ŽILINSKÁ UNIVERZITA V ŽILINE

AUTOREFERÁT
DIZERTAČNEJ PRÁCE

Žilina, apríl 2019

Ing. Jakub Hrabovský

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

Ing. Jakub Hrabovský

Autoreferát dizertačnej práce

DETEKCIA SIEŤOVÝCH ÚTOKOV VO VYSOKO-RÝCHLOSTNÝCH POČÍTAČOVÝCH SIEŤACH

na získanie akademického titulu **„philosophiae doctor“** (**PhD.**)
v študijnom programe doktorandského štúdia

Aplikovaná informatika

v študijnom odbore
9.2.9 aplikovaná informatika

Žilina, apríl 2019

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia na Katedre informačných sietí, Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.

Predkladateľ:	Ing. Jakub Hrabovský Katedra informačných sietí Fakulta riadenia a informatiky Žilinská univerzita v Žiline
Školiteľ:	doc. Mgr. Ondrej Šuch, PhD. Katedra informačných sietí Fakulta riadenia a informatiky Žilinská univerzita v Žiline
Oponenti:	prof. Ing. Martin Klimo, PhD., Katedra informačných sietí Fakulta riadenia a informatiky Žilinská univerzita v Žiline
	doc. Ing. Ivan Kotuliak, PhD. Fakulta informatiky a informačných technológií Slovenská technická univerzita v Bratislave

Autoreferát bol rozoslaný dňa: **.....**

Obhajoba dizertačnej práce sa koná dňa **.....** o **.....** hod. pred komisiou pre obhajobu dizertačnej práce schválenou odborovou komisiou v študijnom odbore **9.2.9 aplikovaná informatika**, v študijnom programe **aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity
v Žiline dňa **.....**

prof. Ing. Karol Matiaško, PhD.
predseda odborovej komisie
študijného programu **aplikovaná informatika**
v študijnom odbore **9.2.9 aplikovaná informatika**

Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

Anotácia

Cieľom práce je vytvorenie metodiky návrhu detektora DoS/DDoS útokov s použitím strojového učenia vo vysoko-rýchlostnej počítačovej sieti. Práca analyzuje klady a nedostatky aktuálnych detekčných metód, ktoré sú založené na strojovom učení. Takto získané trendy sú následne aplikované pri tvorbe vlastnej metodiky návrhu detektora sieťových útokov. Predlohou špecifikácie jednotlivých etáp metodiky je oblasť rozpoznávania vzorov. Okrem metodiky sa práca zaoberá aj generickým návrhom systému konvolučnej neurónovej siete a jeho implementáciou do FPGA obvodu. Najväčšia pozornosť je venovaná návrhu originálnej štruktúry 2D konvolútora. Pre popis subsystémov, navrhnutých v tejto práci, je vytvorený grafický model v nástroji Matlab/Simulink a RTL model v jazyku VHDL. Korektná funkcia modelov je overená formou simulácie.

Kľúčové slová: sieťový útok, odopretie služby, systém detekcie sieťových prienikov, hlboké učenie, konvolučná neurónová sieť, fpga

Počet strán:	152	*Počet použitej literatúry:*	120
Počet obrázkov:	35	*Počet tabuliek:*	15

Annotation

The aim of the thesis is a methodology for a design of DoS/DDoS attacks detector with application of machine learning in high-speed computer network. The thesis analyzes pluses and minuses of current intrusion detection methods based on the principles of machine learning. Identified trends are subsequently applied during the creation of own methodology for a design of network intrusion detector. The field of pattern recognition serves as a template for a specification of the individual methodology stages. Beside the methodology, the thesis deals with a generic design of a convolutional neural network system and its implementation into FPGA circuit. The most attention is given to the novel structure of 2D convolver. The graphical model (built in development tools Matlab/Simulink) and RTL model (written in VHDL) were created in order to describe subsystems, designed in this thesis. The correct function of the models is verified and validated through the simulation.

Key words: network intrusion, denial of service, network-based intrusion detection system, deep learning, convolutional neural network, fpga

Number of pages:	152	*Number of used bibliographics:*	120
Number of figures:	35	*Number of tables:*	15

Úvod

Narastajúca úloha Informačno Komunikačných Technológií (IKT) v nových priemyselných oblastiach, ako sú Cloud Computing (CC) a Internet vecí (angl. *Internet of Things*; IoT), vytvára z počítačových sietí jeden z centrálnych prvkov infraštruktúry IKT. Počítačová sieť musí z dôvodu silnej závislosti IKT spĺňať požiadavky trvalej dostupnosti. Nedostupnosť siete môže spôsobiť bezpečnostné riziká, vysoké finančné straty a predstavuje hrozbu v kritických oblastiach nasadenia, ako sú zdravotníctvo a energetika. Trvalá dostupnosť počítačovej siete je dominantnou úlohou **počítačovej bezpečnosti**.

