



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ODBĚROVÁ ANALÝZA POMOCÍ METOD AI

POWER ANALYSIS USING AI METHODS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Lukáš Skřivánek

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. Zdeněk Martinásek, Ph.D.

BRNO 2025



Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Lukáš Skřivánek

ID: 247611

Ročník: 3

Akademický rok: 2024/25

NÁZEV TÉMATU:

Odběrová analýza pomocí metod AI

POKYNY PRO VYPRACOVÁNÍ:

Student nastuduje a teoreticky popíše současné modely hlubokého učení pro profilující útoky postranními kanály na kryptografické algoritmy (př. AES), dále analyzuje situaci pro neprofilující útoky. Téma je zaměřeno výhradně na odběrovou analýzu. V praktické části student navrhne a realizuje vlastní AI model pro útok SITM a ověří jeho funkčnost na experimentálním pracovišti VUT (proudové průběhy budou ovlivněny posunem a šumem). Výsledný program realizující útok bude ověřen i pro jiné implementace kryptografického algoritmu, které jsou aktuálně dostupné na pracovišti (FPGA, SW implementace čipová karta aj.). V rámci praktické části student ověří funkčnost celého pracoviště s nástrojem TraceXpert (měření proudových průběhů, vytvoření modelu AI, útok pomocí AI). Dosažené výsledky včetně zdrojových kódů v Pythonu přehledně zpracuje.

DOPORUČENÁ LITERATURA:

- [1] Study of deep learning techniques for side-channel analysis and introduction to ascad database. Emmanuel, Prouff, et al. CoRR, 1-45, 2018.
- [2] Keras. Chollet, Francois and others. 2015. [cit. 2022-09-09] Dostupné z: <https://keras.io>

Termín zadání: 10.2.2025

Termín odevzdání: 3.6.2025

Vedoucí práce: doc. Ing. Zdeněk Martinásek, Ph.D.

prof. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ÚPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato práce se zabývá použitím hlubokého učení v útoku See-In-The-Middle (SITM) na šifru AES-128. Teoretická část se věnuje popisu postranních kanálů, jejich rozdělení a útokům na ně. Dále jsou představeny druhy protiopatření proti těmto útokům, je popsána bloková šifra AES a útok SITM na který se práce zaměřuje. Poslední blok teoretické části představuje a stručně popisuje architektury hlubokého učení, které se využívají k analýze postranních kanálů. Praktická část práce popisuje návrh a implementaci vlastních modelů hlubokého učení. Výsledkem této práce je porovnání různých architektur hlubokého učení pro útok SITM.

KLÍČOVÁ SLOVA

odběrová analýza, analýza postranních kanálů, AES, hluboké učení, SITM, See-In-The-Middle

ABSTRACT

This thesis studies the application of deep learning in the See-In-The-Middle (SITM) attack on the AES-128 cipher. The theoretical part covers side channels, their classification and attacks that exploit them. It also introduces the types of countermeasures used against these attacks, the block cipher AES-128 itself and the SITM attack that is being performed. The last section of the theoretical part introduces the different architectures of deep learning, which are used for side channel analysis. The practical section shows the design and implementation of different custom deep learning models. The result of this work is a comparison of different deep learning architectures in the context of the SITM attack.

KEYWORDS

power analysis, side channel analysis, AES, deep learning, SITM, See-In-The-Middle

SKŘIVÁNEK, Lukáš. *Odběrová analýza pomocí metod AI*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: doc. Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora: Lukáš Skřivánek
VUT ID autora: 247611
Typ práce: Bakalářská práce
Akademický rok: 2024/25
Téma závěrečné práce: Odběrová analýza pomocí metod AI

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno
.....
podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Úvod	11
1 Postranní kanály	12
1.1 Profilující/neprofilující útoky	12
1.2 Korelační analýza proudové spotřeby	13
1.3 Protiopatření	14
1.3.1 Skrývání	14
1.3.2 Maskování	15
1.4 Šifrovací algoritmus AES	15
1.5 Útok See-In-The-Middle	15
2 Hluboké učení v kontextu analýzy postranních kanálů	18
3 Druhy hlubokého učení využívané pro analýzu postranních kanálů	20
3.1 Konvoluční neuronové sítě	20
3.2 Rekurentní neuronové sítě	21
3.3 Architektura sítí Long Short Term Memory	21
3.4 Transformerové sítě	22
3.5 Autoenkodéry	23
4 Praktická část	25
4.1 Výchozí dataset	25
4.2 Návrh modelů hlubokého učení pro výchozí dataset	26
4.2.1 Konvoluční neuronové sítě	26
4.2.2 Transformerová síť	34
4.2.3 Předtrénované modely	36
4.2.4 Autoenkodéry	37
4.2.5 Předtrénovaný extraktor prvků	43
4.3 Útoky na výchozí dataset	45
4.4 Dataset naměřený na zařízení založeném na architektuře ARM	46
4.5 Návrh konvolučních neuronových sítí pro ARM dataset	49
4.6 Útoky na ARM dataset	50
Závěr	54
Literatura	55

Seznam obrázků

1.1	Vizualizace šíření aktivních bytů skrze rundy šifry. AK, SB, SR, MC označují operace AddRoundKey, SubBytes, ShiftRows, MixColumns.	16
1.2	Porovnání náměru spotřeby při šifrování jedné zprávy (modře) a rozdílového vzoru po odečtení náměrů dvou zpráv (oranžově).	17
2.1	Graf vydaných prací o hlubokém učení ve spojení s SCA z článku [14].	18
3.1	Zjednodušená ukázka konvolučního filtru.	21
3.2	Ukázka funkce MaxPooling.	22
3.3	Vizualizace autoenkodéru.	24
4.1	Matice záměn CNN trénované datasetu s průměrováním 1024.	29
4.2	Vizualizace modelů CNN.	30
4.3	Matice záměn pro CNN na datech s průměrováním 4.	31
4.4	Rozdíl dvou zpráv, zprůměrováno z 1024 náměrů.	33
4.5	Spektrogram vytvořený z rozdílu zpráv.	33
4.6	Mel spektrogram vytvořený z rozdílu zpráv.	33
4.7	Porovnání GASF obrázků vygenerovaných z časové řady.	34
4.8	Matice záměn CNN trénované na kombinaci GASF a GADF s přesností 46,67 %.	35
4.9	Matice záměn sítě VGG-16 dotrénované na vlastním datasetu.	37
4.10	UMAP vizualizace dobře separované latentní reprezentace ze stránky [34].	39
4.11	UMAP vizualizace autoenkodéru nalezeného nástrojem Keras Tuner.	40
4.12	Matice záměn pro modely s extraktorem prvků z běžného autoenkodéru.	41
4.13	Matice záměn klasifikačního modelu založeného na běžném autoenkodéru na testovacích datech při použití pouze čtyř tříd.	41
4.14	Matice záměn klasifikačního modelu s použitím β -VAE enkodéru.	43
4.15	UMAP reprezentace latentního prostoru pro různé hodnoty β .	44
4.16	UMAP reprezentace modelu trénovaného kontrastivním učením pouze na čtyřech třídách s průměrováním náměrů 1024.	45
4.17	Porovnání UMAP vizualizací při trénování s různými velikostmi várek.	46
4.18	Výsledky modelu s předtrénovaným extraktorem prvků pomocí kontrastního učení na datasetu s průměrováním čtyři.	47
4.19	Matice záměn útoků modelů z kapitoly 4.2.1 pro různé průměrování náměrů na datasetu bez jitteru.	48
4.20	Matice záměn modelů na ARM datasetech s různým průměrováním před určením bodů zájmu.	50
4.21	CPA na ARM datasetu s průměrováním náměrů 1024.	51

4.22 Porovnání CPA na výchozím a ARM datasetu s průměrováním ná- měrů 4.	52
4.23 Matice záměn pro modely trénované na ARM datasetu s různým prů- měrováním.	53
4.24 Matice záměn útoků na ARM dataset.	53

Seznam tabulek

4.1	Vyhledávací prostor pro nalezení modelu CNN na datasetu bez jitteru.	27
4.2	Nalezené hodnoty pro architektury konvolučních sítí pro různé průměrování dat.	28
4.3	Vyhledávací prostor pro nalezení modelu CNN na datasetu chráněném jitterem.	32
4.4	Přidané parametry pro hledání modelu s LSTM vrstvou.	34
4.5	Přesnosti přetrénovaných modelů.	37
4.6	Vyhledávací prostor pro nalezení modelu autoenkodéru.	38
4.7	Vyhledávací prostor pro nalezení klasifikační části modelu s enkodérem.	40
4.8	Nalezený model pro klasifikaci s použitím enkodéru z β -VAE.	42
4.9	Vyhledávací prostor pro ARM dataset bez zkrácení náměrů.	49
4.10	Vyhledávací prostor pro ARM dataset po zkrácení náměrů.	51

Úvod

Informační technologie hrají neustále větší roli v každodenním životě lidí, to lze pozorovat například v neustále rostoucím použití IoT zařízení. S tím se také stává z informací jeden z hlavních ekonomických zdrojů, které je potřeba chránit. K správné a účinné ochraně informací nám dopomáhá kryptografie. Kryptografické systémy nám umožňují zajistit čtyři základní služby bezpečnosti [1]. Těmito službami jsou důvěrnost, autenticita, integrita a nepopíratelnost.

Aby kryptografické systémy mohly plnit svůj účel v zabezpečování dat, je nutné zajistit, aby byly odolné vůči pokusům o jejich prolomení. Důraz by se měl klást nejen na samotný algoritmus, ale také na jeho implementaci, protože i ta může obsahovat nežádoucí zranitelnosti. Špatná implementace může do okolí vypouštět informace o vnitřních stavech kryptografického zařízení v průběhu šifrování, díky kterým může útočník prolomit tajný klíč používaný k zabezpečení dat. Prostředky skrze které mohou tyto informace opouštět kryptografické zařízení se nazývají postranní kanály [1].

Útok na kryptografické systémy pomocí analýzy jeho okamžité proudové spotřeby byl představen již v 90. letech v článku trojice kryptografů Kocher, Jaffe a Jun. Diferenciální analýza proudové spotřeby [2] se dříve spoléhala především na využití statistických funkcí a odbornou interpretaci výsledků pro extrahování citlivých údajů z naměřených hodnot spotřeb zařízení, což bylo náročné a vyžadovalo hlubokou znalost problematiky. S technologickým pokrokem ale přišly nové způsoby pro interpretaci těchto dat, zejména pomocí hlubokého učení, které proces analýzy automatizuje a usnadňuje. Toto je podpořené neustálým nárůstem výkonu specializované, ale i spotřebitelské výpočetní techniky. Díky tomu je dnes možné aplikovat metody hlubokého učení i na běžně dostupném hardwaru, čímž se zvyšuje jejich přístupnost. Použití hlubokého učení zvýšilo úspěšnost a účinnost útoků na postranní kanály. To také snížilo jejich závislost na znalostech analytika, který je provádí [3].

Hlavní cíle práce se dají shrnout do těchto bodů:

- popsat architektury hlubokého učení používané k útokům na postranní kanály,
- navrhnout vícero architektur modelů hlubokého učení pro útok See-In-The-Middle,
- provést simulované útoky pomocí vytvořených modelů pro neupravená data a data s posunem,
- porovnat a vyhodnotit dosažené výsledky.

1 Postranní kanály

Jakákoliv nežádoucí výměna informací mezi kryptografickým zařízením a jeho okolím se nazývá postranním kanálem. Tyto kanály můžeme dělit podle typu informací, které skrz ně prosakují do okolí. Jeden druh dělení se řídí podle fyzikální veličiny, kterou je možné díky nim změřit a následně využít ke kompromitaci tajných informací. Mezi hlavní druhy postranních kanálů podle fyzikálních veličin patří například:

- výkonový postranní kanál,
- elektromagnetický postranní kanál,
- časový postranní kanál,
- optický postranní kanál.

Pokud útočník chce využít postranního kanálu k útoku na kryptografické zařízení, musí z něj data naměřit, zpracovat, a poté interpretovat. Celý tento proces se nazývá analýzou daného kanálu. Dnes nejpoužívanější a nejvíce prozkoumaná technika je analýza proudového kanálu. Ta byla představena v roce 1999 Paulem Kocherem [2]. Jak její název napovídá, zkoumá proudový odběr kryptografického aparátu za účelem získání soukromého klíče. Okamžitý proudový odběr zařízení je přímo závislý na zpracovávaných datech a operacích s nimi spojených. Zpracovávaná data se skládají z nešifrovaných dat (otevřeného textu určeného k šifrování, neboli plaintextu) a tajné hodnoty (např. část klíče)[4]. Díky této závislosti proudové spotřeby a zpracovávaných dat je útočník schopný pomocí naměřených hodnot zjistit tajný klíč, nebo zúžit možnosti pro jeho hledání.

1.1 Profilující/neprofilující útoky

Jedno z možných dělení útoků je na profilující a neprofilující. Neprofilující útoky jsou jednodušší. Útočník u nich nepotřebuje mít předchozí znalost o daném kryptografickém zařízení, což je dělá méně časově náročné. Spoléhají se na statistické techniky a obecné modely k extrahování klíče. Nejběžněji používané z těchto modelů jsou například model Hammingovy váhy, ten zkoumá kolik bitů v hodnotě je nastaveno na hodnotu 1, a model Hammingovy vzdálenosti, zkoumá kolik bitů je změněných mezi dvěma hodnotami [5].

Profilující útoky obsahují dvě fáze. Fázi profilování a fázi útoku. Jejich koncept byl představen kryptografy Fahn a Pearsonem [6] kde navrhli inferenční analýzu spotřeby. Před rozšířením použití hlubokého učení v oblasti profilujících útoků byly nejvíce používané útoky pomocí šablon. Ve fázi profilování potřebuje útočník přístup k zařízení, na které útočí, nebo k zařízení, které je jemu identické, nebo alespoň podobné. Díky přístupu k takovému zařízení má útočník možnost zkoumat hodnoty

mezivýsledků výpočtů pro různá vstupní data. Po zvolení vhodné operace pro sledování změn spotřeb (například výstup operace `AddRoundKey` v šifře AES) se naměří jejich hodnoty pro dvojice dat vstupního textu a šifrovacího klíče. Vstupní text zůstává mezi měřeními stejný a mění se tajný klíč. Změnami vstupů lze pozorovat změny v proudové spotřebě, která je na nich přímo závislá. Útočník si tímto vytvoří modely proudové spotřeby pro jednotlivé klíče. Ty se nazývají šablony. Z těchto šablon se poté dá vytvořit statistický model, který je schopný propojit proudové spotřeby s odpovídajícími hodnotami vnitřních stavů zařízení. Ve fázi útoku útočník pošle jemu známý otevřený text do kryptografického zařízení a měří jeho proudovou spotřebu. Použitím dříve vytvořeného statistického modelu poté získá míry podobnosti naměřené spotřeby k jednotlivým šablonám. Díky tomuto je poté schopný rekonstruovat tajný klíč, nebo jeho části [1].

1.2 Korelační analýza proudové spotřeby

Correlation power analysis (CPA), je druh diferenciální proudové analýzy (DPA). Jeho cílem je odhalit tajný klíč kryptografického zařízení na základě velkého množství naměřených proudových průběhů při šifrování různých zpráv. Jelikož CPA nevyžaduje detailní znalost o zařízení na které se útočí, je jednou z nejpopulárnějších variant proudové analýzy [7]. Skládá se z pěti kroků podle [8]:

Krok 1: Výběr mezivýsledku zkoumaného algoritmu

Prvním krokem je výběr mezivýsledku zkoumaného kryptografického algoritmu. Tento mezivýsledek musí být funkcí $f(d, k)$, kde d jsou známá nekonstantní data a k je malá část klíče. Ve většině útoků je d buď zpráva v otevřeném textu, nebo výsledná zašifrovaná zpráva.

Krok 2: Naměření proudové spotřeby

Zařízení je sledováno během šifrování či dešifrování D datových bloků. Útočník zaznamenává proudovou spotřebu a přiřazuje ji odpovídajícím datovým hodnotám d . Délka náměru je označena T . Jakmile útočník naměří proudové spotřeby pro všechny datové bloky D , je možné z nich sestavit matici $\mathbf{T}$ o velikosti $D \times T$. Náměry by měly být zarovnané stejně, aby jednotlivé sloupce $\mathbf{T}$ odpovídaly stejným operacím. Pokud je otevřený text zprávy známý, útočník může nastavit spoušť osciloskopu na moment odeslání zprávy z počítače na kryptografické zařízení, čímž je schopný naměřit proudovou spotřebu jen pro klíčové momenty šifrování.

Krok 3: Výpočet hypotetických hodnot mezivýsledků

Útočník vypočítá teoretické hodnoty mezivýsledků pro všechny možné hodnoty klíče k , čímž získá matici $\mathbf{V}$ o rozměrech $D \times K$. Jeden ze sloupců této matice odpovídá skutečným mezivýsledkům na sledovaném kryptografickém zařízení. Cílem je poté zjistit, o který sloupec se jedná, pro získání odpovídajícího klíče k .

Krok 4: Mapování hodnot mezivýsledků na hodnoty proudové spotřeby

Matice hypotetických výsledků $\mathbf{V}$ je převedena na matici $\mathbf{H}$, která představuje předpokládané hodnoty proudové spotřeby. Toto je provedeno pomocí modelů jako jsou Hammingova váha, nebo Hammingova vzdálenost.

Krok 5: Porovnání hypotetické proudové spotřeby s hodnotami z naměřené spotřeby

Sloupce matice předpokládaných proudových spotřeb $\mathbf{H}$ jsou porovnány s odpovídajícími sloupci matice $\mathbf{T}$ pomocí Pearsonova korelačního koeficientu. Nejvyšší hodnota v korelační matici $\mathbf{R}$ ukazuje, který sloupec matice $\mathbf{V}$ odpovídá skutečným operacím zařízení, čímž útočník získá správný klíč k .

1.3 Protiopatření

Obrana proti proudové analýze spočívá v zajištění nezávislosti proudové spotřeby na zpracovávaných datech. Dva hlavní přístupy jsou skrývání a maskování.

