které slouží k určení, podle čeho budou sestavovány *labely*, dále cesta k datové sadě, hodnota `batch size`, časový interval $[0; 15]$, určující prvních patnáct sekund toku, a velikostní interval $[0; 1\,500]$, kde hodnota 1 500 odpovídá MTU (*Maximum Transmission Unit*). Dále je třeba nastavit parametr `bidirectional` na hodnotu `False`, aby tensor obsahoval pouze jeden kanál. Nakonec se předává parametr `dp_transform`, který zajišťuje transformaci *labelu* na číselnou hodnotu.

Knihovna rovněž poskytuje implementaci ztrátové funkce NT-Xent, která je matematicky ověřená a byla využita ve fázi učení bez učitele.

5.2 Postup trénování

Pro trénování byly hyperparametry nastaveny tak, aby bylo možné dosažené výsledky porovnávat s referenčními práci [15] a [8].

Pro trénování bez učitele (*unsupervised learning*) byl nastaven *learning rate* na hodnotu 0,001, jako ztrátová funkce byla využita NT-Xent Loss, popsaná v sekci 3.4.4, s teplotním parametrem 0,07. Jako optimalizační algoritmus byl zvolen Adam, který patří mezi metody využívající výpočet gradientu. Samotné trénování využívalo předčasné ukončení (*early stopping*), které sledovalo vývoj ztrátové funkce na validačním setu s maximální trpělivostí (*patience*) 3, přičemž maximální počet epoch byl omezen na 30.

Pro *fine tuning*, tedy trénování s učitelem, byla hodnota *learning rate* nastavena na 0,01, pro ztrátovou funkci byla využita *Cross-Entropy Loss*. Jako optimalizační algoritmus

byl, stejně jako při trénování bez učitele, zvolen Adam. Přesnost modelu se počítala pomocí PyTorch třídy `MultiClassAccuracy`. Také zde bylo využito předčasné ukončení, které sledovalo minimální zlepšení modelu s hodnotou 0,001 a trpělivostí 5. Celkový počet epoch byl omezen na 20.

Všechny datové sady, s výjimkou CESNET popsaného v sekci 4.1.5, jsou rozděleny na části *train*, *val* a *test*, přičemž každá z nich slouží k jiné fázi učení. Datová sada Ucdavis-icdm19, popsaná v sekci 4.1.3, dokonce obsahuje 2 testovací části, *human* a *script*. Část *train* je určena pro trénování CNN a zároveň se na ni aplikují augmentace. Stejná část je, opět s výjimkou datasetu CESNET, využita i při *fine-tuningu*, avšak již bez aplikování augmentací. Část *val* slouží k validaci modelu během trénování a jsou na ni aplikovány stejné augmentace jako na trénovací sadu. Poslední částí je *test*, která slouží k ověření výsledné přesnosti modelu. Vzhledem k již zmíněným odlišnostem datové sady Ucdavis-icdm19 je model testován samostatně jak proti celé části *human*, tak i proti celé části *script*, a tedy celkově testován dvakrát.

Jednotlivé smyčky jsou implementovány pomocí čtyř funkcí. Liší se tím, jaká část datové sady je právě využívána a jakým způsobem je model trénován. Trénovací smyčka nastaví model do trénovacího módu pomocí PyTorch metody `train()`. Z `dataloaderu` jsou postupně vybírány dvojice FlowPic reprezentací, které jsou následně předány modelu. Po získání vektorů z_1 a z_2 je vypočítána ztrátová funkce a aktualizují se váhy.

Validační smyčka se od trénovací liší tím, že celý proces, tedy akce probíhající v trénovací smyčce, je zabalen do PyTorch kontextového manažera `torch.no_grad()`, který během svého trvání (v bloku `with`) vypne výpočet gradientů. Model je zároveň nastaven do evaluačního módu pomocí PyTorch metody `eval()`. V samotné validační smyčce se oproti trénovací již model neoptimalizuje, tedy váhy zůstávají nezměněné. Důvodem je to, že validační smyčka slouží pouze k vyhodnocení modelu, nikoli k jeho trénování.

Třetí je klasifikační smyčka, která je použita při učení s učitelem. Jelikož se jedná o fázi tzv. *fine-tuningu*, je nutné *encoder* $f(\cdot)$ zmrazit. Toho je dosaženo nastavením atributu tenzorů `requires_grad` na hodnotu `False`. Ve smyčce je také odstraněna projekční hlava $g(\cdot)$ CNN modelu a nahrazena lineárním klasifikátorem (MLP modelem). Oba modely musí být náležitě nastaveny, tedy CNN model je přepnut do evaluačního módu pomocí `eval()` a MLP model do trénovacího módu `train()`. Dalším rozdílem je také použití jiné ztrátové funkce.

Poslední smyčkou je testovací smyčka, která nastavuje oba modely do evaluačního módu `eval()` a stejně jako validační smyčka využívá kontextový manažer `torch.no_grad()`.