Z pohľadu bezpečnosti čelí počítačová sieť v reálnom prostredí mnohým masívnym útokom, ktoré bránia v jej dlhodobom neprerušenom používaní. K najrozšírenejším príkladom ničivých sieťových útokov patria **odopretie služby** (angl. *Denial of Service*; DoS) a **distribuované odopretie služby** (angl. *Distributed DoS*; DDoS). Uvedené útoky zásadne znižujú kvalitu sieťových služieb. Obmedzenie ich účinkov vyžaduje riešiť otázky ich skorej detekcie a následnej reakcie s cieľom minimalizovať celkové spôsobené škody.

Po zvážení požiadaviek na súčasné a budúce systémy detekcie sieťových prienikov (angl. *Network-based Intrusion Detection System*; NIDS), je v predkladanej práci akcentovaná aplikácia strojového učenia (angl. *Machine Learning*; ML). Práca sa tak venuje problematike detekcie sieťových prienikov, konkrétne DoS/DDoS s použitím metód strojového učenia. V práci sú zhrnuté trendy v návrhu detekčných mechanizmov, z ktorých následne vychádza nami navrhnutá metodika pre návrh detektora sieťových útokov. Súčasťou práce je návrh subsystémov konvolučnej siete, implementovateľných do FPGA (angl. *Field Programmable Gate Array*; FPGA) obvodu. Pre overenie funkcie subsystémov sú vytvorené viaceré modely. Práca popisuje priebeh a výsledky ich simulácie.

1 Aktuálny stav zvolenej problematiky

Vzhľadom na dôležitosť komunikačných sietí, narastajúce nebezpečenstvo DoS a DDoS útokov, a nedostatky súčasných detekčných metód, je návrh nových, účinnejších metód nutným krokom pre zachovanie dostupnosti IKT. Podľa analýzy viacerých detekčných metód, dostupných vo vedeckých článkoch, je vytvorený prehľad trendov pre návrh podľa nášho názoru účinnejšej detekčnej metódy.

1.1 Útoky odopretia služby

Útoky typu DoS a DDoS patria medzi najznámejšie sieťové útoky, ktoré sú schopné čiastočne alebo úplne zastaviť cieľovú sieťovú službu [1], [2]. Napriek intenzívnemu výskumu v oblasti sieťovej bezpečnosti, účinok týchto útokov narastá každý rok, čo potvrdzujú mnohé oficiálne správy [3]–[5].

Hlavným cieľom DoS/DDoS je priame alebo nepriame vyčerpanie sieťových (šírka pásma), pamäťových (pevný disk, operačná pamäť) a výpočtových (procesor) zdrojov na strane obete zámernými aktivitami útočníka. Postihnutá služba je čiastočne alebo úplne nedostupná, čím výrazne klesá jej kvalita. Obeťou môže byť **koncové zariadenie** (klient, server), **medziľahlý komunikačný uzol** (smerovač, prepínač) alebo aj samotný **komunikačný kanál**.

1.2 Detekcia prienikov

Súčasný výskum a dostupné riešenia eliminácie útokov na počítačovú sieť (kap. 1.1) navrhujú štyri prístupy: **detekcia**, **prevencia**, **potlačenie** (angl. *mitigation*) a **reakcia** (angl. *response*) na útoky [6]. Skorá detekcia môže znížiť účinok útoku na kvalitu služby, pretože vedie ku okamžitej reakcii. Princíp metód detekcie prienikov spočíva v porovnávaní získaných dát z monitorovaného prostredia s vytvoreným modelom. Model slúži ako referenčný vzor v procese rozlíšenia útoku od normálnej sieťovej prevádzky. Konečné rozhodnutie metódy závisí od stanovenej podobnosti medzi odchytenými dátami a vzorom, a od správania, ktoré model reprezentuje. Taxonómia metód detekcie prienikov sa diferencuje v sledovaných parametroch, ako sú typ spracovaných dát a miesto nasadenia metódy v sieti [7], [8].

1.2.1 Klasifikácia detektorov sieťových prienikov podľa prístupu k analýze dát

Podľa toho, aké správanie popisuje detekčnou metódou vytvorený model, sa metódy IDS delia do dvoch tried: **signatúrne** (angl. *signature-based*, známe aj ako misuse-/knowledge-based) a **anomálne** (angl. *anomaly-based*, známe aj ako behavior-based) [7].