1.3.1 Skrývání

Skrývání se zaměřuje na změnu vzorců chování zařízení tak, aby byly pro útočníka neprůkazné, nebo aby analýza vyžadovala značné množství dat a výpočetního výkonu. Toho lze dosáhnout přidáním redundantních výpočtů, které přidávají náhodný, nebo konstantní šum do měření. Zvýšení šumu v signálu lze také dosáhnout pomocí paralelizování výpočtů. Ideální implementace skrývání by zajistila, aby okamžitá spotřeba zařízení byla totožná v jakýkoliv okamžik šifrování, tedy aby byla nezávislá na zpracovávaných datech. K tomu nám může dopomoci spouštění redundantních procesů při šifrování, které „doplň“ spotřebu primárního výpočtu do očekávané výše [9].

1.3.2 Maskování

Maskování znáhodňuje mezivýsledky výpočtů přidáním masky. Maska je náhodně volená hodnota, která je jiná pro každé spuštění algoritmu. Přidat se může k otevřenému textu, nebo tajnému klíči. Výsledkem šifrování je poté zašifrovaný text a jeho maska. Bez znalosti masky útočník není schopný získat tajný klíč pouze z informací, které unikly postranním kanálem. Tyto implementace také vyžadují schéma, podle kterého lze masku ze zašifrovaného textu odstranit pro získání výsledného kryptogramu [1]. Používané masky mohou být předem vygenerované, nebo se mohou vytvářet pro každé šifrování nové, aby se zajistila náhodnost napříč operacemi. Maskování je výhodné, protože nevyžaduje změny hardwaru zařízení které ho používá [9].

1.4 Šifrovací algoritmus AES

Advanced Encryption Standard je symetrická bloková šifra, která pracuje s bloky o délce 128 bitů. Existují různé varianty, které se liší délkou hlavního klíče a počtem rund, ve kterých se tvoří šifrovaný text. Tato práce se zaměřuje na variantu AES-128, která se skládá z 10 rund a její hlavní klíč je 128 bitů dlouhý. Pro ilustraci je uvažováno o každém mezistavu šifry jako síti o rozměrech 4×4 , kde každá buňka představuje jeden byte. Tato síť je zobrazena na obr. 1.1. Bajty na pozici (0,0), (1,1), (2,2) a (3,3) nazýváme hlavní diagonálou mezistavu.

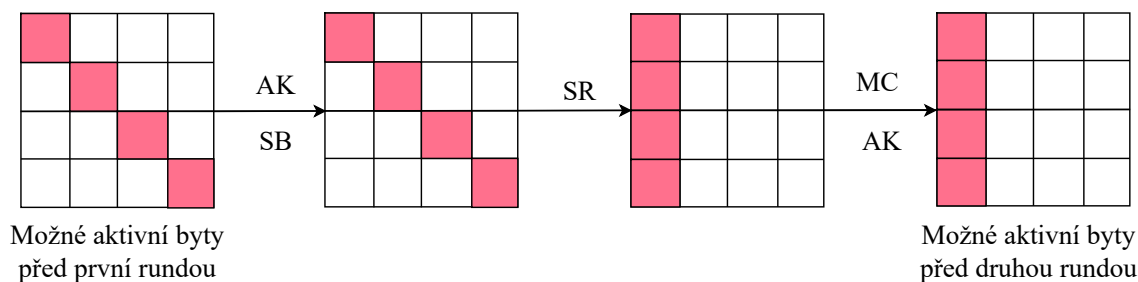
S hlavním klíčem se pracuje pouze na začátku šifrování, kdy je přidán operací XOR k šifrovanému textu. V následujících rundách operací se pracuje s rundovými klíči, které jsou z hlavního klíče odvozené. Prvních 9 rund se skládá z následujících operací:

- SubBytes - aplikuje AES Sbox.
- ShiftRows - posune každý řádek mezistavu šifry doleva (první zůstává, druhý posune o jednu pozici, třetí o dvě a čtvrtý o tři).
- MixColumns - vynásobí každý sloupec pomocí předem dané matice.
- AddRoundKey - bitová XOR operace mezi rundovým klíčem a mezistavem šifry.

Poslední runda neobsahuje krok MixColumns. Během dešifrování se prochází stejným postupem, pouze s inverzními variantami jednotlivých operací (například inverzní ShiftRows, InvShiftRows).

1.5 Útok See-In-The-Middle

U blokových šifer, jako je například AES, lze pozorovat mezivýsledky jednotlivých operací, podle kterých je možné extrahovat tajný klíč. See-In-The-Middle (SITM)



Obr. 1.1: Vizualizace šíření aktivních bytů skrze rundy šifry.

AK, SB, SR, MC označují operace AddRoundKey, SubBytes, ShiftRows, MixColumns.

Útok představený v [10] se tohoto snaží dosáhnout pomocí párů zpráv v otevřeném textu, které vedou ke specifickému rozdílovému vzoru. Konkrétněji pomocí zvolení dvou zpráv s malým rozdílovým vzorem, který by mohl potenciálně konvergovat do menšího rozdílového vzoru napříč mezivýsledky šifrování. Rozdílový vzor je rozdíl mezistavů dvou různých zpráv v daném mezistavu (například po třetí rundě šifrování), pro ilustraci je zobrazený na obrázku 1.2. Autoři používají výraz „malý“ ve spojení s rozdílovým vzorem jako vyjádření nízkého počtu aktivních (nenulových) buněk. Tedy hledané vstupy mají mít podobný jeden z mezivýsledků. Útočník je po SITM útoku schopný určit část hlavního klíče. Poté se buď může pokusit hlavní klíč uhodnout, jelikož značně redukoval jeho kandidáty, nebo útok opakovat, dokud není celý klíč odhalen.

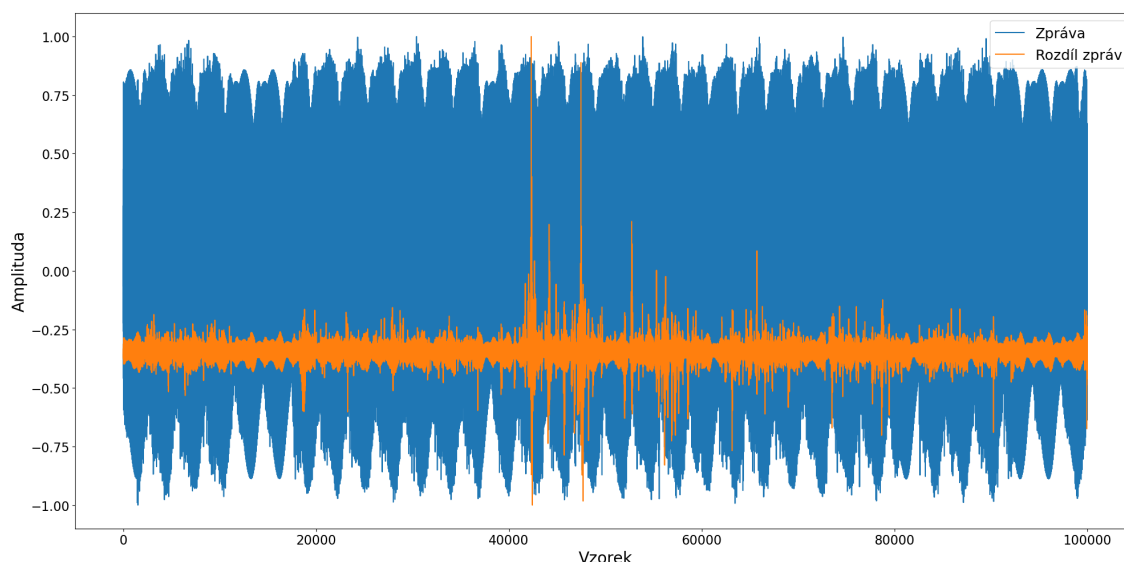
Celý proces se dá rozdělit do následujících kroků:

Krok 1: Hledání cílového rozdílového vzoru. Je hledán podle následujících kroků:

- Je zvolen malý rozdílový vzor v jednom z mezistavů šifrování.
- Z tohoto malého vzoru je prováděna zpětná propagace skrze jednotlivé operace šifrování a je zvažováno, jak mohl tento vzor vzniknout z rozdílů vstupních zpráv. Pokračuje se, dokud pravděpodobnost rozdílu neklesne pod nejnížší přijatelnou prahovou hodnotu.
- Ze stejného rozdílového vzoru je prováděna propagace směrem dopředu do bodu, kdy je stále možné rozpoznat, že výsledek vznikl z původního malého rozdílového vzoru.

Prahová hodnota přijatelnosti závisí na maximálním počtu otevřených textů, které mohou být vyzkoušeny a u nichž mohou být naměřeny hodnoty spotřeby při jejich zpracování.

Krok 2: Hledání páru vstupů s daným vstupním rozdílovým vzorem, které vyústí v hledaný cílový rozdílový vzor mezivýsledku. Tento proces



Obr. 1.2: Porovnání náměru spotřeby při šifrování jedné zprávy (modře) a rozdílového vzoru po odečtení náměrů dvou zpráv (oranžově).

probíhá šifrováním kandidátů na cílovém zařízení. Za předpokladu že pravděpodobnost výskytu chtěného vzoru v mezivýsledcích je 2^{-x} , kde x závisí na prahové hodnotě přijatelnosti. Tedy k nalezení jednoho páru otevřených textů které budou konvergovat do chtěného cílového vzoru je potřeba vyzkoušet přibližně $2^{0,5x+0,5}$ vstupů. Zobrazení možného šíření aktivních bytů skrz rundy je k vidění na obr. 1.1.

Krok 3: Uhodnutí klíče (částečné). Tento krok se liší pro různé šifry.

Krok 4: Opakování kroků 1-3, dokud není znám celý rundový, nebo i hlavní klíč.

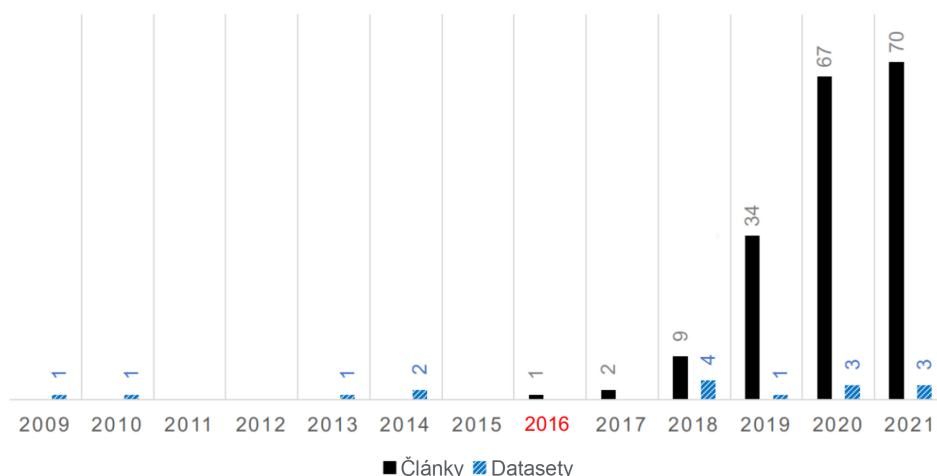
Tato práce je zaměřená na SITM útok na šifru AES-128, při kterém jsou sledovány mezivýsledky operace SubBytes třetí rundy šifrování.

2 Hluboké učení v kontextu analýzy postranních kanálů

Tradiční přístupy k proudové analýze se spoléhají na statistické metody a expertízu analytika. Kvůli přítomnosti šumu je běžným postupem naměřená data průměrovat a jinak předzpracovávat, aby byla jednodušeji interpretovatelná. Modely hlubokého učení jsou schopné najít hledané vzory z náměrů spotřeby zařízení s minimálním předzpracováním, což výrazně zjednodušuje celý proces. Díky tomu je také zapotřebí nižší míra znalostí v tomto oboru, bez které by se jinak extrakce informací neobešla [3, 14, 11].

Nejen že použití hlubokého učení dělá proudovou analýzu snazší, ale tyto metody také dosahují lepších výsledků v přítomnosti protiopatření a vysokého množství šumu oproti běžným šablonovým útokům [12, 13]. Například konvoluční neuronové sítě prokázaly časovou invariabilitu. Jsou tedy do určité míry odolné vůči skrývání pomocí znáhodnění času provedených instrukcí [11].

První použití hlubokého učení pro analýzu postranních kanálů bylo v práci [12] z roku 2016. V [14] autoři uvádí, že mezi roky 2016-2021 vzniklo 183 prací, které zkoumají použití hlubokého učení pro analýzu postranních kanálů. Toto je zobrazené grafem na obr. 2.1. Je tedy zjevné, že je tato problematika v dnešní době vysoce relevantní.



Obr. 2.1: Graf vydaných prací o hlubokém učení ve spojení s SCA z článku [14].

Strojové učení probíhá dvěma hlavními způsoby:

- učení s učitelem,
- učení bez učitele.

Při učení s učitelem je nutné mít trénovací data, která se skládají z vstupních informací a jejich označení. Model se tedy učí na datech, u kterých má jednoznačně označené, o jaká data se jedná. Tím je schopný vyvodit, které vzory a závislosti jednotlivých bodů souvisí s klasifikací vzorku. Naučený model je poté použit k analýze náměrů ze zařízení na které je útok mířen. Přístup ke známým datům ve fázi učení je analogický k profilující fázi profilujících útoků, kde má útočník možnost si vytvořit databázi proudových spotřeb pro známá data, které poté využije k útoku [3].

Podobné spojení lze najít i u učení bez učitele a neprofilujících útoků. Modelu jsou dána data, která nejsou spojena s jejich cílovým označením. Předpokládá se, že útočník tedy nemá přístup k zařízení na které je poté vykonáván útok pro vytvoření označené skupiny dat pro učení. Tyto útoky jsou složitější, protože vyžadují extrakci informací vztahujících se ke klíči přímo z naměřených hodnot [3].

3 Druhy hlubokého učení využívané pro analýzu postranních kanálů

Podle autorů práce [14] jsou nejčastěji používanými architekturami pro analýzu postranních kanálů konvoluční neuronové sítě a vícevrstvé perceptrony. Použití konvolučních neuronových sítí může být motivováno jejich odolností vůči časovému posunu náměrů, což snižuje nároky na zarovnání náměrů [15]. Krom těchto dvou architektur existují i další. Některé z nich jsou v této části krátce popsány.

3.1 Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) jsou specifickým druhem neuronových sítí, které se typicky skládají ze tří druhů vrstev:

1. Konvoluční vrstva.
2. Poolingová (podvzorkovací) vrstva.
3. Plně propojená (fully connected, nebo dense) vrstva.

Konvoluční vrstvy obsahují filtry (jádra), které extrahují vlastnosti ze vstupních dat. Filtr je matice vah, která se posouvá po vstupních datech s definovaným krokem (stride) a vytváří takzvanou mapu vlastností (feature map). Tento proces je zobrazený na obr. 3.1. Umožňuje detekci různých vzorů a složitějších souvislostí v datech. Po konvoluci se obvykle aplikuje aktivační funkce, jako je například ReLU (Rectified Linear Unit), která zavádí nelinearitu do modelu. Po dokončení průchodu daty se síť prochází pozpátku a v každém filtru se upravují váhy. Tyto úpravy se snaží minimalizovat ztrátovou (loss) funkci, která hodnotí, jak daleko od chtěného výsledku je aktuální stav výstupu [16]. Tento proces se nazývá zpětnou propagací (backpropagation).

Poolingová vrstva běžně následuje po konvoluční vrstvě a slouží ke snížení rozměru dat, což vede ke snížení výpočetní náročnosti a omezení přeučení modelu. Toto stejně jako u předešlé vrstvy probíhá pomocí posouvající se matice, která tentokrát nemá aktivační funkci. Nejčastěji se používají metody MaxPooling a AveragePooling. MaxPooling vybírá maximální hodnotu z každé oblasti filtru, zatímco AveragePooling počítá průměrnou hodnotu [16]. Ilustrační příklad MaxPooling je k vidění na obrázku 3.2.

Plně propojená vrstva se nachází na konci sítě. Každý neuron v této vrstvě je propojen se všemi neurony předchozí vrstvy. Tato vrstva kombinuje extrahované vlastnosti a provádí finální klasifikaci nebo regresi. Výstup z této vrstvy často prochází aktivační funkcí, jako je SoftMax, která převádí hodnoty na pravděpodobnostní rozdělení mezi jednotlivé třídy výstupních dat [16].

Konvoluce s filtrem o velikosti 2x2 a krokem 1

in ₁₁	in ₁₂	in ₁₃
in ₂₁	in ₂₂	in ₂₃
in ₃₁	in ₃₂	in ₃₃

 $\ast$

k ₁₁	k ₁₂
k ₂₁	k ₂₂

 $=$

o ₁₁	o ₁₂
o ₂₁	o ₂₂

$$\begin{aligned} o_{11} &= in_{11}k_{11} + in_{12}k_{12} + in_{21}k_{21} + in_{22}k_{22} \\ o_{12} &= in_{12}k_{11} + in_{13}k_{12} + in_{22}k_{21} + in_{23}k_{22} \\ o_{21} &= in_{21}k_{11} + in_{22}k_{12} + in_{31}k_{21} + in_{32}k_{22} \\ o_{22} &= in_{22}k_{11} + in_{23}k_{12} + in_{32}k_{21} + in_{33}k_{22} \end{aligned}$$

5	2	7
3	10	1
6	4	3

 $\ast$

1	2
2	1

 $=$

25	37
39	23

Vstupní data Konvoluční filtr Mapa vlastností

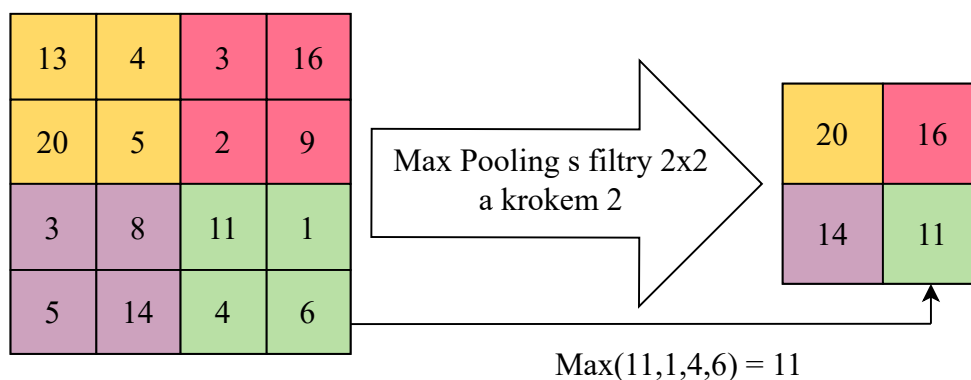
Obr. 3.1: Zjednodušená ukázka konvolučního filtru.