Celý proces trénování byl zopakován v pěti bězích u každé datové sady, přičemž uváděné přesnosti představují průměrné hodnoty napříč všemi běhy. U každé hodnoty je uvedena také směrodatná odchylka.

5.3 Dosažené výsledky

Tato sekce prezentuje výsledky dosažené na vybraných datových sadách popsaných v sekci 4.1. Hodnoty jsou uváděny pro jednotlivé typy augmentací, aby bylo možné je mezi sebou porovnat. Vzhledem k povaze některých datových sad jsou u vybraných případů prezentovány výsledky bez i s aplikovaným filtrováním. Kde je to možné, jsou výsledky této práce porovnány s výsledky dosaženými v referenčních studiích [15, 8]. Na závěr jsou uvedeny výsledky experimentů provedených s větší variantou Mini FlowPic reprezentací 64×64 .

5.3.1 Mirage19 a Mirage22

Výsledky, které byly pozorovány nad datovými sadami Mirage19 a Mirage22 bohužel nelze přímo srovnat ani s referenční prací [15], která tyto datasety nepoužívá, tak ani s prací [8], která sice datové sady používá, ovšem pouze v tzv. *supervised setting*, tedy nepoužívá kontrastivní učení.

Na základě analýzy datové sady Mirage19, popsané v tabulkách 4.1 a 4.2, lze očekávat, že výsledná přesnost modelu bude velmi nízká. Ve srovnání s ostatními datovými sadami obsahuje Mirage19 relativně krátké toky, což samo o sobě nemusí nutně znamenat nízkou výkonnost, nicméně lze předpokládat, že počet paketů v těchto tocích bude výrazně nižší než u déle trvajících přenosů. Klíčovým ukazatelem je průměrný počet paketů, který je u datové sady Mirage19 nejnižší ze všech, což vede k tomu, že výsledné FlowPic reprezentace jsou z velké části nulové, což můžeme pozorovat na reprezentacích 4.1 a 4.2. Tento problém je navíc umocněn tím, že většina paketů dorazí již v prvních desetinách sekundy od začátku komunikace. Všechny tyto faktory významně ztěžují modelu se naučit vzory komunikace.

Analýza datové sady Mirage22, popsaná v tabulkách 4.3 a 4.4, naznačuje, že výkonnost modelu bude výrazně vyšší než v případě datové sady Mirage19. Průměrný počet paketů na tok je řádově vyšší a průměrná délka toku je rovněž výrazně delší. Lze tedy oprávněně očekávat, že přesnost modelu bude ve srovnání s datovou sadou Mirage19 podstatně lepší. Toto je možné pozorovat i na reprezentacích 4.3 a 4.4.

Filtr	Mirage19		Mirage22	
	žádný	>30pkts	>10pkts	>1 000pkts
Modifikace RTT	$39,13 \pm 1,25$	$41,07 \pm 1,64$	$67,96 \pm 2,82$	$77,51 \pm 1,34$
Modifikace IAT	$38,79 \pm 1,71$	$38,49 \pm 0,47$	$60,87 \pm 2,03$	$74,84 \pm 2,83$
Vynechání paketů	$41,90 \pm 0,92$	$42,68 \pm 1,06$	$57,60 \pm 1,50$	$67,35 \pm 1,65$
FlowPic rotace	$31,82 \pm 0,40$	$33,01 \pm 0,77$	$54,81 \pm 1,28$	$71,42 \pm 1,95$

Tabulka 5.1: Dosažené výsledky u datových sad Mirage19 a Mirage22.

Na výsledcích lze pozorovat, že očekávání se potvrdila a přesnost Mirage19 se pohybuje velmi nízko, zatímco Mirage22 dosahuje téměř dvojnásobné přesnosti. Je také zřejmé, že výraznější filtrování počtu paketů v toku viditelně zlepšuje celkovou přesnost modelu a to jak u Mirage19, tak Mirage22. Zajímavé je, že augmentace modifikace RTT, která běžně dominuje jak v referenčních pracích [15, 8], tak v této práci, v případě Mirage19 dosahuje nižších výsledků než augmentace vynechání paketů, která se většinou řadí k těm slabším. Je to dáno tím, že u FlowPiců vytvořených z těchto datových sad je augmentace ve formě vynechávání paketů výraznější, jelikož modifikace RTT u reprezentací, které jsou z velké části prázdné, nemá tak výrazný efekt.

Přestože replikace referenční práce [8] nepoužívá datové sady Mirage19 a Mirage22 pro tzv. *unsupervised setting* a výsledky uvádí pouze pro trénování s učitelem, je možné dosažené výsledky porovnat alespoň nepřímo.