Anomálne metódy Anomálne metódy tvoria modely normálnej prevádzky bez prítomnosti útokov. V procese tvorby modelu sú použité informácie o aktivitách zariadení v monitorovanej sieti za účelom vytvorenia presnej reprezentácie prostredia v jeho normálnom stave. Metóda identifikuje anomáliu ako výraznú odlišnosť od normálnej sieťovej aktivity, určenej modelom.

Pre dosiahnutie použiteľných výsledkov prebieha neustála dynamická aktualizácia referenčného modelu. Pre tento účel existujú rôzne sady vzoriek sieťovej prevádzky, nazývané **datasety**, a algoritmy použiteľné pre spracovanie veľkých dát (angl. *big data*), ktoré podporujú proces prisôsobenia modelu na zvolenú úlohu. Tento proces úpravy modelu sa nazýva učenie alebo aj trénovanie (angl. *learning/training*). Kvalita trénovania modelu závisí na použitom učiacom algoritme, a na veľkosti a kvalite trénovacieho datasetu.

Strojové učenie Vysoká zložitosť súčasných sieťových systémov neumožňuje vytvoriť ich exaktný model. Metódy strojového učenia prinášajú možnosť

vytvoriť približný model len na základe vstupných vzoriek bez znalosti interného správania systému. Model spoznáva nové vzory zo vstupných vzoriek tak, aby v budúcnosti identifikoval aj k nim podobné alebo upravené vzorky. Táto schopnosť sa nazýva *generalizácia* a je dôležitou vlastnosťou ML, najmä v prípade detekcie anomálií. Iteratívny proces nepretržitého učenia zdokonaľuje kvalitu modelu a jeho výstupov.

Niektoré metódy ML sa ukázali ako vhodné pre implementáciu detekcie sieťových anomálií: Bayesovské siete (angl. *Bayesian Networks*; BN), podporné vektory (angl. *Support Vector Machines*; SVM) [9], umelé neurónové siete (angl. *Artificial Neural Networks*; ANN), a samo-organizujúce mapy (angl. *Self-Organizing Maps*; SOM) [10].

Prehľad výskumnej činnosti BN v oblasti NIDS Patel a Buddhadev sa v [11] zaoberajú teóriou BN. Autori uprednostňujú použitie hybridnej metódy, pozostávajúcej z viacerých jednoduchých modelov BN, z ktorých sa každý model špecializuje na konkrétny typ útoku. Článok zdôrazňuje výhodu BN skombinovať počiatočné znalosti z domény a proces trénovania pre zlepšenie celkových výsledkov. Vijayasarathy [12] použil Bayesové učenie, konkrétne NB klasifikátor, pre vytvorenie modelu reálnej sieťovej prevádzky s cieľom detegovať sieťové útoky DDoS. Vijayasarathy vytvoril niekoľko modelov s rôznou kombináciou príznakov s cieľom zvoliť ich najlepšiu podmnožinu. Podobný prístup implementácie hybridnej metódy použili aj G. Kumar a K. Kumar v [13]. V článku bola použitá technika skladania jednoduchých modelov (angl. *ensemble technique*), konkrétne ide o kombináciu genetického algoritmu a skupiny samostatných modelov NB. V kontraste s predchádzajúcim článkom, riešili autori problém optimálneho výberu príznakov ako samostatnú optimalizačnú úlohu. Článok zdôrazňuje zlepšenie celkových výsledkov a univerzálnosť zloženej metódy vďaka spolupráci rôznych prístupov (GA a NB).

Prehľad výskumnej činnosti SVM v oblasti NIDS Chandola a spol. [14] navrhli vizuálny model SVM regiónov, prislúchajúcich k profilom rôznych typov sieťovej prevádzky. Osareh a Shadgar [15] porovnávajú modely SVM a ANN pri riešení úlohy detekcie sieťových prienikov. Podľa dosiahnutých výsledkov viacerých v článku implementovaných modelov, dosahuje metóda SVM lepšie výsledky (okrem presnosti) ako modely ANN za predpokladu, že podiel normálnej prevádzky v datasete je nad 50%. V prípade reálnych počítačových sietí je táto podmienka štandardne splnená, a tak prezentované vlastnosti významne podporujú nasadenie SVM v doméne detekcie sieťových prienikov. Kim a spol. [16] prišli s hybridnou metódou, ktorá pozostáva zo signatúrnej (rozhodovací strom C4.5) aj anomálnej (one-class SVM) detekčnej techniky. Pre každý z listov rozhodovacieho stromu je vytvorený samostatný model SVM, ktorý reprezentuje konkrétny sieťový profil. Kvôli modulárnej štruktúre a špecializácii sú modely SVM oveľa jednoduchšie, používajú len malý počet podporných vektorov, a tak