3.2 Rekurentní neuronové síť

Tento typ sítě byl navržen pro zpracování sekvenčních dat, kde je důležité pořadí prvků sekvence a závislosti mezi nimi. Obsahují zpětné vazby, díky kterým jsou schopné uchovávat informace z předchozích časových kroků. Vrstva nad každým vstupem provede konvoluci a její výsledek sloučí se vstupem následující vrstvy [17]. Díky tomuto je výstup celé sítě závislý na předchozích operacích. Jsou široce používané ve zpracování přirozeného jazyka. Je z nich odvozený model LSTM. V kontextu analýzy postranních kanálů jsou rekurentní neuronové sítě (RNN) užitečné, protože hledané úniky informací se nachází rozprostřené přes vícero bodů [12].

3.3 Architektura sítě Long Short Term Memory

Long Short Term Memory, neboli LSTM, je speciální typ rekurentních neuronových sítí, navržený pro učení dlouhodobých závislostí v sekvenčních datech. Běžně RNN trpí problémem mizení gradientu, který nastává v procesu zpětné propagace chyby



Obr. 3.2: Ukázka funkce MaxPooling.

skrze jednotlivé vrstvy sítě. Pokud jsou data moc dlouhá, nebo má síť velký počet vrstev, může se hodnota ztrátové funkce v pozdějších fázích snížit až k nule. Hodnoty vah dřívějších vrstev sítě jsou tedy málo aktualizovány, což může zpomalit, nebo až zastavit proces učení. LSTM tento problém řeší pomocí paměťových buněk, které uchovávají informace napříč časovými kroky a jsou díky tomu schopné omezit ztrátu gradientu, což jim pomáhá v jejich schopnosti učit se na dlouhých sekvencích dat. S paměťovými buňkami využívají také systému bran (forget, input a output gate). Tyto brány mění tok dat skrz síť a vytváří systém díky kterému je síť schopná uchovávat, nebo zapomínat informace podle potřeby [17].

3.4 Transformerové sítě

Stejně jako ostatní architektury se transformerové sítě skládají z několika na sebe navazujících vrstev. Co je odlišuje od jiných architektur je jejich schopnost zachytit globální závislosti mezi prvky sekvence.

Každá transformerová vrstva se skládá ze dvou vrstev [18]:

1. self-attention vrstva,
2. feed-forward vrstva.

Self-attention mechanismus slouží k zachycení vztahů mezi jednotlivými prvky vstupní sekvence. Každý prvek vstupní sekvence je zpracován na základě své důležitosti vůči ostatním prvkům. Díky tomuto je poté síť schopna zachytit spojitosti mezi vzdálenými prvky. Tato vrstva má také schopnost kombinovat uniklá data z více bodů zájmu, což vede k vyššímu poměru signálu k šumu na jejím výstupu. K transformaci vstupu využívá váženého součtu vstupních prvků, kde váhy vyjadřují míru „pozornosti“, kterou každému prvku věnuje. Tyto váhy se získávají výpočtem pomocí matice dotazů (W_q), klíčů (W_k) a hodnot (W_v) [18].

Feed-forward vrstva transformuje každý prvek pomocí plně propojené neuronové sítě. Tímto zvyšuje nelineární schopnosti modelu a pomáhá identifikovat komplexní vzory. Výstup celé transformerové vrstvy je součtem výstupu self-attention vrstvy a původní sekvence, který je poté zpracován feed-forward vrstvou [18].

Jedna self-attention operace se nazývá jednou „hlavou pozornosti“. V praxi se těchto hlav používá více. Výsledkem je takzvaný Multihead Self-attention mechanismus. Každá hlava má vlastní hodnoty W_q , W_k a W_v . Všechny pracují paralelně na stejných vstupních datech a výstupem je sloučení všech jejich výsledků. Jelikož je výpočetní i paměťová složitost této implementace kvadratická, její aplikace na dlouhá data může být problémová [19].

U běžných architektur hlubokého učení je běžné inicializovat rychlost učení (jak moc velké změny vah neuronů se mohou provádět) na vysokou hodnotu, která je následně během učení snižována. Pro efektivní učení transformerových sítí je nutné, aby na začátku učení bylo období „zahřátí“ (warmup period). Rychlost učení začíná na malých hodnotách a v průběhu zahřívání se pomalu zvyšuje. Pokud je výchozí hodnota rychlosti učení příliš vysoká, síť nejsou schopné dosáhnout dobrých výsledků. Naopak, když je počáteční rychlost učení příliš nízká, proces trénování trvá zbytečně dlouho [20]. Proto je důležité zvolit správný průběh doby zahřívání.

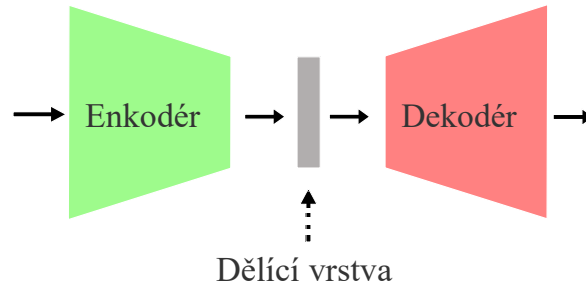
3.5 Autoenkodéry

Autoenkodér je speciální druh neuronové sítě využívaný pro extrakci prvků, vlastností dat a redukce dimenzionality. Skládá se ze tří částí:

- z enkodéru,
- dělicí vrstvy,
- dekodéru.

Enkodér slouží k redukci dimenzionality vstupních dat [14], kdy současně identifikuje důležité vlastnosti dat. Mapuje vstupní data do latentního prostoru. Výstup enkodéru je uložen v dělicí vrstvě, která obsahuje latentní reprezentaci dat. Vstupem dekodéru je obsah dělicí vrstvy, který se poté snaží vyjádřit jako data o stejné dimenzionalitě, jako byla data na vstupu enkodéru [14].

Jedním z hlavních použití autoenkodérů je vytvoření kompaktní reprezentace dat za účelem snížení jejich dimenzionality, nebo odstranění šumu. V tomto případě je cílem autoenkodérů minimalizovat rozdíl mezi vstupními daty a daty rekonstruovanými dekodérem. Tímto procesem se dají efektivně zachytit nejdůležitější rysy vstupních dat [14]. Další časté použití této architektury je při hledání anomálií v časových datech, jako například detekce náhlých změn a extrémních hodnot. Také jdou použít k predikci časových dat, jako je odhad poptávky v daný čas. V kontextu



Obr. 3.3: Vizualizace autoenkodéru.

SCA byly také použity ke klasifikaci, kde autoři přidali na konec architektury vrstvu s aktivační funkcí softmax jako výstup [12].

Existuje více architektur, které jsou od autoenkodérů odvozené, jako například variační autoenkodéry (VAE), které byly představeny Diedrikem P. Kingmou a kolektivem v práci [21]. Ty se dokážou naučit kontinuitu latentního prostoru, díky čemuž se dají použít ke generování nových dat. Jsou také vhodné, pokud není klíčová rekonstrukce dat, ale záleží na dobré separaci tříd v latentní reprezentaci.

Variační autoenkodéry

Variační autoenkodéry (VAE) byly představeny v práci [21]. Na rozdíl od běžných autoenkodérů, které mapují vstupní data do nízkodimenzionálního prostoru jako diskrétní body, variační autoenkodéry předpokládají, že latentní proměnné se řídí pravděpodobnostním rozdělením. Místo mapování vstupních dat na fixní bod, mapuje vstup x na vektor z latentní reprezentace. Díky tomuto jsou schopné vytvářet data z libovolných bodů latentního prostoru. Pokus o rekonstrukci libovolného bodu z latentního prostoru u běžného autoenkodéru vede k nesmyslným datům, zatímco u variačních autoenkodérů vznikne kombinace nejbližších prvků. Jejich ztrátové funkce se skládají ze dvou částí:

- ztrátové funkce rekonstrukce,
- ztrátové funkce KL (Kullback-Leibler) divergence.

Rekonstrukční ztrátová funkce slouží stejně jako u běžných autoenkodérů k optimalizaci samotné rekonstrukce. KL divergence slouží k regularizaci latentního prostoru a zajišťuje, aby rozložení v něm bylo blízko Gaussovskému. Podporuje plynulost a kontinuitu v latentním prostoru.

β -VAE přidávají při výpočtu celkové ztrátové funkce ke KL divergenci hodnotu β , která podle své velikosti může přidávat váhu KL divergenci v porovnání s rekonstrukční ztrátovou funkcí. S větší hodnotou β by měly být jednotlivé třídy dat v latentní reprezentaci lépe separované, za cenu méně přesné rekonstrukce [22].

4 Praktická část

V této kapitole jsou prezentovány výsledky vyzkoušených přístupů k SITM útoku za pomoci hlubokého učení. Modely konvolučních sítí, včetně použitých datasetů k simulování útoku, jsou dostupné ke stažení ¹.

4.1 Výchozí dataset

Dataset se skládá z 15 000 náměrů jednotlivých průběhů šifrování zpráv, které jsou naměřené na zařízení založeném na mikročipu s 8bitovou architekturou. Z těchto náměrů je vypočítáno 7 500 rozdílů potřebných pro SITM útok. Pro usnadnění hledání vhodných dvojic zpráv pro vytvoření diferencí s kolizí byly místo generování otevřených textů k šifrování generovány mezistavy po první rundě šifrování, které se lišily pouze v jednom byte v prvním sloupci. Poté byly tyto mezistavy dešifrovány inverzními operacemi k první rundě, pomocí čehož byly vytvořeny chtěné páry otevřeného textu. Byly vygenerovány i páry zpráv, které se po první rundě liší ve více než jednom byte, aby dataset obsahoval i případy, které nevyhovují kritériím pro SITM útok. Toto může zvýšit robustnost výsledného modelu, jelikož se bude schopný naučit i charakteristiky nevhodných dat, která se při běžném útoku v náměrech také objeví. Diference těchto párů jsou poté rozděleny do pěti tříd.

- **Třída 1-4:** Každá z těchto tříd obsahuje vzorky s jedním aktivním sloupcem před třetí rundou. Tyto třídy tedy tříd obsahují hledané kolize. Skládají se z 1 000 vzorků.
- **Třída 5:** Obsahuje difference, ve kterých je po druhé rundě aktivní více než jeden byte, tedy zprávy neobsahují kolizi. Tyto případy jsou běžnější, a proto je také tato třída obsáhlejší. Čítá 4 500 vzorků.

Původní náměry byly 500 000 bodů dlouhé. Toto je ale nepraktické z důvodu paměťové náročnosti modelů, které by měly pracovat s takto dlouhými daty. Také by byly zapotřebí složitější architektury modelu, které by byly schopné pracovat s takto dlouhými daty, což by dále zvětšilo výpočetní náročnost. Proto byla zvolena část dlouhá 30 000 bodů, která už je pro trénování vhodnější. Tento úsek byl zvolený pomocí korelační analýzy (CPA), díky které bylo možné nalézt v proudových průbězích sledovanou operaci SubBytes (nebo také Sbox) ve třetí rundě. V jiných pracích je časté používat kratší části náměrů. Například v [23] autoři zvolili části náměrů o délce 700 bodů.

¹Výsledné modely a datasety jsou k dispozici na adrese <https://github.com/LukSkriv/DL-SITM-bachelors-thesis/tree/main?tab=readme-ov-file>.

Vzhledem k nevyváženosti velikostí tříd byly při trénování třídám přiřazeny váhy, které zohledňují jejich podíl v datasetu ve snaze, aby žádná z nich nebyla při trénování opomíjena.

Pro snížení dopadu šumu a zvýraznění klíčových částí úniků je běžným postupem náměry průběhů šifrování každého otevřeného textu průměrovat z vícero náměrů. Neprůměrovaná data často způsobují sítím problémy, jelikož je pro ně kvůli šumu složité v náměrech nalézt důležité vzory. Datasety pro tuto práci byly naměřeny s různými úrovněmi průměrování (0, 2, 4 a 1024 náměrů). Toto průměrování bylo provedeno přímo v osciloskopu. Data s průměrováním z 1024 náměrů jsou jednoduše interpretovatelná i pouhým okem, ale jak je zmíněné v práci [3], vysoké průměrování může odstranit malé detaily, které mohou být klíčové pro zachycení souvislostí trénovaným modelem.

Data byla rozdělena na tři části. Trénovací, validační a testovací dataset, a to v poměru 60:20:20. To umožňuje ověřit schopnost modelů generalizovat naučené vzory na nová data, která nebyla použita při trénování.

4.2 Návrh modelů hlubokého učení pro výchozí dataset

Tato sekce se zaměřuje na detaily implementací trénovaných modelů. Všechny modely byly hodnoceny podle metriky přesnosti. K hledání vhodné struktury modelů a jejich hyperparametrů byl použit nástroj Keras Tuner [24]. Tento nástroj slouží k automatizaci procesu hledání použitých hyperparametrů. Po definování parametrů jako jsou počet vrstev, počet filtrů v konvoluční vrstvě, velikost filtrů a další, nástroj sám vyzkouší kombinace těchto možností a určí z nich tu nejvhodnější.

4.2.1 Konvoluční neuronové sítě

Jako výchozí bod pro návrh CNN (Convolutional Neural Network) byl zvolen model VGG-16. Tato volba vychází z práce [23], kde autoři ukázali, že tento model je vhodným základem pro práce spojené s analýzou postranních kanálů. Podle autorů oproti jiným veřejným modelům, jako jsou například ResNet nebo Inception-v3, dosahuje vyšší úspěšnosti oproti maskovaným implementacím šifer.

Dataset bez jitteru

Zvolený vyhledávací prostor pro Keras Tuner je znázorněn v tab. 4.1. Po vzoru sítě VGG-16 byla po každé konvoluční vrstvě přidána average pooling vrstva, která pokaždé snížila dimenzionalitu dat. Počet konvolučních filtrů byl v každé následující

Tab. 4.1: Vyhledávací prostor pro nalezení modelu CNN na datasetu bez jitteru.

Hyperparametr	Hodnoty k prozkoumání
Konvoluční vrstvy	[4; 5; 6; 7]
Konvoluční filtry	[4; 8; 12; 16]
Velikost konvolučního filtru	[4; 5; 6; 8]
Aktivační funkce	[ReLU; SeLU]
Plně propojené vrstvy	[1; 2; 3]
Neurony plně propojené vrstvy	[256; 512; 1024]
Velikost pooling matice	[2; 3; 4; 5]
Krok pooling matice	[2; 3; 4]
Přidat L2 regularizaci	[Ano; Ne]
Přidat regularizaci do plně propojených vrstev	[Ano; Ne]
Dropout	[0,3; 0,4; 0,5]
Optimalizátor učení	[Adam; RMSprop]

konvoluční vrstvě zdvojnásoben, tedy stejně jako u VGG-16 byl jejich počet rostoucí s hloubkou sítě.

Proudové průběhy průměrované ze čtyř náměrů nejsou sice jednoduše interpretovatelná člověkem, ale úroveň detailnosti pro hluboké učení je dostačující. Také jsou čtyři náměry realistický počet, který by měl být útočník schopný naměřit. Nejlepší nalezený model pro toto průměrování dosáhl přesnosti 99,52 % na validačních datech. Jeho F1 skóre je 0,97. Skládá se z šesti konvolučních vrstev bez L2 regularizace s aktivační funkcí ReLU. První vrstva obsahovala 16 filtrů. Poslední vrstva tedy obsahovala 512 filtrů. Ještě před klasifikací data projdou plně propojenou vrstvou, která obsahuje 1024 neuronů. Tento model je vizuálně znázorněn na obr. 4.2a. Matice záměn tohoto modelu pro testovací data je k vidění na obr. 4.3a.

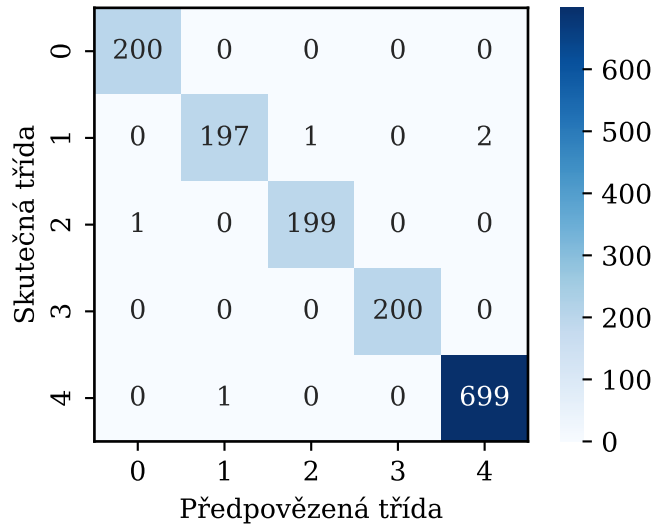
Pro průměrování 1024 obsahoval nalezený model čtyři konvoluční vrstvy s aktivační funkcí SeLU, kde první vrstva obsahovala 16 filtrů. Po poslední konvoluční vrstvě je finální klasifikační vrstva. Tento model dosahoval přesnosti 99,77 % na testovacích datech. Jeho matice záměn na testovacích datech je k vidění na obr. 4.1. Obdobné modely byly nalezené pomocí stejného vyhledávacího prostoru pro hodnoty průměrování 0 a 2. Ty dosáhly přesnosti 97,87 %, resp. 98,19 % na testovacích datech. Nalezené hodnoty pro jejich architektury jsou k vidění v tab. 4.2.