Filtr	Tato práce		Výsledky z [8]	
	Mirage22 ->10pkts	Mirage22 ->1 000pkts	Mirage22 ->10pkts	Mirage22 ->1 000pkts
Modifikace RTT	67,96 \pm 2,82	77,51 \pm 1,34	93,75 \pm 0,83	91,48 \pm 2,12
Modifikace IAT	60,87 \pm 2,03	74,84 \pm 2,83	92,80 \pm 1,21	86,73 \pm 3,88
Vynechání paketů	57,60 \pm 1,50	67,35 \pm 1,65	92,34 \pm 1,10	87,19 \pm 2,52
FlowPic rotace	54,81 \pm 1,28	71,42 \pm 1,95	88,25 \pm 1,20	87,32 \pm 2,24

Tabulka 5.2: Porovnání dosažených výsledků na Mirage22 s prací [8] (*supervised setting*).

Jak již bylo zmíněno, hodnoty nelze přímo porovnávat, je ovšem patrné, že modifikace RTT představuje nejefektivnější formu augmentace, zatímco rotace FlowPic reprezentace dosahují nejhorších výsledků. Zároveň lze pozorovat opačný trend ve vztahu k intenzitě filtrování, zatímco u kontrastivního učení vede silnější filtrování ke zlepšení výkonu, v případě učení s učitelem má opačný efekt.

Filtr	Tato práce	Výsledky z [8]
	Mirage19 ->30pkts	Mirage19 ->10pkts
Modifikace RTT	41,07 \pm 1,64	74,28 \pm 1,22
Modifikace IAT	38,49 \pm 0,47	70,33 \pm 1,26
Vynechání paketů	42,68 \pm 1,06	67,55 \pm 1,46
FlowPic rotace	33,01 \pm 0,77	60,35 \pm 1,17

Tabulka 5.3: Porovnání dosažených výsledků na Mirage19 s prací [8] (*supervised setting*).

Ze výsledných hodnot je patrné, že učení s učitelem je pro datovou sadu Mirage19 výrazně vhodnější než kontrastivní přístup. Hlavním důvodem je již zmíněná struktura toků, která není pro kombinaci kontrastivního učení a FlowPic reprezentace příliš vhodná.

5.3.2 Ucdavis-icdm19

Datová sada Ucdavis-icdm19 dosahovala velmi dobrých výsledků v obou referenčních pracích [15, 8]. Výsledky dosažené v této práci lze přímo porovnat jak s originální studií [15], tak s její replikací [8], jelikož obě tyto práce využívají danou datovou sadu i pro kontrastivní učení.

Z nalýzy této datové sady, uvedené v tabulce 4.5, vyplývá, že průměrná délka toku není nijak zvlášť vysoká. Díky vysokému průměrnému počtu paketů na tok je však datová sada ideální pro tvorbu FlowPic reprezentací, což je ukázáno na 4.5. Tyto reprezentace jsou tzv. velmi bohaté, což modelu výrazně usnadňuje schopnost data generalizovat, na rozdíl od jiných datových sad, jako například Mirage19 nebo Mirage22.

Test set	Ucdavis-icdm19	
	script	human
Modifikace RTT	$97,20 \pm 0,87$	$73,01 \pm 1,37$
Modifikace IAT	$97,33 \pm 0,67$	$76,14 \pm 2,32$
Vynechání paketů	$96,40 \pm 1,30$	$72,29 \pm 4,09$
FlowPic rotace	$96,40 \pm 0,60$	$77,59 \pm 1,37$

Tabulka 5.4: Dosažené výsledky u datové sady Ucdavis-icdm19.

Lze pozorovat rozdílnou přesnost mezi testovací sadou *script* a *human*. Důvodem pro tento rozdíl je s nejvyšší pravděpodobností způsob, kterým byly tyto sady vytvořeny. Zatímco sada *script* je generována automaticky, sada *human* zachycuje provoz reálných uživatelů. Z toho důvodu je provoz méně předpovídatelný a obsahuje více šumu.

Zajímavá je zde výkonnost augmentace rotace FlowPicu. Zatímco tato augmentace většinou dosahuje nejnižších přesností, jak je možné pozorovat na ostatních výsledcích prezentovaných v této práci, na testovací sadě *human* dosahuje lepších výsledků než, například, augmentace modifikace RTT, která ve většině případů dominuje.

Test set	Tato práce		Výsledky z [8]		Výsledky z [15]	
	script	human	script	human	script	human
Nejlepší výsledek	$97,33 \pm 0,67$	$77,59 \pm 1,37$	$92,18 \pm 0,31$	$74,69 \pm 1,13$	94,50	82,30

Tabulka 5.5: Porovnání nejlepších dosažených výsledků s referenčními pracemi [8, 15].

Jelikož ani jedna z referenčních prací [8, 15] neuvádí dosažené výsledky pomocí kontrastivního učení pro jednotlivé augmentace, je možné porovnat pouze nejlepší dosažené výsledky. Dosažené výsledky na testovací sadě *script* překonávají přesností obě referenční práce a to při použití augmentace modifikace IAT. Je však důležité zdůraznit, že práce [15] dosáhla uvedené přesnosti na sadě *script* při využití pouze 10 vzorků na třídu a na sadě *human* dokonce pouze 7 vzorků na třídu. Podobně práce [8] použila pro obě sady 10 vzorků na třídu. Oproti tomu tato práce využívala pro testování celou sadu *script* i *human*.