výrazne zlepšujú celkový výkon. Článok zdôrazňuje výhody hierarchickej hybridnej metódy v porovnaní s konvenčnými metódami, ktoré vedú k vyššej detekčnej presnosti, a k rýchlejšej fáze trénovania a testovania. She a spol. [17] použili model, založený na one-class SVM, ako reprezentáciu normálneho správania HTTP klientov s cieľom detegovať aplikačné DDoS útoky. Článok je demonštráciou toho, ako SVM pomáha analyzovať dáta aplikačnej vrstvy.

Prehľad výskumnej činnosti ANN v oblasti NIDS Alfantookh [18] predstavil jeden z prvých modelov ANN so spätným šírením chyby (BP-ANN) v úlohe detektora sieťových prienikov. Medzi vylepšenia, ktoré model prináša, patria hierarchické viacúrovňové spracovanie vstupných dát vo forme postupnej extrakcie príznakov a pridanie tretej, imaginárnej triedy na vyjadrenie určitej miery nerozhodnosti. Osareh a Shadgar [15] porovnali model BP-ANN s modelom SVM. Tang [19] popisuje model hlbokej ANN s vyladenými hyperparametrami pre detekciu DDoS útokov. Metóda využíva výhody hlbokého učenia, ktoré umožňujú spájať skupiny jednoduchých príznakov do skrytých zložených príznakov. Wei a spol. [20] zlepšili presnosť a rýchlosť konvergencie pôvodného modelu BP-ANN použitím špecifického optimalizačného algoritmu – *Particle Swarm Optimization* (PSO). Metóda LVQ, uvedená v [21], je ďalší príklad metódy ANN v úlohe detektora sieťových útokov (DDoS útokov na webovú službu – HTTP) s lepšou presnosťou ako v prípade tradičných modelov BP-ANN.

Prehľad výskumnej činnosti SOM v oblasti NIDS Patel a Buddhadev [11] považujú neriadené učenie v metóde SOM ako podstatnú výhodu vzhľadom na zložitosť, chybovosť a časovú náročnosť procesu značkovania sieťových vzoriek. Článok [22] predstavuje vylepšenú verziu SOM – Emergent SOM (ESOM). Mitrokotsa a Douligeris uvádzajú niektoré prínosy SOM v oblasti detekcie DoS útokov: vizualizácia sieťovej prevádzky, paralelné spracovanie, a analýza v reálnom čase. Ich metóda ESOM zlepšuje mieru detekcie vďaka vloženiu väčšieho počtu neurónov do topologickej mapy. Výhody metódy SOM, samoučenie a detekcia nových útokov, sa v [23] stali inšpiráciou pre implementáciu IDS, založeného na SOM. Wang a Yu vytvorili IDS z troch modelov SOM, ktoré pracujú na rôznej úrovni: systém, proces, a sieť. Kim a Park [24] zvolili pre implementáciu NIDS v súčasnosti často používaný prístup distribuovaného spracovania. Ich metóda – distribuovaná SOM (DSOM) – rozkladá zložitosť detekcie medzi viacero jednoduchých modelov SOM, umiestnených v sieťovej infraštruktúre. Takéto zaobchádzanie so sieťovou prevádzkou rieši všeobecný problém centrálne orientovaného NIDS – škálovateľnosť. Modely sú trénované v priebehu reálnej prevádzky. Distribuovaný prístup umožňuje pracovať aj s veľkým objemom sieťovej prevádzky, eliminuje zahltenie a oneskorenie IDS, a dosahuje rovnako dobrú presnosť ako NIDS implementácie, založené na metóde tradičnej SOM.