Dataset s jitterem

Datasety použité v této práci byly naměřeny pomocí laboratorního vybavení s vysokou přesností, proto lze tedy předpokládat dokonalou synchronizaci náměrů. Při útoku v reálném světě ale útočník k takto přesné technice nebude mít přístup. Ná-

Tab. 4.2: Nalezené hodnoty pro architektury konvolučních sítí pro různé průměrování dat.

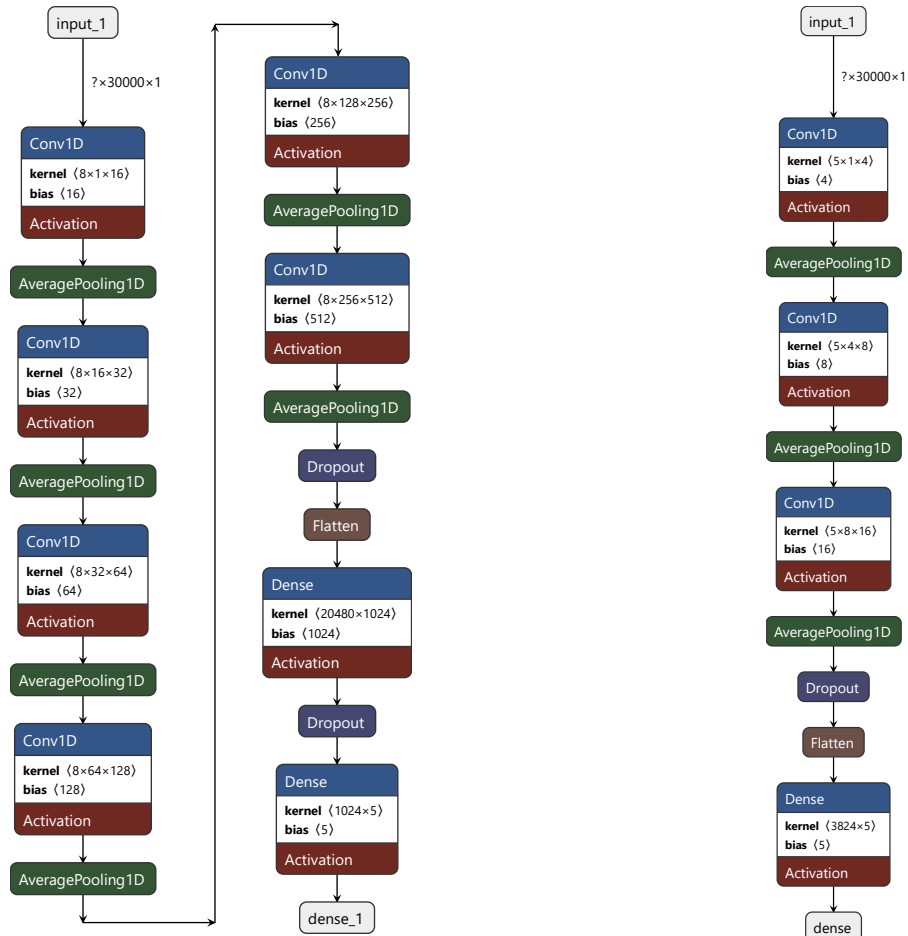
	Stupeň průměrování náměřů									
	Bez jitteru					S jitterem ± 4				
	0	2	4	1024	0	2	4	4	1024	
Hyperparametr										
Konvoluční vrstvy	7	7	6	4	6	6	3		6	
Konvoluční filtry	16	8	16	16	16	8	4		16	
Velikost konvolučního filtru	6	6	8	4	6	8	5		4	
Aktivační funkce	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	
Plně propojené vrstvy	1	1	2	1	2	2	1		1	
Neurony plně propojené vrstvy	-	-	1024	-	256	256	-		-	
Velikost pooling matice	4	2	5	2	4	2	8		4	
Krok pooling matice	3	2	3	2	4	3	5		3	
Přidat L2 regularizaci	Ne	Ano	Ne	Ne	Ne	Ano	Ano		Ne	
Hodnota L2 regularizace	-	0,05	-	-	0,10	0,05	0,15		-	
Přidat regularizaci do plně propojených vrstev	Ano	Ne	Ano	Ano	Ano	Ano	Ne		Ne	
Dropout	0,4	0,3	0,5	0,5	0,3	0,3	0,5		0,4	
Optimalizátor učení	Adam	Adam	Adam	Adam	Adam	RMSprop	Adam		RMSprop	
Přesnost na testovacích datech [%]	97,87	98,19	99,52	99,77	87,93	59,00	77,67		98,73	
F1 skóre na testovacích datech	0,90	0,94	0,97	0,99	0,81	0,39	0,74		0,98	



Obr. 4.1: Matice záměn CNN trénované datasetu s průměrováním 1024.

měry tedy nemusí být synchronizované, čímž se ovlivní přesnost rozdílů. Proto je důležité vyzkoušet zkoumané přístupy i na datech s posunem. V tomto případě byla desynchronizace simulována pomocí uměle vloženého jitteru. Jeho hodnota se pohybuje v rozmezí ± 4 body.

Vyhledávací prostor pro Keras Tuner byl od předchozích pokusů upravený pro zefektivnění trénování. Tento upravený prostor je zobrazený v tab. 4.3. Nalezený model pro průměrování 4 je znázorněný na obr. 4.2b. Dosáhl přesnosti 77,67 % na testovacích datech. Jeho F1 skóre je 0,74. Jeho matice záměn na testovacích datech je zobrazena na obr. 4.3b. Oproti modelu prezentovanému dříve je jeho architektura jednodušší. Skládá se pouze ze tří konvolučních vrstev, které se skládají z 4, 8 a 16 filtrů. Je nutno podotknout, že v tomto vyhledávání parametrů bylo změněno chování parametru na přidání L2 regularizace. V předchozím hledání byla regularizace přidávána pouze do poslední konvoluční vrstvy a to s fixní hodnotou 0,01. Zde byla přidána do každé vrstvy s proměnnou hodnotou podle volby tuneru. L2 regularizace přidává do ztrátové funkce penalizace pro jednotlivé váhy modelu. To může pomoci snížit pravděpodobnost přetrénování. V tomto konkrétním modelu byla použita s hodnotou 0,15 pro každou konvoluční vrstvu. Hledání modelu pro dataset bez jitteru bylo provedeno i pomocí tohoto vyhledávacího prostoru, ale nalezený model dosahoval horších výsledků (validační přesnost 98,93 %), proto byla uvedena předchozí metoda. Porovnání všech modelů konvolučních neuronových sítí se nachází v tab. 4.1.



(a) Pro dataset bez jitteru.

(b) Pro dataset s jitterem ± 4 .

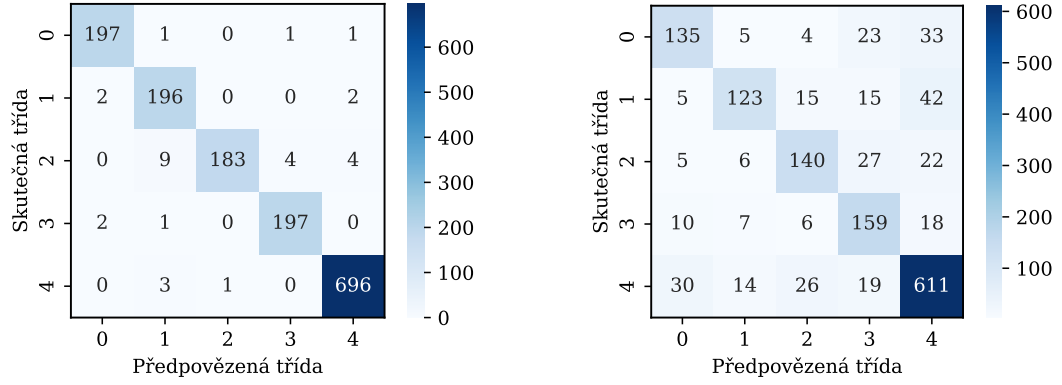
Obr. 4.2: Vizualizace modelů CNN.

Náměry převedené na spektrogram

Konvoluční neuronové sítě se často používají k rozpoznávání obrazu a dosahují v něm dobrých výsledků, díky své schopnosti extrahovat důležité rysy z obrazových dat. Proto se tedy dá předpokládat, že převod na spektrogram by mohl přinést dobré výsledky při klasifikaci.

Spektrogramy se používají především k zobrazení zvukových dat. Jsou to dvou-rozměrné obrazy reprezentující sekvence spektra kde jedna osa zastupuje čas a druhá frekvenci. Jas, nebo různé barvy, reprezentují sílu frekvenčního komponentu v daném čase [25]. Díky tomuto by mělo být možné použít konvoluční neuronové sítě určené ke klasifikaci obrazu na původně jednorozměrná data.

Pro vytvoření spektrogramu ze zvuku se často používají takzvané mel spektrogramy. Stupnice mel, představená v [26], bere v potaz, že lidské ucho nevnímá výšku tónu lineárně podle jejich frekvence. Na nízkých frekvencích jsme schopni rozpoznat



(a) Učené na datasetu bez jitteru.

(b) Učené na datasetu s jitterem ± 4 .

Obr. 4.3: Matice záměn pro CNN na datech s průměrováním 4.

jemnější rozdíly. Například vnímaný rozdíl mezi 500 a 600 Hz je podobný, jako rozdíl který vnímáme mezi 5 000 a 6 000 Hz. Díky této stupnici se tedy dají zvuková data zobrazit věrněji našemu vjemu. Přestože se zde nepracuje se zvukovými daty, byl vyzkoušen i převod na spektrogramy v mel stupnici, protože se jedná o určitou redukci dat, která by mohla ovlivnit výsledky.

Konvoluční vrstvy byly upravené, aby byly schopné pracovat s dvourozměrnými daty. Stejně tak byla změněna standardizace dat. Spektrogramy byly vytvořeny z datasetu s průměrováním 4 bez jitteru. Příklady vytvořených spektrogramů jsou na obr. 4.5, 4.6. Pro Keras Tuner byl zvolen vyhledávací prostor který je zobrazený v tab. 4.1. Tento přístup nepřinesl vhodné výsledky, kdy modely nepřesáhly 65% přesnost. Toto platilo pro běžné, i mel spektrogramy.

Spektrogram lze také reprezentovat formou matice hodnot. Trénování konvolučních sítí bylo vyzkoušeno i na těchto datech. Nebylo však zaznamenáno žádné zlepšení výsledku, a to ani při použití mel spektrogramů.

Spektrogramy se vytváří pomocí Fourierovy transformace, jelikož představují rozložení signálu na jeho jednotlivé složky. Běžně se používají krátkodobé Fourierovy transformace (short-time Fourier transform, STFT). Pro ní je třeba specifikovat vícero parametrů, jako je například velikost okna pro výpočet transformace, počet vzorků mezi po sobě jdoucími okny, frekvence vzorkování vstupních dat, nebo u mel spektrogramů počet frekvenčních pásem pro jeho vytvoření. V této práci nebylo dostatečně prozkoumáno, jaký dopad na výsledek mají tyto jednotlivé parametry. Proto by bylo vhodné vytvořit více datasetů s jinými hodnotami parametrů pro vytváření spektrogramů a porovnat jejich výsledky. Ale jak prezentuje práce [27], existují efektivnější způsoby převodu náměrů proudové spotřeby na obrazová data v kontextu analýzy postranních kanálů.

Tab. 4.3: Vyhledávací prostor pro nalezení modelu CNN na datasetu chráněném jitterem.

Hyperparametr	Hodnoty k prozkoumání
Konvoluční vrstvy	[3; 4; 5; 6; 7]
Konvoluční filtry	[4; 8; 12; 16]
Velikost konvolučního filtru	[3; 4; 5; 6]
Aktivační funkce	[ReLU; SeLU]
Plně propojené vrstvy	[1; 2; 3]
Neurony plně propojené vrstvy	[512; 1024; 2048]
Velikost pooling matice	[6; 8; 10]
Krok pooling matice	[5; 6]
Přidat L2 regularizaci	[Ano; Ne]
Hodnota L2 regularizace	[0,05; 0,1; 0,15]
Přidat regularizaci do plně propojených vrstev	[Ano; Ne]
Dropout	[0,3; 0,4; 0,5]
Optimalizátor učení	[Adam; RMSprop; SGD]

Náměry převedené na GASF a GADF obrazová data

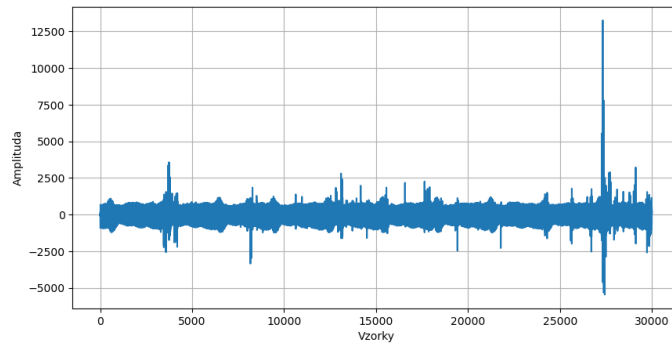
V práci [28] autoři ukázali metodu GAF (Gramian Angular Field), která slouží k převodu časové řady na obrazová data. Tato metoda využívá Gramovy matice v polárních souřadnicích. Přemění časovou řadu o délce N bodů na obraz s rozměrem $N \times N$, což tuto techniku dělá velice paměťově náročnou pro delší data. GASF a GADF jsou její specifické implementace.

V **GASF (Gramian Angular Summation Field)** se prvky Gramovy matice vypočítávají jako kosinus součtu úhlů souřadnic. Výsledný obraz zachycuje globální trendy a podobnosti v časové řadě.

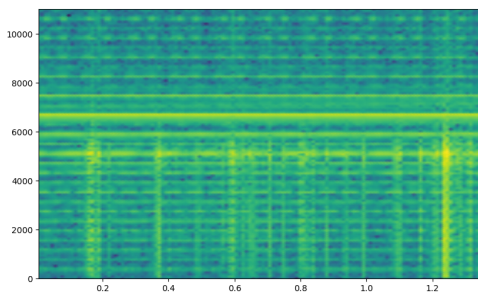
Prvky **GADF (Gramian Angular Difference Field)** se počítají jako sinus rozdílu úhlů. Jeho výsledek obsahuje rozdíly a změny mezi časovými okamžiky. Proto je vhodný pro detekci anomálií, nebo náhlých změn v datech.

Autoři v práci [27] spojili GASF a GADF obrázky podél osy y a použili tuto dvojici jako vstup do konvoluční sítě. Jejich implementace dosáhla výrazně lepších výsledků než sítě trénované pouze na náměrech spotřeby. Jimi použité náměry byly ale pouze 70 bodů dlouhé, což výrazně zmenšilo paměťové požadavky. Náměry používané v této práci jsou 30 000 bodů dlouhé, což je činí nepoužitelné pro převod pomocí GAF technik bez předchozího snížení dimenzionality.

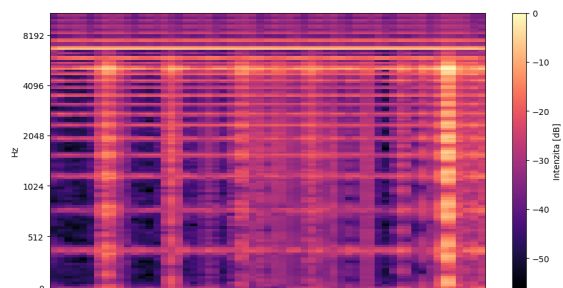
Technik na snížení dimenzionality je mnoho. V práci [27] zmiňují například difference of means (DOM), sum of squared differences (SOSD), nebo principal component analysis (PCA). V této práci byla zvolena technika PCA pro zmenšení paměťo-



Obr. 4.4: Rozdíl dvou zpráv, zprůměrováno z 1024 náměřů.



Obr. 4.5: Spektrogram vytvořený z rozdílu zpráv.



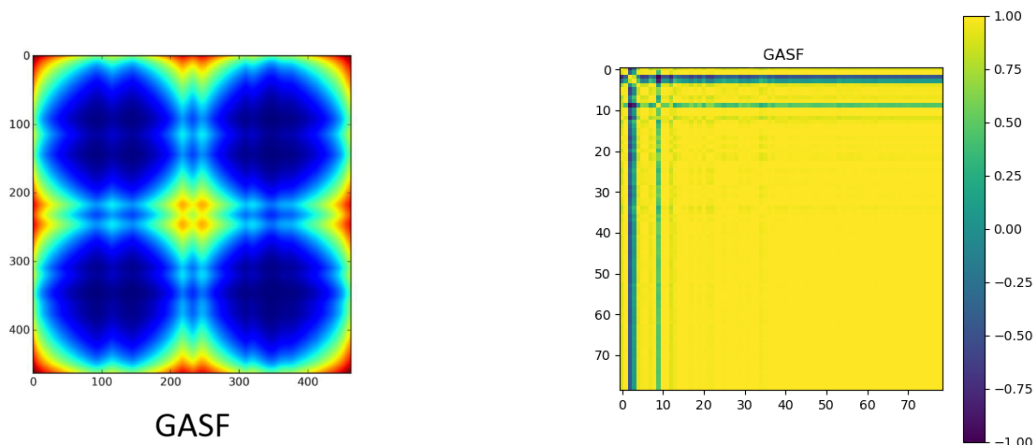
Obr. 4.6: Mel spektrogram vytvořený z rozdílu zpráv.

vých nároků řešení. Tato volba byla nevhodná, protože při zmenšení dimenzionality dat pomocí PCA zaniká časová informace, která je pro GAF klíčová. Takto nevhodně vygenerovaný GAF obrázek je k vidění na obr. 4.7. Výsledné modely sice dosahovaly přesnosti 46,67 % na validačních datech, ale to pouze klasifikováním všech dat do třídy 4. Tato třída představuje přibližně 46,67 % prvků celého datasetu. Matice záměn takového modelu je k vidění na obr. 4.8. To může vypovídat o nevhodné implementaci vah pro jednotlivé třídy, nebo snahu optimalizátoru dosáhnout co největší validační přesnosti, která byla použita jako cílová metrika.