5.3.3 UTMobileNetTraffic2021

Podobně jako u datových sad Mirage19 a Mirage22, i analýza této datové sady, prezentovaná v tabulkách 4.6 a 4.8, naznačuje, že model zde nemůže dosahovat vysoké přesnosti. Hlavním důvodem jsou krátké průměrné délky toků jak v trénovací, tak v testovací části datové sady. Zejména testovací toky jsou natolik krátké, že jsou pro tvorbu FlowPic reprezentací téměř nepoužitelné, což je možné pozorovat na ukázce 4.7.

Referenční práce [8] dokonce tuto datovou sadu vůbec nepoužívá pro kontrastivní učení. Podobně jako u datových sad Mirage19 a Mirage22 je tak možné dosažené výsledky porovnat pouze v rámci učení s učitelem. Lze přitom oprávněně předpokládat, že výsledky dosažené v *supervised setting* budou výrazně lepší.

UTMobileNetTraffic2021		
Filtr	žádný	>30pkts
Modifikace RTT	47,12\pm 2,65	36,30 \pm 5,04
Modifikace IAT	42,41 \pm 2,82	37,04 \pm 5,54
Vynechání paketů	44,46 \pm 5,03	37,50\pm 6,85
FlowPic rotace	39,56 \pm 1,52	35,83 \pm 2,29

Tabulka 5.6: Dosažené výsledky u datové sady UTMobileNetTraffic2021.

Jak bylo již zmíněno, dosažené výsledky nelze přímo s referenční prací [8] srovnávat. Přestože [8] uvádí výsledky pro datovou sadu s aplikovaným filtrem minimálního počtu paketů na tok, výsledky dosažené v této práci vykazují lepší výkonnost v případě, že filtr použitý není, a proto budou pro srovnání využity právě ty.

Filtr	Tato práce	Výsledky z [8]
	žádný	>10pkts
Modifikace RTT	47,12\pm 2,65	81,32\pm 1,54
Modifikace IAT	42,41 \pm 2,82	81,91 \pm 2,12
Vynechání paketů	44,46 \pm 5,03	72,07 \pm 1,73
FlowPic rotace	39,56 \pm 1,52	79,45 \pm 1,28

Tabulka 5.7: Porovnání dosažených výsledků s prací [8] (*supervised setting*).

Ze srovnání je naprostě zřejmé, že tato datová sada pro kontrastivní učení v kombinaci s FlowPic reprezentacemi není vhodná a oproti učení s učitelem dosahuje výrazně nižší přesnosti.

5.3.4 CESNET

Postup trénování u této datové sady probíhal odlišně než u zbytku. Jelikož byl model trénován pouze na datové sadě od CESNETu, je možné ho uložit a pro následný *fine-tuning*, který probíhal nad jinými datovými sadami, jej pouze načíst.

Model byl tedy trénován na datové sadě CESNET. Jako augmentace byla zvolena modifikace RTT, která u datových sad generujících podobně vypadající FlowPic reprezentace dosahovala nejlepších výsledků. Při trénování nebyl aplikován žádný filtr na minimální počet paketů ani na minimální délku toku. *Fine-tuning* byl poté experimentálně vyzkoušen se všemi datovými sadami. Stejná datová sada byla také použita pro testování modelu.

Výsledky jsou rovnou také porovnány s modely, které byly čistě trénované na datové sadě, která byla použita pro *fine-tuning*. Tyto sady jsou označeny jako *Ostatní*.

Ucdavis-icdm19						
Test set	script	human	Mirage19	Mirage22	UTMobileNetTraffic2021	
CESNET	$98,30 \pm 0,67$	$74,46 \pm 1,01$	$30,72 \pm 0,02$	$73,39 \pm 0,63$	$62,77 \pm 0,36$	
Ostatní	$97,20 \pm 0,87$	$73,01 \pm 1,37$	$39,13 \pm 1,25$	$77,51 \pm 1,34$	$47,12 \pm 2,65$	
Rozdíl	1,1	1,45	-8,41	-4,12	15,65	

Tabulka 5.8: Dosažené výsledky modelu CESNET a dalších datových sad použitých pro *fine-tuning* a jejich porovnání.

Na výsledných hodnotách a řádku *Rozdíl* můžeme pozorovat, že výsledky dosažené pomocí *fine-tuningu* u datové sady Ucdavis, konkrétně části *script*, dosahují vyšší přesnosti než v případě modelu, který byl trénován pouze na této datové sadě. Dosahuje tak nejvyšší naměřené přesnosti v této práci. Výrazné zlepšení lze také pozorovat u datasetu UTMobileNetTraffic2021, který si polepšil o několik desítek procentních bodů.

Dalším zajímavým ukazatelem je směrodatná odchylka mezi jednotlivými běhy, která je výrazně nižší ve srovnání s experimenty provedenými na ostatních datových sadách.