1.2.2 Trendy v návrhu systémov pre detekciu sieťových prienikov

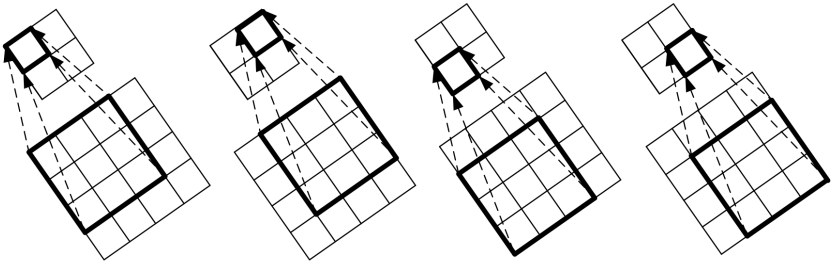
Vykonaná analýza a porovnanie metód NIDS, popísaných v predchádzajúcich podkapitolách, vedú k potenciálnym vylepšeniam a návrhovým trendom. Niektoré z nich (najpodstatnejšie z nášho pohľadu) sú zhrnuté v nasledujúcom zozname:

- **Viac-úrovňové predspracovanie vstupných dát** - Nepretržité čistenie a opracovávanie dát umožňuje extrahovať hlbšie doménovo špecifické príznaky, ktoré vedú k odhaleniu skrytých súvislostí v dátach. Výskumné oblasti, ako je spracovanie obrazu a prirodzeného jazyka, používajú modely s hlbokou štruktúrou (hlboké neurónové siete) pre zvýšenie abstraktnej úrovne skrytých príznakov.
- **Prechod od tradičných ku hybridným metódam** - Prístup hybridných metód eliminuje nedostatky riadeného a neriadeného učenia ich vzájomnou kooperáciou (tieto prístupy učenia sa navzájom dopĺňajú). Špecializácia umožňuje jednotlivým častiam modelu riešiť úlohy, v ktorých excelujú.
- **Automatizovaný výber príznakov** - Úloha výberu príznakov je dôležitým krokom ľubovoľnej metódy strojového učenia bez ohľadu na jej aplikačnú doménu. Úloha výberu príznakov je často braná ako individuálny problém, ktorý je možné taktiež vyriešiť aplikáciou strojového učenia.
- **Reálna sieťová prevádzka pre trénovanie**
- **Distribuovaný výpočet** - Metóda, zložená z viacerých elementárnych modelov, umožňuje vďaka paralelným výpočtom spracovanie v reálnom čase. Sieť detektorov (spolupracujúce modely, ktoré poskytujú spoločné výstupy) súčasne zjednodušuje adaptáciu zloženého modelu na rozsiahlu, zložitú infraštruktúru.
- **Grafický formát výstupných údajov a medzivýsledkov** - Vizualizácia podporuje lepšie pochopenie správania a princípov použitého algoritmu, a poskytuje ďalšiu formu výstupov.
- **Aktualizácia modelu v reálnom čase** - Dynamické prostredie súčasných počítačových sietí vyžaduje učenie v reálnom čase, aby model dostatočne rýchlo reagoval na nepravidelné zmeny v správaní siete.

1.3 Konvolučná neurónová sieť

Konvolučná neurónová sieť [25]–[27] vychádza z klasickej doprednej umelej neurónovej siete a preto preberá väčšinu jej základných princípov štruktúry, trénovania aj inferencie. Konvolučná sieť je postavená na nasledujúcich troch princípoch: **zdieľanie váh**, **lokálne receptívne pole** (angl. *receptive field*) a **podvzorkovanie** [28]. Zdieľanie váh spočíva v doplnení obmedzujúcich podmienok pre

návrh synapsií medzi neurónmi – synapsie v rámci skupiny neurónov musia použiť rovnaké váhy. Následkom zdieľania je podstatná redukcia počtu voľných parametrov pri zachovaní počtu synapsií ovplyvňujúcich schopnosť konvolučnej siete. Skupina neurónov, ktoré používajú pre výpočet výstupov rovnaký váhový vektor, generujú príznakovú mapu (angl. *feature map*). Proces výpočtu predstavuje skenovanie vstupnej mapy metódou posuvného okna s cieľom nájsť všetky výskyty lokálneho príznaku. Obrázok 1 ilustruje metódu použitia posuvného okna na vstupnú mapu.



Obr. 1: Skenovanie mapy pomocou filtra – posuvné okno o veľkosti $\mathbf{3 \times 3}$ (silne zvýraznené políčka) postupne prechádza vstupnou mapou (skupina políčok tvoriacich štvorec s rozmermi $\mathbf{4 \times 4}$ umiestnená nižšie) v smere zľava doprava; výsledné hodnoty získané postupným prekryvaním okna s rôznymi časťami vstupnej mapy sú v tom istom poradí umiestnené na výstupnej mape (skupina políčok s rozmermi $\mathbf{2 \times 2}$ umiestnená vyššie).