Pro zlepšení výsledků v budoucích pokusech je třeba použít jinou techniku zmenšení dimenzionality dat, nebo se zaměřit na přesnější zvolení bodů zájmu (points of interest, POI) na náměrech. Tím by bylo možné získat kratší data, která by byla pro převod pomocí techniky GAF vhodnější.

Použití LSTM vrstev

Jako další bylo vyzkoušeno zakomponování LSTM vrstev do modelu. Opět byl použit vyhledávací prostor znázorněný v tab. 4.3. Byly k němu ale přidány dva hyperparametry k prohledání. Přidání LSTM vrstvy a počet LSTM buněk v dané vrstvě, viz



(a) Korektně vytvořený GASF obrazec z práce [28].

(b) GASF vytvořený z dat po redukci pomocí PCA.

Obr. 4.7: Porovnání GASF obrázků vygenerovaných z časové řady.

tab. 4.4.

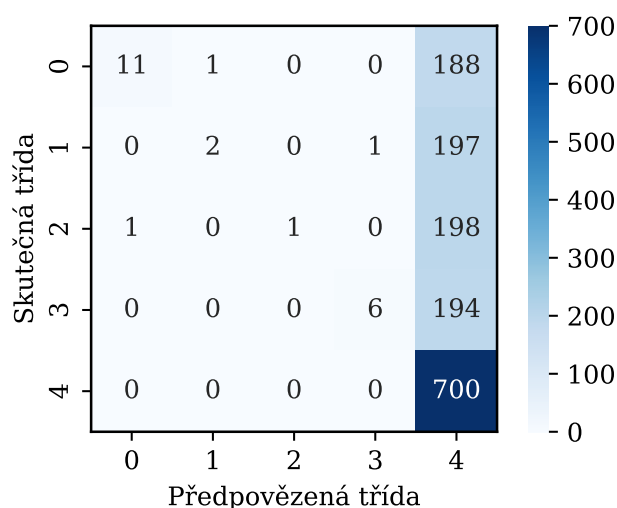
Tab. 4.4: Přidané parametry pro hledání modelu s LSTM vrstvou.

Hyperparametr	Hodnoty k prozkoumání
Přidat LSTM vrstvu?	[Ano; Ne]
Počet LSTM buněk	[32; 64; 128]

Po prohledání 100 možných kombinací modelů ani jeden z deseti nejlepších kandidátů neobsahoval LSTM vrstvu. Kód byl navržený tak, aby se přidala jedna LSTM vrstva za poslední běžný konvoluční blok. Je možné, že použité zakomponování této technologie do modelu nebylo vhodné, což mohlo omezit její účinnost. LSTM technologie je určena k zachycení vazeb napříč dlouhými úseky času. Úniky z proudového postranního kanálu mohou být soustředěné na malý počet bodů, tudíž by tato výhoda udržovat data v paměti po delší dobu nemusela být využita [12]. Pro možné zlepšení výsledků by bylo vhodné prozkoumat více různých implementací.

4.2.2 Transformerová síť

Mechanismus multihead self-attention, který je pro tuto architekturu unikátní, zpracovává data paralelně napříč všemi hlavami, namísto sekvenčně, jak tomu je například u konvolučních sítí. Kvůli tomu jsou tyto techniky paměťově náročné a vyžadují předzpracování dat, aby mohly efektivně fungovat na běžně dostupném hardwaru, jako jsou například herní GPU. Použité náměry, které měly velikost 30 000 bodů



Obr. 4.8: Matice záměn CNN trénované na kombinaci GASF a GADF s přesností 46,67 %.

nebyly v kombinaci s náročností této techniky použitelné. Jako předzpracování byly zvoleny již vytvořené variace datasetu pomocí převodu na spektrogramy. V jejich 1D i 2D podobě. Oba datasety jsou totožné těm, které byly použity v části 4.2.1.

Model byl převzat z článku [29]. Při trénování byly pozorovány přesnosti, které se také pohybovaly okolo 46,67 % na validačních datech. Pro to může být více vysvětlení. Model například nemusí být ideálně strukturovaný pro použitá data, nebo provedený převod náměrů na spektrogramy ztratil přespříliš informace z původních dat, což by odpovídalo výsledkům z části 4.2.1.

EstraNet

Hajra a kolektiv v práci [18] představili model EstraNet, který navazuje na jejich předchozí práci [19], ve které použili transformerovou architekturu k útoku na proudový postranní kanál. Podle autorů má tento model mít lineární paměťovou složitost, což je velké zlepšení, oproti kvadratické paměťové složitosti běžných transformerových sítí.

Menší paměťová složitost byla ověřena použitím základního datasetu s dlouhými náměry. Oproti běžnému modelu transformerové sítě bylo využití paměti značně nižší. I přes tento úspěch se nepodařilo natrénovat model s vysokou přesností. Opět bylo dosaženo nejvyšší přesnosti 46,67 % a v průměru modely dosahovaly pouze na 14% přesnost na testovacích datech. Výsledky byly podobné pro základní dataset, i pro dataset chráněný jitterem s hodnotou ± 4 . Pro použití architektury EstraNet by použité náměry pravděpodobně vyžadovaly odlišné předzpracování. Je možné, že

použitá data se až přespříliš lišila těm, na která byla tato architektura navržena. Dalším bodem pro zlepšení by mohla být optimalizace zahřívací doby a rychlosti učení.

4.2.3 Předtrénované modely

V různých aplikacích je běžné používat již natrénované modely, které se pouze „dotrénují“ na vlastních datech. Z již natrénovaného modelu se odstraní poslední klasifikační vrstva a ponechají se pouze části modelu, které už jsou naučené k extrakci důležitých prvků. Na konec modelu jsou přidány vlastní vrstvy, typicky jedna plně propojená vrstva a poslední klasifikační vrstva. Trénování jde provádět na nově přidaných vrstvách, nebo i na původních vrstvách originálního modelu. To je vhodné, pokud se použitá data při dotrénování výrazně liší od původních trénovacích dat modelu a je nutné model naučit rozpoznávat jejich specifické vlastnosti. Je možné „odmrazit“ několik vrstev z konce modelu, díky čemuž bude možné, aby se v těchto vrstvách měnily již naučené váhy. V kombinaci s odmrazováním vrstev je běžné použít malou rychlost učení, aby se předešlo velkým změnám.

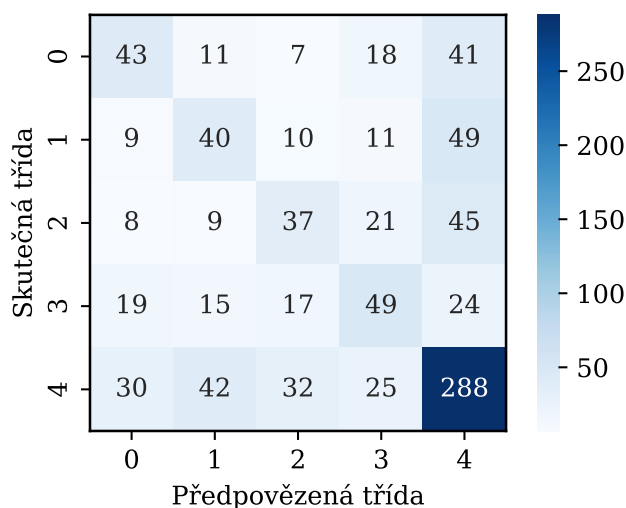
Pro přetrénování byly zvoleny modely VGG-16 a ResNet50. Oba tyto modely slouží ke klasifikaci obrazu. Oba byly také naučeny na datasetu ImageNet [30], který obsahuje přes 14 milionů označených obrázků, které jsou rozděleny do 1000 tříd (v jeho menším subsetu ILSVRC, ImageNet Large Scale Vision Recognition Challenge). V datech jsou zastoupené různé kategorie, jako například zvířata, objekty a scény (pláž, jeskyně, svítání...).

Pro trénování byly zvoleny již dříve vytvořené spektrogramy. Přestože se nemusí jednat o ideální vyjádření původních dat, bylo tak učiněno pro ověření konceptu. Jelikož se použitá data výrazně liší od dat na kterých byly tyto sítě původně natrénované, bylo zvoleno přetrénování posledních pěti vrstev u obou modelů.

Model VGG-16 dosáhl lepších výsledků v přesnosti na trénovacích datech než ResNet50. Přesto oba modely neuspěly na validačních datech, jak je možné vidět v tab. 4.5. To může značit špatnou generalizaci, způsobenou například overfittingem. Model se až moc přesně naučil vlastnosti trénovacích dat, takže je schopný je efektivně klasifikovat, ale nezvládne tuto schopnost přenést i na nová, dříve neviděná data. Jedním z důvodů může být velice rozdílná povaha použitého datasetu od dat obsažených v ImageNetu, na kterém byly sítě původně trénované. Overfittingu se dá předejít pomocí střídavého vynechání určitých neuronů při trénování (dropout), aby se na nich model nestal příliš závislý při vytváření výsledku. Dalším řešením může být použití L2 regularizace vrstev, která byla vysvětlena v části 4.2.1. Matice záměn modelu VGG-16 na testovacích datech je zobrazena na obr. 4.9. Dosáhl F1 skóre pouze 0,42.

Tab. 4.5: Přesnosti přetrénovaných modelů.

	Trénovací přesnost	Validační přesnost
VGG-16	97,97 %	50,78 %
ResNet50	67,97 %	57,44 %



Obr. 4.9: Matice záměn sítě VGG-16 dotrénované na vlastním datasetu.

4.2.4 Autoenkodéry

Enkodéry jsou schopné v daném datasetu efektivně hledat části dat, které se odlišují od ostatních. Proto by jimi vytvořené reprezentace v latentním prostoru měly být ovlivněné především vlastnostmi dat, které je odlišují od ostatních dat. To může vést ke chtěné extrakci bodů zájmu. V ideálně vytvořené latentní reprezentaci dat by se měly tvořit skupiny vzorků, které patří do stejné třídy. Tímto by také měly pomoci při finální klasifikaci.

Dalším způsobem pro možné nalezení bodů zájmu je natrénování dekodéru, který z latentní reprezentace vytvoří data kratší, než byla data na výstupu. Tento dekodér by měl být schopný zachovat nejdůležitější informace z původního vstupu a odstranit redundantní nebo šumové složky. Pokud je autoenkodér natrénovaný správně, měl by při rekonstrukci upřednostnit nejdůležitější rysy, včetně důležitých bodů zájmu.

Samotné trénování může probíhat více způsoby. Lze vytvořit jeden model, který obsahuje jak enkodér pro extrakci důležitých částí, tak klasifikační síť. Obě tyto části se mohou trénovat společně a optimalizovat je podle ztrátové funkce klasifikace, což by mělo vést enkodér k vytváření reprezentace dat vhodné pro klasifikaci. Tímto

Tab. 4.6: Vyhledávací prostor pro nalezení modelu autoenkodéru.

Hyperparametr	Hodnoty k prozkoumání	Nejlepší hodnoty
Vrstvy enkodéru	[2; 3; 4; 5]	5
Vrstvy dekodéru	[1; 2; 3]	2
Konvoluční filtry enkodéru	[32; 64; 128]	64
Konvoluční filtry dekodéru	[32; 64; 128]	64
Velikost konvolučního filtru enkodéru	[3; 4; 5; 6; 7]	7
Velikost konvolučního filtru dekodéru	[3; 4; 5; 6; 7]	5
Krok první podvzorkovací konvoluční vrstvy	[10; 15; 20]	10
Použít dávkovou normalizaci v enkodéru	[Ano; Ne]	Ne
Použít dropout v enkodéru	[Ano; Ne]	Ne
Použít dávkovou normalizaci v dekodéru	[Ano; Ne]	Ano
Použít dropout v dekodéru	[Ano; Ne]	Ne
Hodnota dropout enkodéru	[0,1; 0,2; 0,3; 0,4]	-
Hodnota dropout dekodéru	[0,1; 0,2; 0,3; 0,4]	-
Optimalizátor učení	[Adam; RMSprop]	Adam

způsobem byly v této práci doposud trénované všechny modely. Jde o běžné použití konvolučních vrstev k extrakci vlastností dat.

Další možný přístup je přidat do modelu dekódovací větve, počítat i její ztrátovou funkci a celkový model hodnotit podle váženého průměru obou těchto hodnot. Korektně rekonstruovaná data by zároveň měla naznačovat správně vyjádřené vlastnosti v latentní reprezentaci, které by mohly vést ke správné klasifikaci.

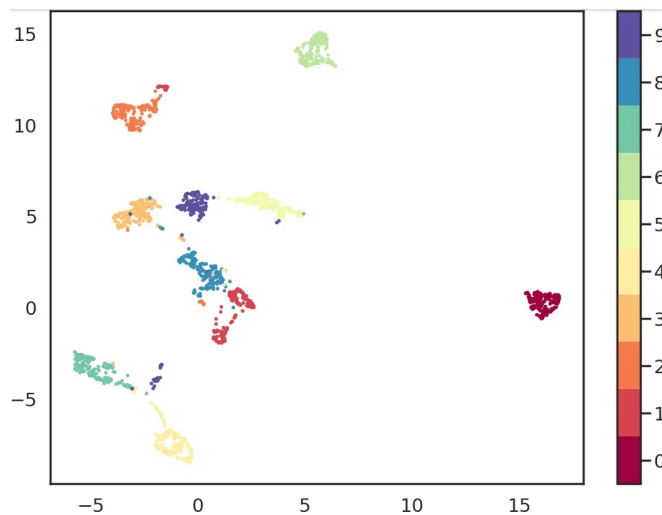
Možností je také natrénovat nejdříve samotný autoenkodér. Trénovat lze pro správnou rekonstrukci, nebo při použití architektury jako jsou β -VAE pro více separovanou latentní reprezentaci. Po dokončení trénování modelu lze zmrazit jeho vrstvy, získat výsledek jeho enkodéru a ten použít pro trénování klasifikační sítě. Toto má také výhodu menší paměťové náročnosti, jelikož se učí vždy pouze jedna část sítě. Pokud se učí autoenkodér společně s klasifikační sítí, je nutné mít dostatek paměti pro zpětný výpočet chyby všemi vrstvami.

Běžný předtrénovaný autoenkodér

Architektura modelu běžného autoenkodéru pro dataset s průměrováním 4 byla opět hledána pomocí nástroje Keras Tuner. Prostor pro hledání hyperparametrů je zobrazen v tab. 4.6. Tato nalezená architektura dosáhla dobrých výsledků při rekonstrukci dat, konkrétněji hodnot 0,000017 rekonstrukční ztrátové funkce MSE (Mean Squared Error).

Enkodérem vytvořenou reprezentaci dat je možné vizualizovat například pomocí

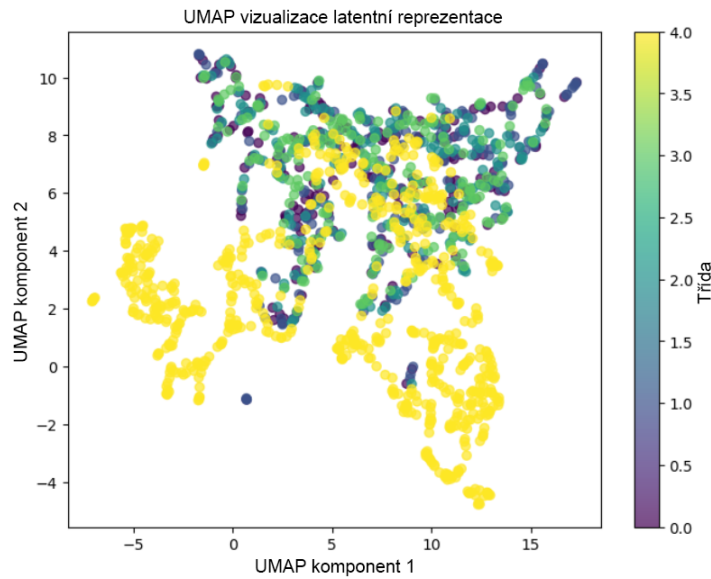
technik t-SNE (t-distributed Stochastic Neighbor Embedding), nebo UMAP (Uniform Manifold Approximation and Projection). Technika t-SNE slouží k redukci dimenzionality dat, podobně jako PCA, ale také se často využívá k vizualizaci dat v 2D nebo 3D prostoru. Snaží se vypočítat podobnosti bodů v původním prostoru, jako je například jejich vzdálenost, poté nadefinovat, jak by tyto vztahy mohly vypadat v nízkodimenzionálním prostoru, a poté minimalizovat rozdíly mezi těmito vyjádřeními. Na rozdíl od PCA dokáže t-SNE správně reprezentovat datasety, které obsahují nelineární závislosti [32]. Je ale také mnohem náročnější na výpočet. UMAP funguje na podobných principech, ale je jednodušší na výpočet [33]. Proto byl zvolen pro vizualizaci latentních reprezentací enkodérů. Nezávisle na použité technice vizualizace by měl dobrý enkodér být schopný vytvořit jednoznačně oddělené skupiny dat podle jejich vlastností. Očekávanou vizualizaci správně separovaných dat je možné vidět na obr. 4.10.



Obr. 4.10: UMAP vizualizace dobře separované latentní reprezentace ze stránky [34].

Jak je vidět na obr. 4.11, nalezený model autoenkodéru není schopný správně separovat jednotlivé třídy dat v jeho latentní reprezentaci. Jedině pátá třída datasetu, která obsahuje rozdíly zpráv, které se liší ve více než jednom bytu, je částečně oddělená od zbytku dat. Toto může být způsobené velkou podobností dat v našem datasetu, jelikož místa důležitého úniku informací na náměrech jsou malá a málo odlišná mezi jednotlivými třídami. Pouze prvky páté třídy jsou výrazněji odlišné od zbytku, což by také společně s vyšším počtem vzorků ve třídě mohlo vysvětlovat její jednodušší separovatelnost od zbytku dat.