5.3.5 FlowPic 64×64

Doposud tato práce ve všech případech používala FlowPic reprezentace o velikosti 32×32 . Tato velikost byla zvolena nejen proto, že je primárně využívána v referenčních pracích [15, 8], ale také s ohledem na výpočetní náročnost větších reprezentací. Referenční studie však uvádějí výsledky i pro FlowPic reprezentace velikosti 64×64 a 1500×1500 (tzv. plný FlowPic). Vzhledem k tomu, že trénování modelu s dvojnásobnou velikostí reprezentací trvá více než dvojnásobný čas, konkrétně přes 11 hodin, byla tato větší varianta testována pouze na datové sadě Ucdavis-icdm19, která v této práci dosahuje nejlepších výsledků, jak je prezentováno v sekci 5.3.2.

Kvůli větší velikosti reprezentací musí být drobně upraven i samotný model, konkrétně poslední lineární vrstva *encoderu* $f(\cdot)$, kde musí být upraven počet vstupních neuronů na 2 704.

```

1 self.encoder = nn.Sequential(
2     nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5),
3     nn.BatchNorm2d(6),
4     nn.ReLU(),
5     nn.MaxPool2d(kernel_size=2, stride=2),
6
7     nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5),
8     nn.BatchNorm2d(16),
9     nn.ReLU(),
10    nn.MaxPool2d(kernel_size=2, stride=2),
11
12    nn.Flatten(),
13    nn.Linear(2704, 120),
14    nn.ReLU(),
15 )

```

Výpis 5.3: Upravená implementace *encoderu* $f(\cdot)$.

Dosažené výsledky lze porovnat s menší variantou FlowPic reprezentací 32×32 .

Test set	Ucdavis-icdm19			
	FlowPic 32×32		FlowPic 64×64	
	script	human	script	human
Modifikace RTT	$97,20 \pm 0,87$	$73,01 \pm 1,37$	$97,73 \pm 1,12$	$73,49 \pm 1,90$
Modifikace IAT	$97,33 \pm 0,67$	$76,14 \pm 2,32$	$98,27 \pm 0,60$	$73,98 \pm 3,77$
Vynechání paketů	$96,40 \pm 1,30$	$72,29 \pm 4,09$	$94,93 \pm 0,76$	$71,33 \pm 5,41$
FlowPic rotace	$96,40 \pm 0,60$	$77,59 \pm 1,37$	$96,53 \pm 0,87$	$71,08 \pm 2,41$

Tabulka 5.9: Porovnání výsledků různých velikostí FlowPic reprezentací na datové sadě Ucdavis-icdm19.

Ze srovnání je patrné, že větší varianty FlowPic reprezentací v naprosté většině případů dosahují lepších výsledků na testovací sadě *script* než menší varianta. Naopak u sady *human* dosahuje model lepších výsledků při použití menších FlowPiců o velikosti 32×32 . Větší reprezentace zároveň obsahují více prázdných (nulových) oblastí ve výsledném FlowPicu, což s největší pravděpodobností snižuje účinnost augmentace formou vynechávání paketů. Tuto skutečnost potvrzují i dosažené výsledky.

Ovšem vzhledem k malým rozdílům ve finální přesnosti mezi jednotlivými velikostmi reprezentací a s ohledem na výpočetní náročnost se jeví velikost 32×32 jako výrazně vhodnější varianta než její větší protějšek.

Porovnání dosažených výsledků s referenčními pracemi není zcela přesné. Studie [15] sice prezentuje výsledky i pro kontrastivní učení, avšak přesné hodnoty neuvedí a výsledky jsou dostupné pouze graficky. Konkrétní hodnoty je tedy možné pouze odhadnout z grafů. Druhá referenční práce [8] výsledky pro 64×64 FlowPic v rámci kontrastivního učení vůbec nepublikuje a uvádí pouze výsledky dosažené při učení s učitelem.

Test set	Ucdavis-icdm19					
	Tato práce		Výsledky z [15]		Výsledky z [8]	
	script	human	script	human	script	human
Modifikace RTT	$97,73 \pm 1,12$	$73,49 \pm 1,90$	100,0	88,60	$97,02 \pm 0,46$	$71,49 \pm 1,59$
Modifikace IAT	$98,27 \pm 0,60$	$73,98 \pm 3,77$	99,53	87,33	$97,16 \pm 0,49$	$71,89 \pm 1,59$
Vynechání paketů	$94,93 \pm 0,76$	$71,33 \pm 5,41$	99,60	85,60	$96,84 \pm 0,63$	$71,33 \pm 1,45$
FlowPic rotace	$96,53 \pm 0,87$	$71,08 \pm 2,41$	98,87	87,07	$96,93 \pm 0,46$	$71,08 \pm 1,51$

Tabulka 5.10: Porovnání dosažených výsledků touto prací s výsledky publikované v referenčních pracích při učení s učitelem.