Aby bola sieť minimálne citlivá na rotáciu a skreslenie vzorov vo vstupných obrazoch, musí po detekcii výskytov príznakov postupne nahrádzať ich presné pozície za približné, t. j. vykonať aproximáciu susedných bodov príznakovej mapy. Táto aproximácia je realizovaná operáciou zlučovania, ktorá nahrádza skupiny susedných bodov v mape ich zástupcom. Všetky vstupné mapy, ktoré získame permutáciou pixelov v rámci jedného okna, budú prislúchať jednej a tej istej výstupnej mape. Operáciou zlučovania budú všetky lokálne rozdiely v natočení konkrétneho vzoru a jeho skreslenie namapované na jednu spoločnú mapu. Konvolučná sieť je schopná s použitím adaptívnych metód súčasne vykonať extrakciu príznakov aj následnú klasifikáciu [28].

1.3.1 Typy vrstiev

Konvolučná vrstva Úlohou konvolučnej vrstvy je extrakcia rôznych príznakov zo vstupnej príznakovej mapy.

$$\boldsymbol{Y}^{l}_{i,j} = \boldsymbol{b}^{l} + \sum_{h=1}^{H} \sum_{m=1}^{K} \sum_{n=1}^{K} \boldsymbol{X}^{h}_{i+m,j+n} \times \boldsymbol{W}^{h}_{m,n} \tag{1}$$

Matematický výraz uvedený v (1) určuje spôsob spracovania konvolučnej vrstvy,

kde $\boldsymbol{X}^{\boldsymbol{h}}_{\boldsymbol{i+m,j+n}}$ je bod na pozícii $\boldsymbol{(i+m, j+n)}$ v $\boldsymbol{h}$-tej vstupnej mape, podobne $\boldsymbol{Y}^{\boldsymbol{l}}_{\boldsymbol{i,j}}$ je bod na pozícii $\boldsymbol{(i, j)}$ v $\boldsymbol{l}$-tej výstupnej mape, $\boldsymbol{W}^{\boldsymbol{h}}_{\boldsymbol{m,n}}$ je koeficient na pozícii $\boldsymbol{(m, n)}$ v kerneli s rozmermi $\boldsymbol{K} \times \boldsymbol{K} \times \boldsymbol{H}$ použitom pre $\boldsymbol{h}$-tú vstupnú mapu a $\boldsymbol{b}^{\boldsymbol{l}}$ je bias pre $\boldsymbol{l}$-tú výstupnú mapu.

Zlučovacia vrstva Zlučovacia vrstva (angl. *pooling*) vykonáva operáciu zlučovania (2). Táto operácia je vo svojej podstate rovnaká ako v prípade konvolučnej vrstvy. Rozdiel spočíva len vo funkcii, ktorú použijeme nad skupinou bodov v lokálnom okolí (v okne). V prípade zlučovacej vrstvy sú najpoužívanejšie funkcie: priemer a maximum. Zlučovanie vedie ku zmenšeniu rozmerov príznakových máp na ďalších vrstvách, zníženiu počtu synapsii a voľných parametrov, a následne aj k zníženiu pamäťových a výpočtových nárokov.

$$\boldsymbol{Y}^{\boldsymbol{l}}_{\boldsymbol{i,j}} = \boldsymbol{f}(\boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i,j}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i+1,j}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i,j+1}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i+1,j+1}}) \tag{2}$$

Plne-prepojená vrstva Plne-prepojená vrstva sa správaním zhoduje s vrstvami klasickej umelej neurónovej siete. Preto je formát vstupných dát jednorozmerný vektor. Správanie tejto vrstvy je matematicky popísané v (3)

$$\boldsymbol{Y_i} = \vec{\boldsymbol{X}} \cdot \vec{\boldsymbol{W}}^{\intercal}_{\boldsymbol{i}} + \boldsymbol{b_i} \tag{3}$$

ako skalárny súčin vstupného vektora $\vec{\boldsymbol{X}}$ a váhového vektora $\vec{\boldsymbol{W}}_{\boldsymbol{i}}$ prislúchajúceho neurónu, pripočítaný k jeho biasu $\boldsymbol{b_i}$. V porovnaní s konvolučnou vrstvou existuje v tomto prípade synapsia medzi každou dvojicou neurónov susedných vrstiev, teda nie je aplikovaný princíp lokality. Plne-prepojená vrstva slúži ako klasifikátor, kde vstupný vektor reprezentuje vektor zložitých príznakov (extrahovaných v predchádzajúcich vrstvách).