Pro následnou klasifikaci byla použita pouze enkodérová část nalezeného modelu. Enkodér tedy zastává funkci extraktoru příznaků jako v běžném modelu CNN. Byl pouze naučený předem, separátně od učení klasifikační části sítě. Pro hledání



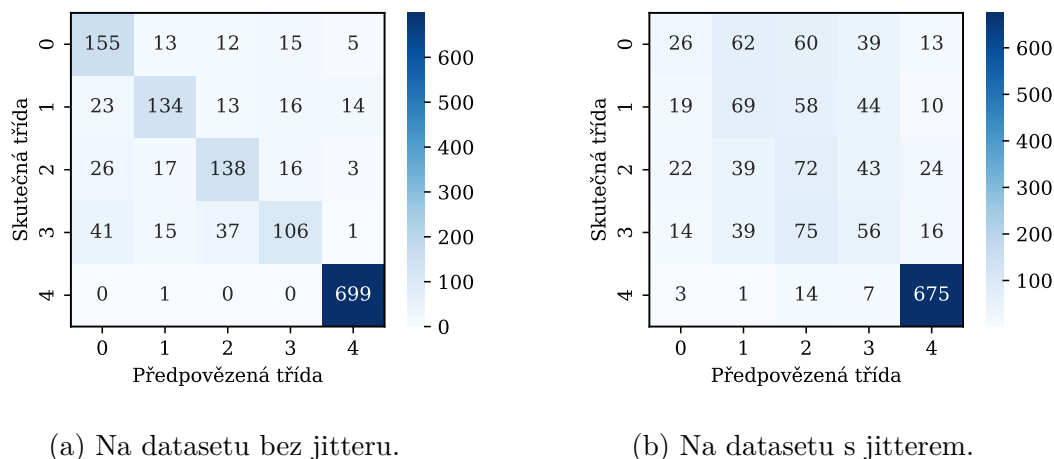
Obr. 4.11: UMAP vizualizace autoenkodéru nalezeného nástrojem Keras Tuner.

Tab. 4.7: Vyhledávací prostor pro nalezení klasifikační části modelu s enkodérem.

Hyperparametr	Hodnoty k prozkoumání	Nejlepší hodnoty
Počet vrstev enkodéru k odmražení	[0; 1; 2]	2
Konvoluční vrstvy	[2; 3; 4]	2
Konvoluční filtry	[4; 8; 16; 32; 64]	32
Velikost konvolučního filtru	[3; 5; 6]	3
Počet plně propojených vrstev	[1; 2]	2
Neurony plně propojené vrstvy	[256; 512; 1024]	512
Hodnota dropout	[0,1; 0,2; 0,3; 0,4]	0,3
Velikost pooling matice	[2; 3; 4]	2

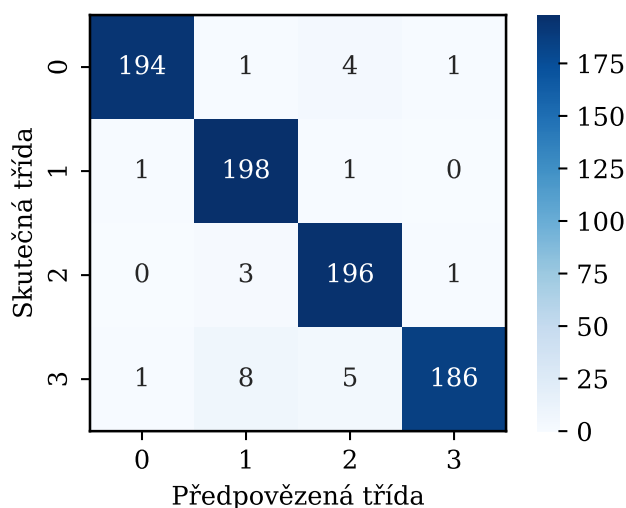
parametrů klasifikační části sítě byl opět použit Keras Tuner. Prohledávané hyperparametry jsou uvedené v tab. 4.7.

Pro dataset s průměrováním 4 bez jitteru dosáhl nalezený model přesnosti 82,40 % na testovacích datech. Po přetrénování enkodéru na datasetu s jitterem ± 4 však ale klasifikační přesnost na testovacích datech klesla na 59,87 %. Enkodér nebyl dostatečně odolný vůči změnám v datech způsobených jitterem. Pokud se model naučil extrahovat hlavně lokální rysy, které se vlivem jitteru objevují na jiných pozicích, enkodér je nemusí efektivně rozpoznat. Jedním z možných řešení by bylo použití robustnější architektury, která by byla schopná lépe zachytit klíčové vlastnosti dat i v přítomnosti jitteru. Toto by ale také přineslo větší paměťovou náročnost. Matice záměn obou modelů jsou k vidění na obr. 4.12. Pro dataset bez jitteru dosáhl model



Obr. 4.12: Matice záměn pro modely s extraktorem prvků z běžného autoenkodéru.

F1 skóre 0,74 a na datasetu s jitterem 0,41.



Obr. 4.13: Matice záměn klasifikačního modelu založeného na běžném autoenkodéru na testovacích datech při použití pouze čtyř tříd.

Vizualizace latentní reprezentace i matice záměn modelů naznačují, že učení extraktoru prvků může být zkreslené přítomností páté třídy dat, která se jako jediná výrazně odlišuje od zbytku. Při trénování autoenkodéru pouze na prvních čtyřech třídách je následný klasifikační model schopen dosáhnout přesnosti 98,35 % na testovacích datech a F1 skóre 0,97. Jeho matice záměn je zobrazena na obr. 4.13. Bylo by vhodné prozkoumat natrénování binárního klasifikátoru, který by dataset rozdělil na rozdíly zpráv obsahující kolizi a ty bez kolize. Tímto způsobem by se následný

Tab. 4.8: Nalezený model pro klasifikaci s použitím enkodéru z β -VAE.

Hyperparametr	Nejlepší hodnoty
Počet vrstev enkodéru k odmražení	0
Konvoluční vrstvy	3
Konvoluční filtry	4
Velikost konvolučního filtru	6
Počet plně propojených vrstev	1
Neurony plně propojené vrstvy	512
Hodnota dropout	0,4
Velikost pooling matice	4

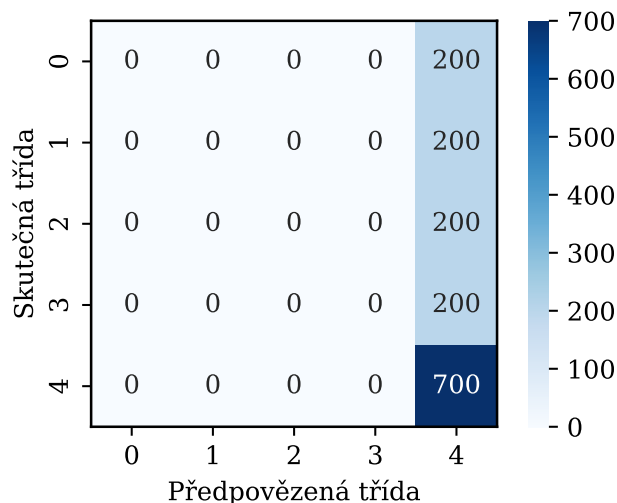
extraktor prvků klasifikační sítě mohl lépe soustředit na relevantní vlastnosti kolizí.

β -variační autoenkodér

Jelikož byl pro klasifikaci použit pouze výstup enkodéru, přesnost rekonstrukce dat není klíčová. Naopak, dobře separovaný latentní prostor by mohl napomoci ke správné klasifikaci jednotlivých prvků. K tomuto je nutné experimentovat s hodnotou β a zjistit, jaká je vhodná pro chtěný výsledek. Porovnání UMAP vizualizací latentních reprezentací datasetu bez jitteru s průměrováním 4 je k vidění na obr. 4.15. Tyto vizualizace napovídají, že model není schopný správně separovat jednotlivé třídy a to i přes vysokou hodnotu β . Ve všech případech jsou třídy stále promíchané a není patrná jasná separace. Jedině pro třídu čtyři, ve které mohou být prvky velmi odlišné od ostatních tříd, se vytváří skupiny bodu separované od ostatních tříd, přesto jsou tyto skupiny ale málo oddělené od prvků ostatních tříd.

Pro nalezení klasifikační části modelu byl použit stejný vyhledávací prostor jako byl v předchozí části, který je zobrazený v tab. 4.7. Parametry nalezeného modelu jsou k vidění v tab. 4.8. Jako extraktor prvků byl zvolen enkodér z modelu s hodnotou $\beta = 30$, kvůli dobrému kompromisu separace a kvalitní rekonstrukce dat. Vyšší hodnoty β podle porovnání reprezentací na obr. 4.15 nepřinesly značné zlepšení.

Nalezený model dosahoval přesnosti pouze 46,67 % na datasetu bez jitteru. Jeho matice záměn je zobrazena na obr. 4.14. Přesnosti pro dataset s jitterem naměřené nebyly. Špatná klasifikace může být způsobena nevhodnou architekturou klasifikačního modelu, nebo samotným enkodérem, který se naučil extrahovat špatné vlastnosti dat. Také tomu mohl přispět samotný princip variačních enkodérů. Jelikož reprezentace do latentního prostoru mapují ne na diskrétní body, ale na vektory, je možné, že se ve spojitosti se špatnou separací tříd vektory jednotlivých prvků překrývají a dále tedy ztěžují následnou klasifikaci.



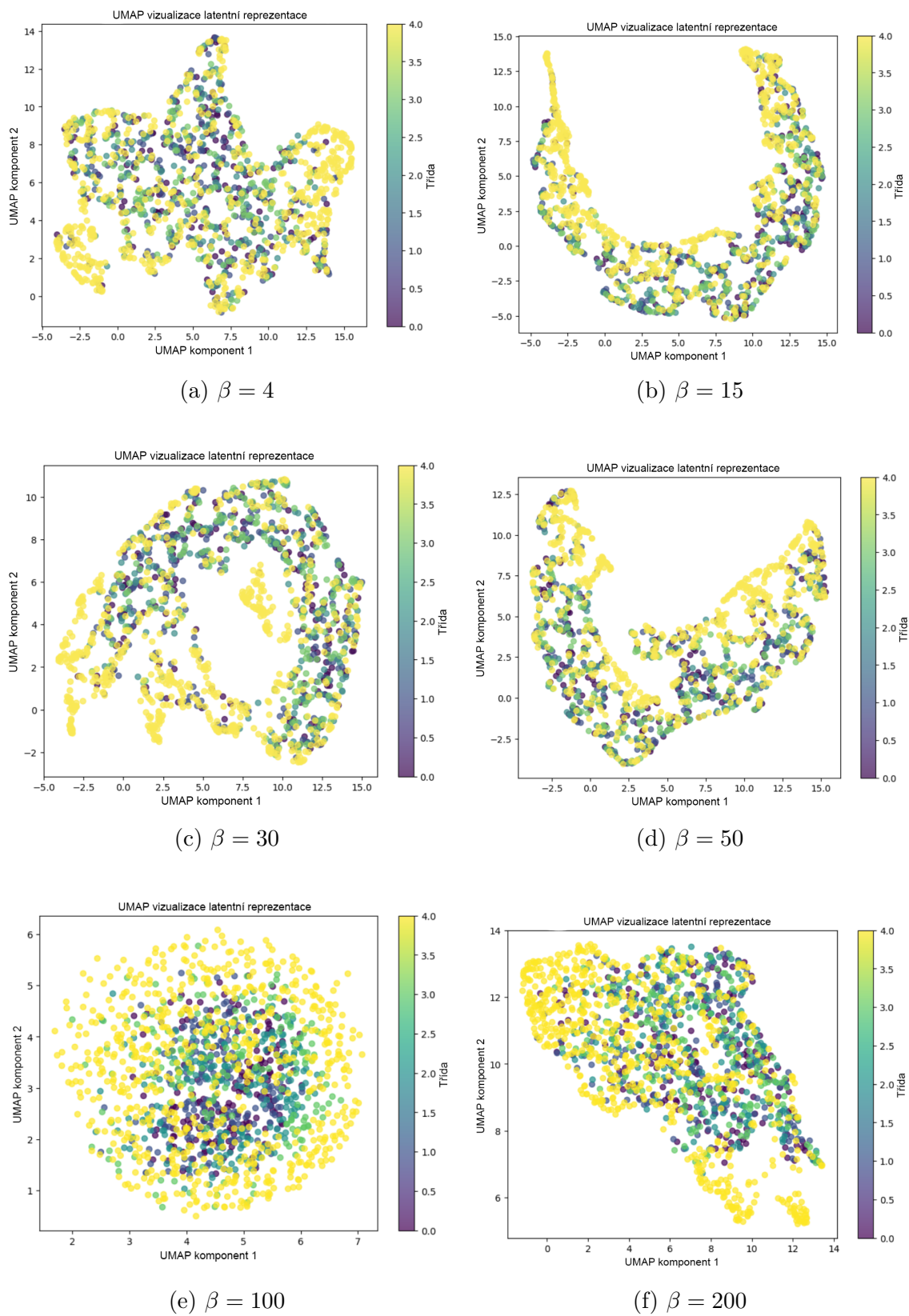
Obr. 4.14: Matice záměn klasifikačního modelu s použitím β -VAE enkodéru.

4.2.5 Předtrénovaný extraktor prvků

V předchozí části byl enkodér extrahován z autoenkodéru, který se původně učil na rekonstrukci vstupních dat. Model se snažil upravovat váhy v jednotlivých vrstvách takovým způsobem, aby se výsledná rekonstrukce co nejvíce podobala vstupním datům. Tato část se zaměřuje pouze na enkodér, který se učí extrahovat užitečné rysy tak, aby si byl vědom které vzorky si jsou podobné a které ne. Tento přístup se nazývá kontrastní učení [35]. Model sám označí data podle libovolných vlastností a snaží se je podle těchto označení rozdělit. Učení probíhá na takzvaných pretextových úlohách, kde se model snaží například doplnit chybějící, nebo deformované části vstupních dat. Při trénování autoenkodéru se enkodér naučí vytvářet reprezentace dat vhodné pro jejich rekonstrukci. Kontrastivním učením ale model získá univerzálnější reprezentace dat, které jsou vhodnější pro více úkonů, jako jsou například rozpoznávání řeči, strojový překlad, nebo klasifikace [35].

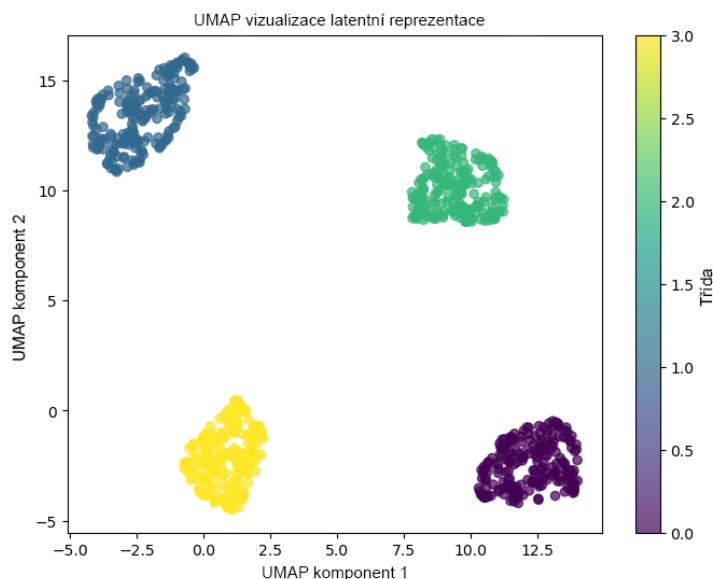
Jednou z největších výhod self-supervised učení je, že nevyžaduje označená data, která jsou často drahá k získání. Dataset použitý v této práci ale obsahuje označená data, díky tomu se naskytuje možnost využít kontrastivní učení s učitelem, které bylo představené v práci [36]. Tento přístup při učení ulehčuje problém identifikování podobných a odlišných dat.

Pro dataset kde byly zprávy průměrované z 1024 náměrů byl model schopný správně separovat jednotlivé třídy, jak je vidět na obr. 4.16. U self-supervised kontrastivního učení hraje velkou roli použitá velikost dávky (batch size) při učení [36]. Implementace SimCLR například používá velikosti várek až 8196. Model v rámci dávky porovnává negativní vzorky k pozitivním. Pokud je várka větší, je schopný se



Obr. 4.15: UMAP reprezentace latentního prostoru pro různé hodnoty β .

lépe naučit malé rozdíly mezi daty, protože je může porovnat s více vzorky. Tento problém by měl být do určité míry vyřešen použitím varianty kontrastivního učení s učitelem, kdy by mělo být rozlišení mezi jednotlivými daty podpořené přítomností označení. Přesto byl pozorován rozdíl v separaci při trénování s velikostí dávky 32 a 64. Tento rozdíl je více patrný při trénování se všemi třídami. Porovnání těchto reprezentací je k vidění na obr. 4.17. Při trénování s menší várkou je přesah třídy 4 do jiných větší než s větší várkou. Tento rozdíl může být způsoben vyšším počtem prvků ve třídě 4 oproti ostatním třídám. Při menších velikostech dávky se mohlo stát, že model při trénování dostal várku z většiny obsahující data ze třídy 4, a poté nebylo možné správně porovnávat data s ostatními, což mohlo zkreslit učení.