Přestože se jedná o porovnání výsledků dosažených pomocí kontrastivního učení a výsledků pomocí učení s učitelem, hodnoty dosažené touto prací jsou více než kompetitivní. Pokud nejdříve srovnáme výsledky s referenční prací [15], tak je patrné, že učení s učitelem dosahuje mnohem lepších výsledků. k hodnotám, kterých bylo v práci [15] dosaženo, se model v této práci přiblížil pouze na trénovacím setu, který ovšem reálnou přesnost modelu nereflektuje. Zajímavější porovnání je s referenční prací [8]. Zde můžeme pozorovat, že kontrastivní učení využité v této prací v naprosté většině případů dosahuje stejných, ba dokonce lepších výsledků než učení s učitelem, které bylo využito prací [8]. Tento experiment

je tedy první v této práci, který dokázal dosáhnout lepších výsledků pomocí kontrastního učení.

Práce [15] uvádí také výsledky pro kontrastivní učení v kombinaci s FlowPic reprezentacemi 64×64 . Tyto výsledky však nejsou prezentovány samostatně pro jednotlivé augmentace a ani konkrétní číselné hodnoty nebyly publikovány. Autoři sice uvádějí, že výsledků bylo dosaženo pomocí augmentací změny RTT a časového posunu, avšak jejich přesné použití není specifikováno, tedy zda byly použité spolu, samostatně a v jakém pořadí. Na základě dostupných grafů dosahují přesnosti přibližně 0,96 na testovací sadě *script* a 0,83 na sadě *human*. Obecně tedy vykazují srovnatelnou úspěšnost na sadě *script* jako tato práce, zatímco na sadě *human* dosahují výrazně lepších výsledků.

Kapitola 6

Závěr

Cílem této bakalářské práce byla replikace výsledků dosažených v referenčních pracích a experimentální ověření modelu také na jiných datových sadách, než které byly použity v původních pracích. Odlišností je také *few shots* přístup, který spočívá v použití pouze malého množství označených dat pro závěrečnou fázi trénování. Tento přístup nebyl v této práci aplikován, jelikož záměrem bylo pracovat s větším objemem dat.

Specifikem navrženého řešení je reprezentace síťového toku ve formě dvourozměrného histogramu nazývaného FlowPic, respektive její zmenšené verze mini FlowPic, která umožnuje aplikovat metody strojového učení původně určené pro klasifikaci obrazových dat. Jedním z takových přístupů je metoda SimCLR. Navržený model neuronové sítě vychází z architektury LeNet-5 a byl upraven tak, aby co nejvěrněji odpovídalo modelům použitým v referenčních studiích, což umožňuje co nejpřesnější porovnání dosažených výsledků. K implementaci neuronové sítě byl použit jazyk Python a knihovna PyTorch, zatímco pro načítání dat sloužila knihovna FlowMind, vyvíjená na Fakultě informačních technologií Vysokého učení technického v Brně.

Na referenční datové sadě Ucdavis-icdm19 se podařilo výsledky úspěšně replikovat a dokonce dosáhnout vyšší přesnosti modelu na testovací sadě *script* v porovnání s výsledky uvedenými v obou referenčních pracích. Model byl následně otestován i na dalších datových sadách, které původní práce buď vůbec nevyužívají, nebo k trénování modelů používají odlišné přístupy učení. Z těchto důvodů nelze výsledky přímo porovnávat, nicméně ze srovnání vyplývá, že kontrastivní učení v kombinaci s reprezentací toku pomocí FlowPic obecně dosahuje horších výsledků než standardní učení s učitelem. Podobně byly otestovány také větší verze mini FlowPic reprezentací a následně porovnány s výsledky dosaženými učením s učitelem.

Model byl také trénován na reálném provozu, který byl zachycen sdružením CESNET a poté testován na zbývajících datových sadách. Pomocí tohoto modelu se podařilo experimentálně dosáhnout nejvyšší přesnosti v této práci.