Aktivačná vrstva Pre zachytenie nelineárnych závislostí vo vstupnom obraze slúži aktivačná vrstva. Ide o aplikáciu vhodnej nelineárnej funkcie na body príznakových máp. Funkcia je aplikovaná samostatne na každý bod mapy. Efektívnou aktivačnou funkciou je rektifikovaná lineárna jednotka (angl. *Rectified Linear Unit*; ReLU) (4). ReLU zrýchľuje trénovanie siete zjednodušením výpočtov a súčasne prináša lepšie výsledky.

$$\textbf{ReLU}\,(\boldsymbol{x}) = \textbf{max}\,(\boldsymbol{0}, \boldsymbol{x}) \tag{4}$$

1.3.2 Aplikácie konvolučnej siete na reálne úlohy

Najznámejšie modely, ktoré boli ako prvé oficiálne pomenované konvolučné siete, ponúkli riešenia takých praktických problémov ako je automatizované rozpoznávanie rukou písaných číslic [28]. Vďaka úspechu týchto architektúr sa konvolučné siete uchytili aj v iných oblastiach ako je spracovanie zvuku a časových radov [25], [29]. Vďaka rýchlemu rozvoju hardvéru sú konvolučné siete v súčasnosti

nasadené do rôznych tak odlišných oblastí ako sú **medicína** (spracovanie medicínskych snímok pre detekciu chorôb [30], [31]), **bezpečnosť počítačových systémov** (detekcia škodlivých programov [32]), **spracovanie signálov produkovaných nervovými bunkami mozgu** (EEG a detekcia P300 [33]) a iné. Vzhľadom na široké nasadenie konvolučných sietí a ich pozitívne výsledky sme sa rozhodli použiť konvolučnú sieť v oblasti **bezpečnosti počítačových sietí**.

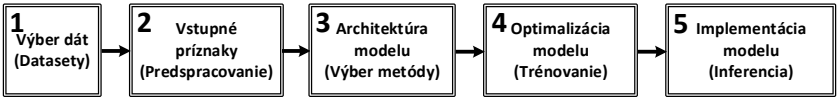
2 Ciele práce

Hlavným cieľom predkladanej práce je **vytvorenie metodiky návrhu detektora DoS/DDoS útokov na báze analýzy paketových tokov s použitím strojového učenia vo vysoko-rýchlostnej počítačovej sieti**. Práca sa zameriava na **hlboké neurónové siete**, ktoré predstavujú v súčasnosti najsľubnejší prístup strojového učenia.

Kroky pre dosiahnutie uvedeného cieľa práce sú: návrh etáp metodiky, výber vhodných technických prostriedkov pre implementáciu zvolenej etapy metodiky, návrh architektúry konvolučnej neurónovej siete (odpovedá zvolenej etape) s cieľom aplikácie FPGA obvodov, experimentálne overenie a vyhodnotenie.

3 Metodika návrhu detekčnej metódy

Sekvencia etáp navrhovanej metodiky je ilustrovaná na obr. 2. Navrhovaná metodika nadväzuje na prax v rozpoznávaní vzorov vzhľadom na súvislosť medzi úlohami rozpoznávania vzorov a aplikáciou konvolučnej siete na detekciu sieťových anomálií [34].



Obr. 2: Sekvencia etáp navrhovanej metodiky

3.1 Dáta - Výber a analýza dostupných datasetov

Úspešnosť modelu hlbokého učenia do značnej miery závisí od dostupných trénovacích a testovacích dát. Zameraním tejto práce je oblasť detekcie prienikov v sieťovej prevádzke, preto sa ďalej venujeme datasetom z tejto domény.

3.1.1 KDDcup99

KDDcup99 [35] bol niekoľko rokov najpoužívanejší dataset pre ohodnotenie detekčnej metódy a jej porovnanie s inými metódami formou benchmarku. Dataset

obsahuje normálne a útočné vzorky, ktoré sú rozdelené do štyroch hlavných kategórií útokov: DoS, R2L (Remote-to-Local), U2R (User-to-Root), a Probing. Detailný popis útokov je dostupný v [36].

Každá vzorka obsahuje 41 rôznych príznakov a finálne označenie typu spojenia (normálne alebo útok spolu s konkrétnym typom). Príznaky sú rozdelené do troch kategórií: **základné** informácie vychádzajúce z TCP/IP, informácie o **prevádzke** počítané cez posuvné okno podľa dvoch spôsobov zoskupovania spojení: "same host" a "same service", informácie o **obsahu** sledujúce niektoré vlastnosti spojení a vychádzajúce z obsahu paketov skrytých v aplikačných dátach najmä kvôli útokom typu U2R a R2L.