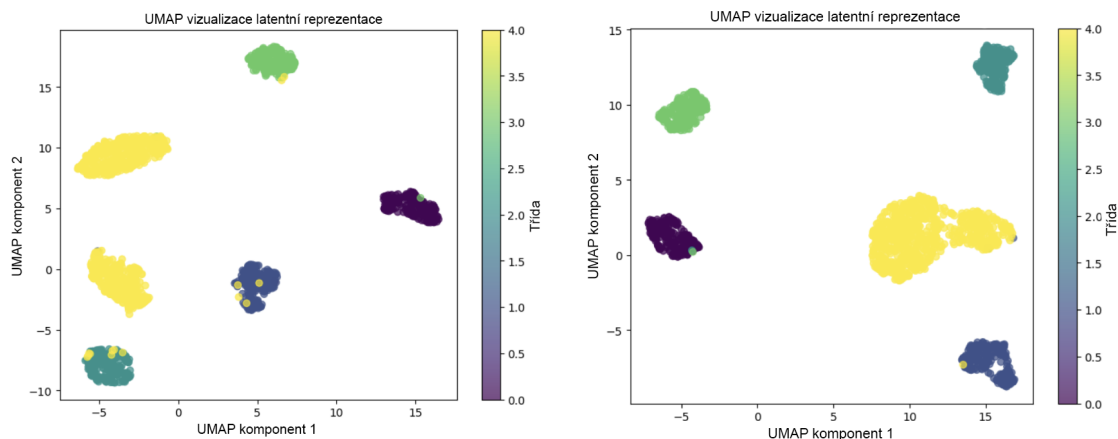


Obr. 4.16: UMAP reprezentace modelu trénovaného kontrastivním učením pouze na čtyřech třídách s průměrováním náměrů 1024.

Pro dataset s průměrováním čtyři nebyl model schopný správně rozdělit třídy, jak je možné vidět na obr. 4.18a. Tomu odpovídají i špatné výsledky při klasifikaci. Model dosahoval přesnosti 47,33 % na validačních datech datasetu bez jitteru. Jeho matice záměn je k vidění na obr. 4.18b. Výsledky pro dataset s jitterem nebyly naměřeny. Při trénování byla nastavena hodnota velikosti dávky na 32. Je možné, že experimentace s tímto parametrem by mohla přinést určité zlepšení.

4.3 Útoky na výchozí dataset

Podle původního článku [10], kolize nastane mezi dvěma z přibližně $2^{11,5}$ zpráv. Datasetsy pro simulaci útoků se skládají z 2 901 jednotlivých náměrů zpráv. Z těch



(a) Model trénovaný s várkou 32.

(b) Model trénovaný s várkou 64.

Obr. 4.17: Porovnání UMAP vizualizací při trénování s různými velikostmi várek.

jsou vytvořené všechny kombinace bez opakování, čímž vznikne 4 206 450 různých rozdílů, což je lehce více než 2^{22} . Mezi těmito rozdílů se nachází dvě kolize. Jedna ve třídě 1 a druhá ve třídě 3. Stejně jako trénovací datasety, i tyto datasety určené pro simulované útoky byly vytvořeny s různými hodnotami průměrování náměrů (0, 2, 4 a 1024). K simulovaným útokům byly použity modely které byly vytvořené v kapitole 4.2.1.

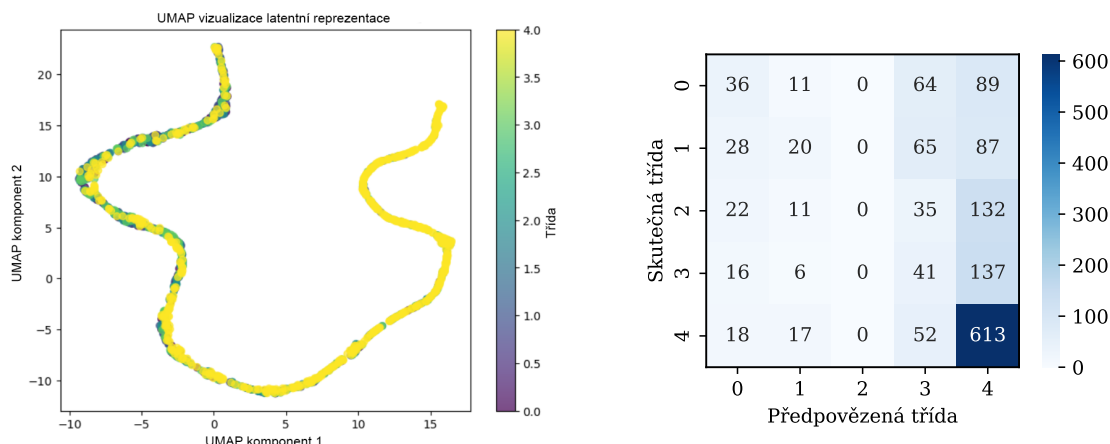
Dataset bez jitteru

Matice záměn útoků všemi použitými modely na jejich odpovídající datasety jsou zobrazeny na obr. 4.19.

Jak je vidět, z hlediska falešně pozitivních výsledků si vedl nejlépe model pro průměrování 1024, který jich určil ze všech modelů nejméně. Přestože korektně určil kolizi s třídou 1, kolizi třídy 3 nebyl schopný identifikovat. Oproti tomu modely pro průměrování 2 a 4 korektně identifikovaly obě kolize které se v datasetu nachází, ale také určily spoustu falešných pozitiv, které je tvoří nepoužitelné pro reálné útoky.

4.4 Dataset naměřený na zařízení založeném na architektuře ARM

Jako další implementace kryptografického algoritmu dostupná na pracovišti VUT je zařízení založené na architektuře ARM. Stejně jako výchozí dataset, tento obsahuje pět tříd. Třídy 0-3 pro jednotlivé aktivní byty po operaci Sbox třetí rundy šifry. Třída 4 poté obsahuje rozdíly zpráv, které obsahují kolizi ve více než jednom bytu.



(a) UMAP vizualizace latentní reprezentace dat. (b) Matice záměn klasifikačního modelu.

Obr. 4.18: Výsledky modelu s předtrénovaným extraktorem prvků pomocí kontrastního učení na datasetu s průměrováním čtyři.

Na rozdíl od výchozího datasetu není ale tato třída obsáhlejší jak zbytek. V tomto datasetu obsahují všechny třídy 1 000 rozdílů zpráv a třídy jsou tedy vyvážené.

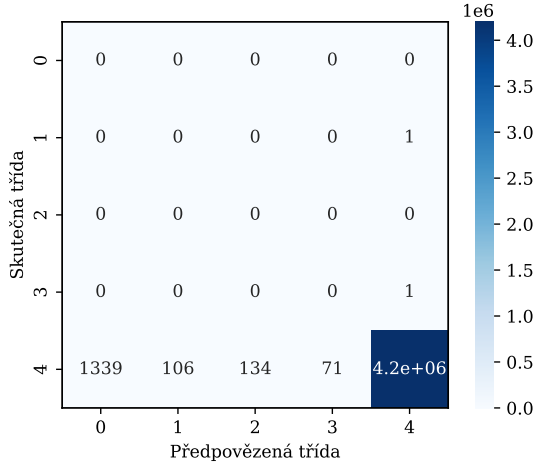
Dalším rozdílem je délka náměrů. Předchozí data byla 30 000 bodů dlouhá, kde tyto se skládají ze 100 000 bodů. Tato vyšší délka by mohla představovat problém pro učené modely, jelikož chtěné úniky informací jsou obsažené pouze v malém časovém okně. Rozdíly zpráv tedy obsahují velké množství redundantních informací, které nijak nepřispívají ke klasifikaci jednotlivých tříd. Pro ověření této teorie byly natrénovány dva modely, použitím upraveného vyhledávacího prostoru z části 4.2.1. Tento upravený prostor je zobrazený v tab. 4.9.

Hledání modelu proběhlo na proudových průbězích průměrovaných z 512 a 4 náměrů. Ani při jedné z těchto variant nebyly nalezené modely schopné překročit přesnosti 25 %. Jejich matice záměn na validačních datech jsou k vidění na obr. 4.20. Je očividné, že tyto architektury nejsou vhodné pro dlouhé náměry.

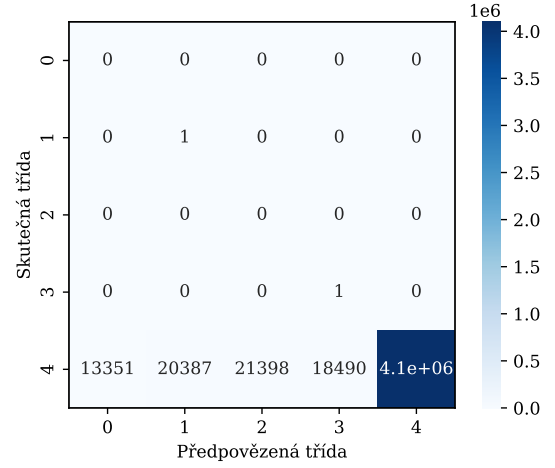
Pro snížení dimenzionality dat bez ztráty důležitých informací je nejvhodnější možností ruční zmenšení časového okna které je sledované, po zjištění pozice zkoumaných operací. Ty je možné nalézt pomocí korelační proudové analýzy (CPA).

Určení bodů zájmu

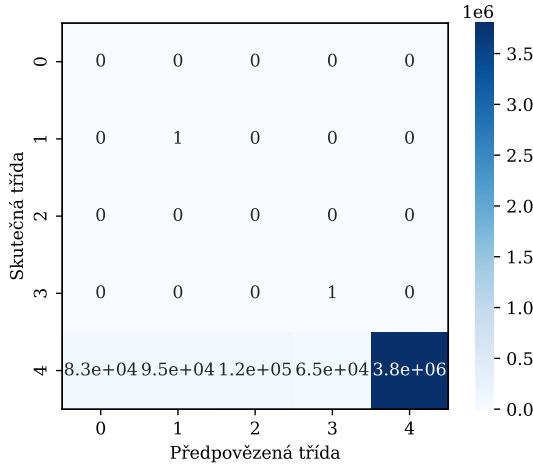
Pro provedení CPA, která byla vysvětlena v kapitole 1.2, byly použity náměry proudové spotřeby měřené při šifrování s konstantním klíčem. Na rozdíl od datasetu používaného k učení neuronových sítí, který je měřený s náhodnými klíči. Oproti výchozímu datasetu vyžadovala korelační analýza náměry s vyšším průměrováním,



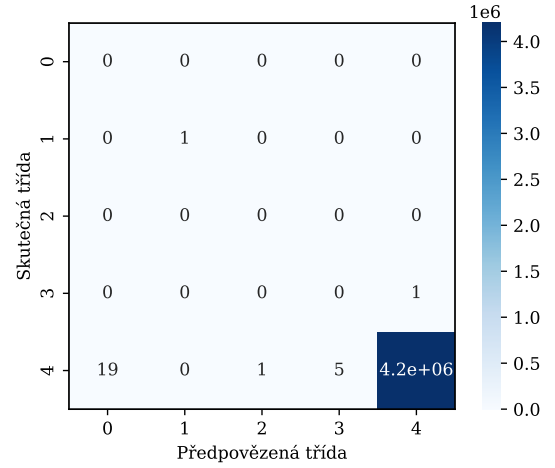
(a) S průměrováním 0.



(b) S průměrováním 2.



(c) S průměrováním 4.



(d) S průměrováním 1024.

Obr. 4.19: Matice záměn útoků modelů z kapitoly 4.2.1 pro různé průměrování náměrů na datasetu bez jitteru.

aby bylo možné určit pozice jednotlivých operací šifry. Toto může být výsledkem rozdílných architektur procesorů, na kterých šifrování probíhalo. Výchozí dataset byl naměřený na mikrokontroleru s osmibitovou architekturou. ARM procesor použitý v této implementaci využívá 32bitovou architekturu. Tento procesor je tedy schopný dělat více operací najednou, což může ztěžovat následnou analýzu. Okamžitá proudová spotřeba totiž není tak blízce vázána ke konkrétním operacím, když operací probíhá souběžně vícero. Na obr. 4.22 je k vidění porovnání CPA provedené na obou datasetech s průměrováním náměrů 4. Jednoznačné určení pozice Sboxu třetí rundy bylo možné až na náměrech s průměrováním 1024, jak je k vidění na obr. 4.21.

Tab. 4.9: Vyhledávací prostor pro ARM dataset bez zkrácení náměrů.

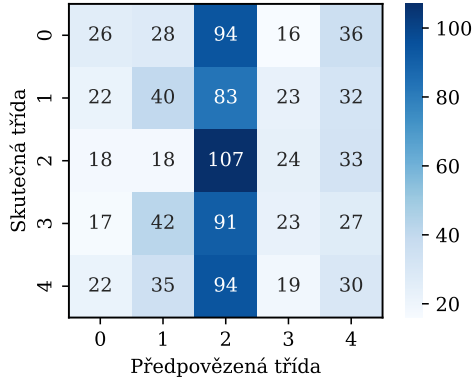
Hyperparametr	Hodnoty k prozkoumání
Konvoluční vrstvy	[4; 5; 6]
Konvoluční filtry	[4; 8; 12; 16]
Velikost konvolučního filtru	[4; 5; 6]
Aktivační funkce	[ReLU; SeLU]
Plně propojené vrstvy	[1; 2]
Neurony plně propojené vrstvy	[256; 512; 1024]
Velikost pooling matice	[3; 4; 5]
Krok pooling matice	[3; 4; 6]
Přidat L2 regularizaci	[Ano; Ne]
Hodnota L2 regularizace	[0,05; 0,1; 0,15]
Přidat regularizaci do plně propojených vrstev	[Ano; Ne]
Dropout	[0,3; 0,4; 0,5]
Optimalizátor učení	[Adam; RMSprop]

4.5 Návrh konvolučních neuronových sítí pro ARM dataset

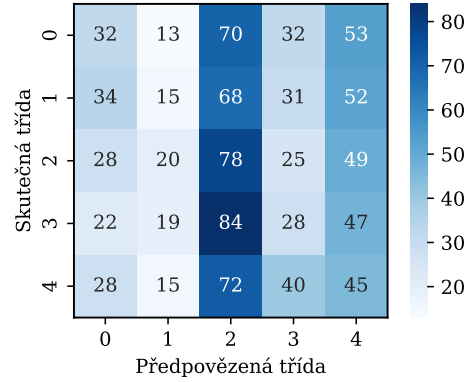
Podobně jako část 4.2 se tato sekce práce zabývá návrhem modelů hlubokého učení pro klasifikaci rozdílů zpráv naměřených na zařízení s ARM procesorem. K hledání modelů byl opět použit nástroj Keras Tuner.

Pomocí CPA byla určena opět část náměrů o délce 30 000 bodů, která obsahuje požadované kolize při šifrování třetí rundy. Prostor pro hledání hyperparametrů modelu byl lehce rozšířený oproti tomu použitému v minulé části. Tento prostor je k vidění v tab. 4.10, společně s nejlepšími hodnotami pro průměrování náměrů 1024. Nalezený model dosahuje přesnosti 91 % na testovacích datech a F1 skóre 0,92. Jeho matice záměn je k vidění na obr. 4.23a.

Stejný vyhledávací prostor byl použit i pro náměry s průměrováním 4. Žádný z nalezených modelů nepřesáhl 25% přesnost. Vyhledávání s rozšířenými hyperparametry pro hlubší síť také nepřineslo dobré výsledky. Jedním z možných odůvodnění může být, že stejně jako u CPA je na 32bitové architektuře průměrování 4 nedostatečující k oddělení hledaných operací v proudových náměrech od ostatních výpočtů. Tedy že hledané rozdílů v jednom bytu jsou až příliš zamaskované proudovou spotřebou jiných operací, což může dělat náměry neinterpretovatelnými.



(a) S průměrováním 512.



(b) S průměrováním 4.

Obr. 4.20: Matice záměn modelů na ARM datasetech s různým průměrováním před určením bodů zájmu.

ARM dataset s jitterem

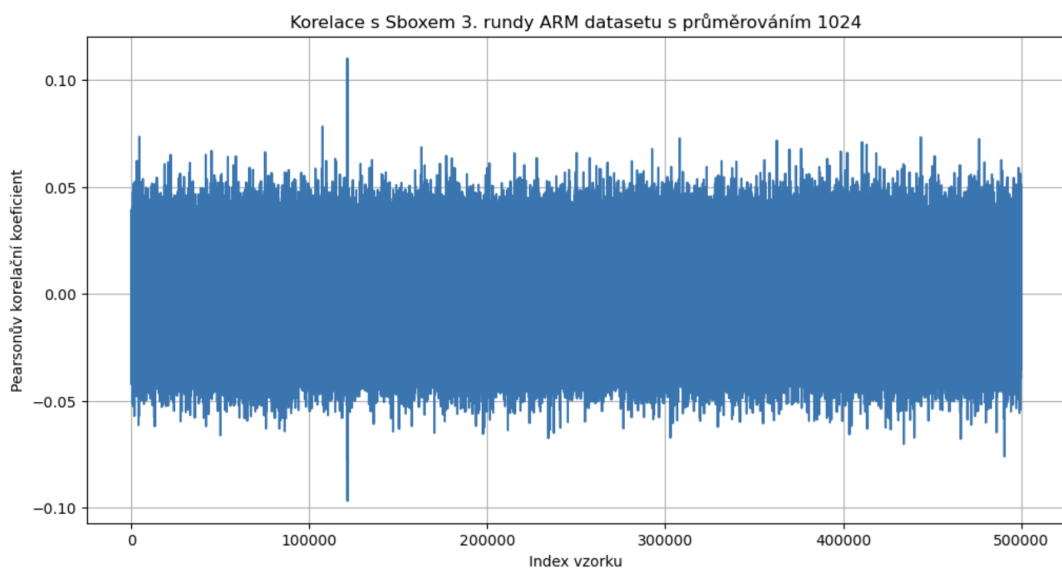
Stejně jako u předchozího datasetu byl do náměrů uměle přidán jitter pro vyzkoušení odolnosti neuronových sítí vůči špatně synchronizovaným náměrům. Jitter byl přidán v rozsahu ± 4 . Pro vyhledávání architektury pro tento dataset byl využit stejný prostor hyperparametrů jako je uveden v tab. 4.10, kde je také uvedena nalezená architektura modelu. Nalezený model dosahoval přesnosti 67,53 % a F1 skóre 0,68.

4.6 Útoky na ARM dataset

Útoky probíhaly obdobně jako v části 4.3. Datasety pro útok obsahují 2 901 jednotlivých náměrů, ze kterých jsou vytvořeny kombinace bez opakování. Tyto kombinace opět obsahují dvě kolize. Ve třídě 1 a ve třídě 3. Kvůli špatným výsledkům na náměrech s nižším průměrováním byly útoky provedeny pouze pro průměrování 1024.