Literatura

- [1] ACETO, G.; CIUONZO, D.; MONTIERI, A.; PERSICO, V. a PESCAPÉ, A. MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation. In: *2019 4th International Conference on Computing, Communications and Security (ICCCS)* online. Rome, Italy: IEEE, říjen 2019, s. 1–8. ISBN 978-1-7281-0875-9. Dostupné z: <https://doi.org/10.1109/CCCS.2019.8888137>. [cit. 2025-01-10].
- [2] AITKEN, P.; CLAISE, B. a TRAMMELL, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* online. RFC 7011. RFC Editor, září 2013. 76 s. Dostupné z: <https://doi.org/10.17487/RFC7011>. [cit. 2024-11-13].
- [3] BUDUMA, N.; BUDUMA, N. a PAPA, J. *Fundamentals of deep learning*. O'Reilly Media, Inc., 2022. ISBN 978-1-491-92561-4.
- [4] CHEN, T.; KORNBLITH, S.; NOROUZI, M. a HINTON, G. A Simple Framework for Contrastive Learning of Visual Representations. In: III, H. D. a SINGH, A., ed. *Proceedings of the 37th International Conference on Machine Learning* online. PMLR, červenec 2020, sv. 119, s. 1597–1607. Proceedings of Machine Learning Research. Dostupné z: <https://proceedings.mlr.press/v119/chen20j.html>. [cit. 2024-11-02].
- [5] CLAISE, B. *Cisco Systems NetFlow Services Export Version 9* online. RFC 3954. RFC Editor, říjen 2004. 33 s. Dostupné z: <https://doi.org/10.17487/RFC3954>. [cit. 2024-11-13].
- [6] CUNNINGHAM, P.; CORD, M. a DELANY, S. J. *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval* online. 1. vyd. Springer Berlin Heidelberg, 2008. 289 s. ISBN 978-3-540-75171-7. Dostupné z: <https://doi.org/10.1007/978-3-540-75171-7>. [cit. 2024-10-20].
- [7] DONGARE, A.; KHARDE, R.; KACHARE, A. D. et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)* online. Citeseer, 2012, sv. 2, č. 1, s. 189–194. ISSN 2277-3754. Dostupné z: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06>. [cit. 2024-10-26].
- [8] FINAMORE, A.; WANG, C.; KROLIKOWSKI, J.; NAVARRO, J. M.; CHEN, F. et al. Replication: Contrastive Learning and Data Augmentation in Traffic Classification Using a Flowpic Input Representation. In: *Proceedings of the 2023 ACM on Internet Measurement Conference* online. New York, NY, USA: Association for Computing

- Machinery, říjen 2023, s. 36–51. IMC '23. ISBN 9798400703829. Dostupné z: <https://doi.org/10.1145/3618257.3624820>. [cit. 2024-11-02].
- [9] GILL, J. K. Vision Transformers (ViTs). *Xenonstack* online. 3. října 2024. Dostupné z: <https://www.xenonstack.com/blog/vision-transformers>. [cit. 2024-12-12].
- [10] GOODFELLOW, I.; BENGIO, Y. a COURVILLE, A. *Deep Learning* online. MIT Press, 2016. ISBN 02-620-3561-8. Dostupné z: <http://www.deeplearningbook.org>. [cit. 2024-11-01].
- [11] GUARINO, I.; ACETO, G.; CIUONZO, D.; MONTIERI, A.; PERSICO, V. et al. Contextual counters and multimodal Deep Learning for activity-level traffic classification of mobile communication apps during COVID-19 pandemic. *Computer Networks* online, Duben 2022, sv. 219, s. 109452. revidováno 4. 11. 2022. ISSN 1389-1286. Dostupné z: <https://doi.org/10.1016/j.comnet.2022.109452>. [cit. 2025-01-10].
- [12] HENG, Y.; CHANDRASEKHAR, V. a ANDREWS, J. G. UTMobileNetTraffic2021: A Labeled Public Network Traffic Dataset. *IEEE Networking Letters* online. IEEE, Září 2021, sv. 3, č. 3, s. 156–160. ISSN 2576-3156. Dostupné z: <https://doi.org/10.1109/LNET.2021.3098455>. [cit. 2025-01-10].
- [13] HERVE ABDI, B. E. *Neural Networks*. SAGE Publications, 1999. ISBN 07-619-1440-4.
- [14] HOFSTEDE, R.; ČELEDA, P.; TRAMMELL, B.; DRAGO, I.; SADRE, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials* online. IEEE, Květen 2014, sv. 16, č. 4, s. 2037–2064. ISSN 1553-877X. Dostupné z: <https://doi.org/10.1109/COMST.2014.2321898>. [cit. 2024-11-13].
- [15] HOROWICZ, E.; SHAPIRA, T. a SHAVITT, Y. A few shots traffic classification with mini-FlowPic augmentations. In: *Proceedings of the 22nd ACM Internet Measurement Conference* online. New York, NY, USA: Association for Computing Machinery, říjen 2022, s. 647–654. IMC '22. ISBN 9781450392594. Dostupné z: <https://doi.org/10.1145/3517745.3561436>. [cit. 2024-11-27].
- [16] JAISWAL, A.; BABU, A. R.; ZADEH, M. Z.; BANERJEE, D. a MAKEDON, F. A Survey on Contrastive Self-Supervised Learning. *Technologies* online, 2021, sv. 9, č. 1. ISSN 2227-7080. Dostupné z: <https://doi.org/10.3390/technologies9010002>. [cit. 2024-10-28].
- [17] KARAGIANNIS, T.; BRODO, A.; FALOUTSOS, M. a CLAFFY, K. Transport layer identification of P2P traffic. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* online. New York, NY, USA: Association for Computing Machinery, říjen 2004, s. 121–134. IMC '04. ISBN 1581138210. Dostupné z: <https://doi.org/10.1145/1028788.1028804>. [cit. 2024-11-13].
- [18] KETKAR, N. a MOOLAYIL, J. Introduction to PyTorch. In: *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch* online. 2. vyd. Berkeley, CA: Apress, Duben 2021, kap. 2, s. 27–91. ISBN 978-1-4842-5364-9. Dostupné z: https://doi.org/10.1007/978-1-4842-5364-9_2. [cit. 2025-04-04].

- [19] LECUN, Y.; BOTTOU, L.; BENGIO, Y. a HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* online. IEEE, Listopad 1998, sv. 86, č. 11, s. 2278–2324. ISSN 1558-2256. Dostupné z: <https://doi.org/10.1109/5.726791>. [cit. 2025-01-13].
- [20] LEINEN, S. *Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)* online. RFC 3955. RFC Editor, říjen 2004. 23 s. Dostupné z: <https://doi.org/10.17487/RFC3955>. [cit. 2024-11-13].
- [21] MOORE, A. W. a PAPAGIANNAKI, K. Toward the Accurate Identification of Network Applications. In: DOVROLIS, C., ed. *Passive and Active Network Measurement* online. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, sv. 3431, s. 41–54. Lecture Notes in Computer Science. ISBN 978-3-540-31966-5. Dostupné z: https://doi.org/10.1007/978-3-540-31966-5_4. [cit. 2024-11-13].
- [22] NIELSEN, M. A. *Neural Networks and Deep Learning* online. Determination Press, prosinec 2015. Dostupné z: <http://neuralnetworksanddeeplearning.com>. [cit. 2024-10-20].
- [23] REZAEI, S. a LIU, X. How to Achieve High Classification Accuracy with Just a Few Labels: A Semi-supervised Approach Using Sampled Packets. *CoRR* online. 2, Prosinec 2018. revidováno 16. 5. 2020. Dostupné z: <https://doi.org/10.48550/arXiv.1812.09761>. [cit. 2025-01-10].
- [24] SADASIVAN, G.; BROWNLEE, N.; CLAISE, B. a QUITTEK, J. *Architecture for IP Flow Information Export* online. RFC 5470. RFC Editor, březen 2009. 31 s. Dostupné z: <https://doi.org/10.17487/RFC5470>. [cit. 2024-11-13].
- [25] SCHMITT, N. *Measurement, Modeling, and Emulation of Power Consumption of Distributed Systems* online. Würzburg, 2022. Doctoral Thesis. Universität Würzburg. Dostupné z: <https://doi.org/10.25972/OPUS-27658>. [cit. 2024-12-12].
- [26] SHAFIQ, M.; YU, X.; LAGHARI, A. A.; YAO, L.; KARN, N. K. et al. Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms. In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)* online. Chengdu, China: IEEE, říjen 2016, s. 2451–2455. ISBN 978-1-4673-9026-2. Dostupné z: <https://doi.org/10.1109/CompComm.2016.7925139>. [cit. 2024-11-02].
- [27] SHAPIRA, T. a SHAVITT, Y. FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* online. IEEE, Duben 2019, s. 680–687. ISBN 978-1-7281-1878-9. Dostupné z: <https://doi.org/10.1109/INFOWKSHPS.2019.8845315>. [cit. 2024-11-27].
- [28] SHAPIRA, T. a SHAVITT, Y. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Transactions on Network and Service Management* online, Červen 2021, sv. 18, č. 2, s. 1218–1232. ISSN 1932-4537. Dostupné z: <https://doi.org/10.1109/TNSM.2021.3071441>. [cit. 2024-11-27].
- [29] SHEN, M.; YE, K.; LIU, X.; ZHU, L.; KANG, J. et al. Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey. *IEEE*

- Communications Surveys & Tutorials* online. IEEE, Září 2023, sv. 25, č. 1, s. 791–824. ISSN 1553-877X. Dostupné z: <https://doi.org/10.1109/COMST.2022.3208196>. [cit. 2024-11-18].
- [30] TRAMMELL, B. a BOSCHI, E. An introduction to IP flow information export (IPFIX). *IEEE Communications Magazine* online. IEEE, Duben 2011, sv. 49, č. 4, s. 89–95. ISSN 1558-1896. Dostupné z: <https://doi.org/10.1109/MCOM.2011.5741152>. [cit. 2024-11-13].
- [31] VELAN, P.; ČERMÁK, M.; ČELEDA, P. a DRAŠAR, M. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management* online, Červenec 2015, sv. 25, č. 5, s. 355–374. ISSN 1055-7148. Dostupné z: <https://doi.org/10.1002/nem.1901>. [cit. 2024-11-18].
- [32] ZSEBY, T.; BOSCHI, E.; BROWNLEE, N. a CLAISE, B. *IP Flow Information Export (IPFIX) Applicability* online. RFC 5472. RFC Editor, březen 2009. 31 s. Dostupné z: <https://doi.org/10.17487/RFC5472>. [cit. 2024-11-13].
- [33] ZSEBY, T.; CLAISE, B.; QUITTEK, J. a ZANDER, S. *Requirements for IP Flow Information Export (IPFIX)* online. RFC 3917. RFC Editor, říjen 2004. 33 s. Dostupné z: <https://doi.org/10.17487/RFC3917>. [cit. 2024-11-13].