Přestože při trénování dosahoval model natrénovaný na náměrech bez jitteru dobrých výsledků, při útoku nedokázal správně určit rozdíly zpráv kde došlo ke kolizi, jak je možné vidět na obr. 4.24a.

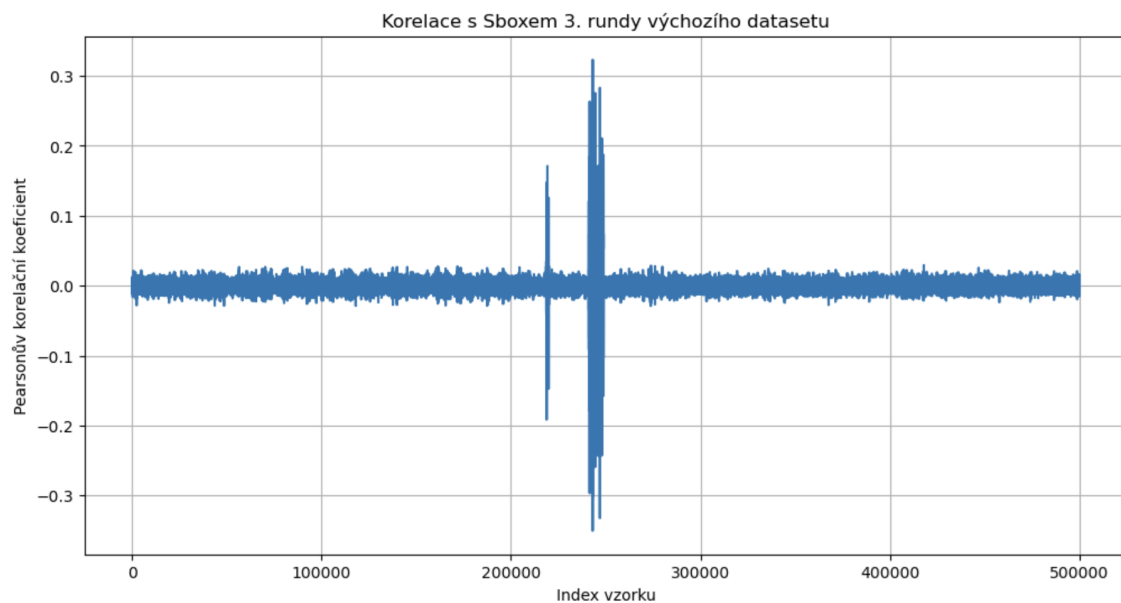
Model pro náměry s jitterem ± 4 byl schopný správně určit jednu ze dvou kolizí. Pokud nejsou modely samotné schopné určit kolize perfektně, měly by alespoň zjednodušit proces manuální analýzy rozdílů snížením potenciálních kandidátů na kolize. Ani k tomuto účelu není tento model vhodný, jelikož určil velké množství falešně pozitivních označení, jak je možné vidět na obr. 4.24b.



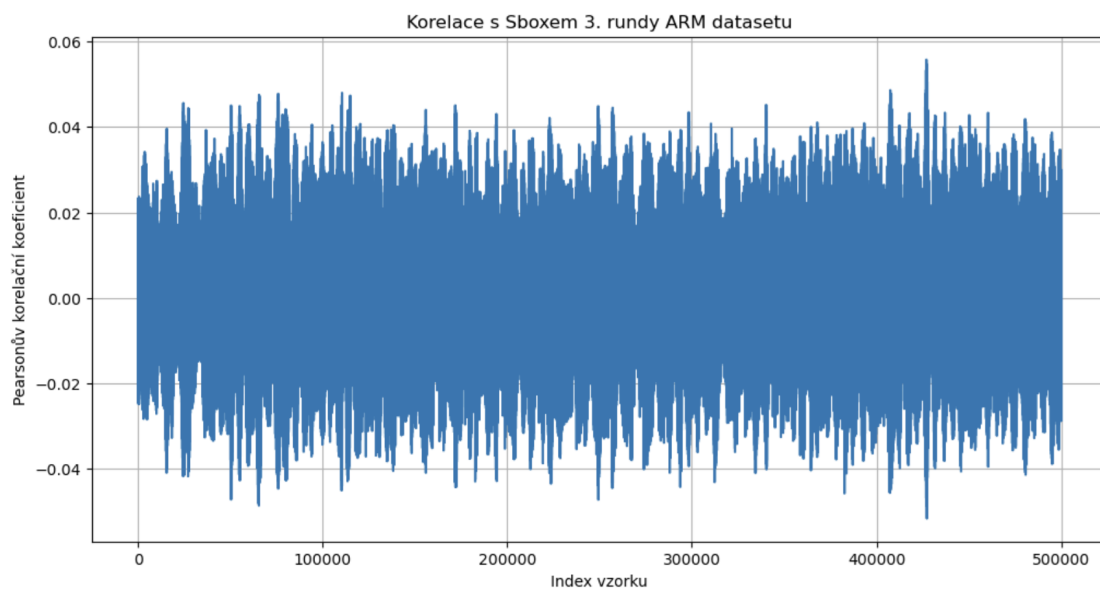
Obr. 4.21: CPA na ARM datasetu s průměrováním náměrů 1024.

Tab. 4.10: Vyhledávací prostor pro ARM dataset po zkrácení náměrů.

Hyperparametr	Hodnoty k prozkoumání	Hodnoty pro průměrování 1024	
		Bez jitteru	S jitterem ± 4
Konvoluční vrstvy	[4; 5; 6; 7]	5	6
Konvoluční filtry	[4; 8; 12; 14; 16]	12	12
Velikost konvolučního filtru	[4; 5; 6; 7]	5	6
Aktivační funkce	[ReLU; SeLU]	SeLU	SeLU
Plně propojené vrstvy	[1; 2]	2	2
Neurony plně propojené vrstvy	[256; 512; 1024]	512	1024
Velikost pooling matice	[2; 3; 4; 5]	4	4
Krok pooling matice	[2; 3; 4; 6]	3	3
Přidat L2 regularizaci	[Ano; Ne]	Ne	Ne
Hodnota L2 regularizace	[0,05; 0,1; 0,15]	0,15	0,1
Regularizace plně propojených vrstev	[Ano; Ne]	Ano	Ano
Dropout	[0,3; 0,4; 0,5]	0,4	0,3
Optimalizátor učení	[Adam; RMSprop]	Adam	Adam
Testovací přesnost [%]		91,00	67,53
Testovací F1 skóre		0,92	0,68

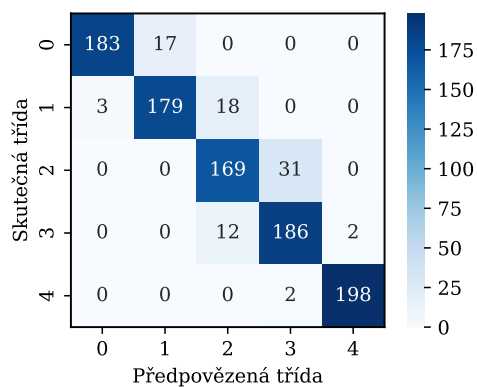


(a) Výchozí dataset.

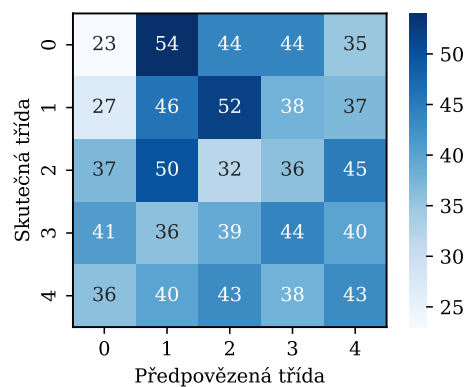


(b) ARM dataset.

Obr. 4.22: Porovnání CPA na výchozím a ARM datasetu s průměrováním náměrů 4.

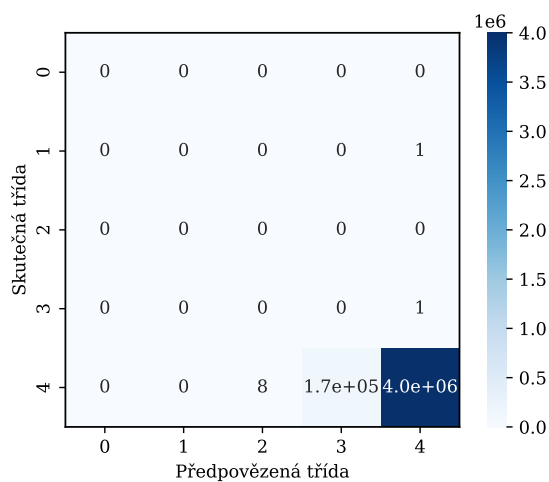


(a) Průměrování 1024.

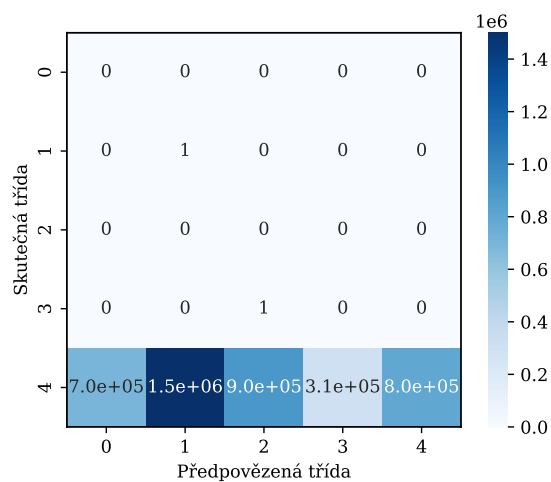


(b) Průměrování 4.

Obr. 4.23: Matice záměn pro modely trénované na ARM datasetu s různým průměrováním.



(a) Bez jitteru.



(b) S jitterem ± 4 .

Obr. 4.24: Matice záměn útoků na ARM dataset.

Závěr

Tato práce se zaměřila na použití hlubokého učení pro klasifikaci náměrů pro See-In-The-Middle útok na šifru AES-128. Bylo vyzkoušeno více druhů hlubokého učení ve snaze nalézt vhodné řešení.

Z provedených pokusů dosahovaly nejlepších výsledků konvoluční neuronové sítě. Prokázaly schopnost správně klasifikovat rozdíly zpráv i s malým průměrováním na datasetu měřeném na mikrokontroleru, a to i s určitou odolností vůči desynchronizaci náměrů. Při provedených útocích často určily velké množství falešných pozitiv. Proto by bylo vhodné prozkoumat možnosti použití prahových hodnot. Modely by poté klasifikovaly rozdíly jako kolize pouze při vyšší úrovni jistoty, což by mohlo napomoci redukovat počet falešných pozitiv.

Sítě nebyly schopné dosáhnout stejných výsledků na datasetu naměřeném na ARM procesoru, a také vyžadovaly vyšší úroveň průměrování. Toto je pravděpodobně spojené právě s odlišnou architekturou procesoru, která dělá okamžitou spotřebu procesoru méně závislou na jedné konkrétní operaci, kvůli současnému vykonávání více výpočtů.

Jelikož sítě často byly schopné správně určit rozdíly zpráv bez kolize, dalším možným krokem ve výzkumu by mohlo být využití binárního klasifikátoru k určení rozdílů bez kolize a s kolizí. Poté na jeho výstup navázat klasifikačním modelem podobným těm, které byly použity v této práci. Ty by se soustředily pouze na rozdělení kolizí do jednotlivých tříd. V této situaci, kdy finální klasifikace probíhala pouze do čtyř tříd kolizí, dosahovaly dobrých výsledků i předtrénované extraktory prvků z autoenkodérů.

Za další vhodný směr výzkumu lze považovat využití transformerových architektur, které byly v jiných studiích úspěšně použity k analýze postranních kanálů.

Veškeré stanovené cíle bakalářské práce byly splněny.

Literatura

- [1] Zdeněk Martinásek. *Kryptoanalýza postranními kanály [online]*. Doctoral theses, dissertations, Brno University of Technology Brno, 2013 [cit. 2024-12-01]. URL: <https://theses.cz/id/pnrkd6/>.
- [2] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. 1999.
- [3] Tomáš Gerlich, Jakub Breier, Pavel Sikora, Zdeněk Martinásek, Aron Gohr, Anubhab Baksi, and Xiaolu Hou. DL-SITM: Deep learning-based see-in-the-middle attack on AES. Cryptology ePrint Archive, Paper 2024/1389, 2024. URL: <https://eprint.iacr.org/2024/1389>.
- [4] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A Comprehensive Study of Deep Learning for Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020. doi:10.13154/tches.v2020.i1.348–375.
- [5] Neil Hanley, Maire O’Neill, Michael Tunstall, and William P. Marnane. Empirical evaluation of multi-device profiling side-channel attacks. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, October 2014. doi:10.1109/SiPS.2014.6986091.
- [6] Paul N. Fahn and Peter K. Pearson. IPA: A New Class of Power Attacks. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, pages 173–186, Berlin, Heidelberg, 1999. Springer. doi:10.1007/3-540-48059-5_16.
- [7] Yen-Ting Kuo and Atsushi Takayasu. A lattice attack on CRYSTALS-kyber with correlation power analysis. Cryptology ePrint Archive, Paper 2023/1781, 2023. URL: <https://eprint.iacr.org/2023/1781>.
- [8] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Differential Power Analysis. In *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, pages 119–165. Springer US, Boston, MA, 2007. doi:10.1007/978-0-387-38162-6_6.
- [9] Brojogopal Sapui, Jonas Krautter, Mahta Mayahinia, Atousa Jafari, Dennis Gnadt, Sergej Meschkov, and Mehdi B. Tahoori. Power Side-Channel Attacks and Countermeasures on Computation-in-Memory Architectures and Technologies. In *2023 IEEE European Test Symposium (ETS)*, pages 1–6, May 2023. doi:10.1109/ETS56758.2023.10173981.

- [10] Shivam Bhasin, Jakub Breier, Xiaolu Hou, Dirmanto Jap, Romain Poussier, and Siang Meng Sim. SITM: See-In-The-Middle-Side-Channel Assisted Middle Round Differential Cryptanalysis on SPN Block Ciphers, 2020. doi:10.13154/tches.v2020.i1.95–122.
- [11] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures – Profiling Attacks without Pre-Processing –, 2017. URL: <https://eprint.iacr.org/2017/740>.
- [12] Housseem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. Cryptology ePrint Archive, Paper 2016/921, 2016. URL: <https://eprint.iacr.org/2016/921>.
- [13] Amjed Abbas Ahmed, Mohammad Kamrul Hasan, Shayla Islam, Azana Hafizah Mohd Aman, and Nurhizam Safie. Design of convolutional neural networks architecture for non-profiled side-channel attack detection. *Elektronika ir Elektrotechnika*, 29(4):76–81, August 2023. doi:10.5755/j02.eie.33995.
- [14] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), February 2023. doi:10.1145/3569577.
- [15] Marina Krček, Lichao Wu, Guilherme Perin, and Stjepan Picek. Shift-invariance Robustness of Convolutional Neural Networks in Side-channel Analysis, 2023.
- [16] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL: <https://arxiv.org/abs/1511.08458>, arXiv:1511.08458.
- [17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [18] Suvadeep Hajra, Siddhartha Chowdhury, and Debdeep Mukhopadhyay. Estranet: An efficient shift-invariant transformer network for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(1):336–374, 2024. doi:10.46586/tches.v2024.i1.336–374.
- [19] Suvadeep Hajra, Sayandeep Saha, Manaar Alam, and Debdeep Mukhopadhyay. TransNet: Shift Invariant Transformer Network for Side Channel Analysis. In Lejla Batina and Joan Daemen, editors, *Progress in Cryptology - AFRI-CACRYPT 2022*, pages 371–396, Cham, 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-17433-9_16.

- [20] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4475–4483. PMLR, 13–18 Jul 2020. URL: <https://proceedings.mlr.press/v119/huang20f.html>.
- [21] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. [arXiv:1312.6114](https://arxiv.org/abs/1312.6114), doi:10.48550/arXiv.1312.6114.
- [22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL: <https://openreview.net/forum?id=Sy2fzU9gl>.
- [23] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. doi:10.1007/s13389-019-00220-8.
- [24] Tom O’Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. <https://github.com/keras-team/keras-tuner>, 2019.
- [25] L. Wyse. Audio Spectrogram Representations for Processing with Convolutional Neural Networks, June 2017. [arXiv:1706.09559](https://arxiv.org/abs/1706.09559), doi:10.48550/arXiv.1706.09559.
- [26] S. S. Stevens and J. Volkman. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940. URL: <http://www.jstor.org/stable/1417526>.
- [27] Benjamin Hettwer, Tobias Horn, Stefan Gehrler, and Tim Güneysu. Encoding Power Traces as Images for Efficient Side-Channel Analysis. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 46–56, December 2020. doi:10.1109/HOST45689.2020.9300289.
- [28] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15*, page 3939–3945. AAAI Press, 2015.
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

- [30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Image-net: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi:10.1109/CVPR.2009.5206848.
- [31] Mick G.D. Remmerswaal, Lichao Wu, Sébastien Tiran, and Nele Mentens. AutoPOI: Automated points of interest selection for side-channel analysis. Cryptology ePrint Archive, Paper 2023/008, 2023. URL: <https://eprint.iacr.org/2023/008>.
- [32] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
- [33] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. arXiv:1802.03426, doi:10.48550/arXiv.1802.03426.
- [34] GEEKS4GEEKS. *UMAP: Uniform Manifold Approximation and Projection*. Online. Geeks4Geeks. Dostupné z: <https://www.geeksforgeeks.org/umap-uniform-manifold-approximation-and-projection/> [cit. 2025-26-2].
- [35] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-Supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, January 2023. doi:10.1109/TKDE.2021.3090866.
- [36] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning, March 2021. arXiv:2004.11362, doi:10.48550/arXiv.2004.11362.