3.1.2 NSL-KDD

NSL-KDD [37] je vylepšená verzia KDDcup99, ktorá rieši viaceré nedostatky pôvodného datasetu, ako sú veľká duplicita vzoriek najmä v prípade niektorých typov útokov, nedostatok celkového počtu dostupných vzoriek, a veľmi nerovnomerné zastúpenie rôznych tried v trénovacej a testovacej sade. NSL-KDD upravuje pomer ťažko detegovateľných a ľahko detegovateľných vzoriek v oboch sadách tým, že z originálnej KDDcup99 sady vyberá náhodne len určitý počet záznamov podľa toho, ako úspešné boli rôzne metódy pri ich klasifikácii.

3.1.3 ISCX-2012

ISCX-2012 [38] je dataset sieťovej prevádzky obsahujúci okrem normálnych tokov aj rôzne formy útokov vrátane DDoS. Pre dosiahnutie realistickej a automatizovanej tvorby sieťových datasetov existujú v prípade ISCX-2012 datasetu dva typy profilov s popisom správania normálnych používateľov a útočníkov: α-profily popisujú útoky, β-profily reprezentujú normálnu prevádzku. Zvolené charakteristiky α-profilov vychádzajú z analýzy scenárov rôznych typov útokov. β-profily vychádzajú z analýzy reálnej sieťovej prevádzky, ktorá obsahuje toky štandardných sieťových služieb, ako sú HTTP, SMTP, SSH, IMAP, POP3, a FTP. Následné generovanie sieťovej prevádzky je úlohou automatizovaných agentov, ktorých správanie je určené α- a β-profilmi. ISCX-2012 dataset poskytuje všetky odchytené správy vo formáte *pcap*. Výhodou datasetu ISCX-2012 oproti predchádzajúcim uvedeným datasetom je množstvo použiteľných vzoriek, aktuálne typy útokov, a neobmedzenosť výberu príznakov, ktoré sú vďaka formátu *pcap* voľne extrahovateľné.

3.2 Vstupné príznaky

Pre praktické použitie vzoriek z datasetu je prvým krokom výber príznakov, použitých ako vstupy do modelu. V prípade datasetov s malým počtom dostupných príznakov (kap. 3.1.1 a kap. 3.1.2) môžeme použiť všetky príznaky bez

prvotnej analýzy. Ale ak sú vzorky dostupné vo formáte, ktorý priamo neposkytuje konkrétnu množinu príznakov (napr. *pcap* formát uvedený v kap. 3.1.3), je voľba príznakov nejasná a vyžaduje manipuláciu so vzorkami so zámerom ich predspracovania.

3.2.1 Predspracovanie

Predspracovanie sieťovej prevádzky pred jej použitím v detekčnej metóde značne zlepšuje presnosť detekcie, ale vyžaduje hlbšie doménové znalosti [39]. Predspracovanie zahŕňa čistenie, škálovanie a normalizáciu dát, výber, extrakciu a redukciu príznakov, a diskretizáciu symbolických príznakov. Predspracovanie tak chápeme ako transformáciu odchytených paketov zo sieťovej prevádzky na postupnosť upravených vzoriek, ktoré sú vhodné pre danú detekčnú metódu.

3.2.2 Výber príznakov

Výber priamych príznakov zo sieťovej prevádzky je podmienený snahou o dosiahnutie najväčších možných rozdielov v hodnotách príznakov získaných z normálnej prevádzky a z útokov. Vychádzajúc z [40] je tok vhodnejším objektom analýzy, keďže zachytáva vzájomné vzťahy medzi komunikujúcimi stranami v čase.

Unikátnym identifikátorom toku je n-tica základných primárnych príznakov: zdrojové a cieľové IP adresy, zdrojové a cieľové porty, a protokol. Kvôli časovej synchronizácii tokov je k uvedeným primárnym príznakom priložená aj časová pečiatka. Okrem toho sú dostupné aj sekundárne príznaky špecifické len pre konkrétny protokol (TCP, UDP, a ICMP). Na zachytenie časových závislostí majú pozitívny efekt časové príznaky, ktoré sú vypočítané agregáciou (súčtom) alebo spriemerovaním primárnych príznakov v rámci časového okna. Pre zachytenie krátkodobých aj dlhodobých udalostí je vhodné použiť viacero okien s rôznou šírkou.

3.2.3 Čistenie, škálovanie a normalizácia dát

Príznaky použité v prípade sieťovej prevádzky majú vzájomne veľmi odlišné rozsahy hodnôt. Vzhľadom na detekčné metódy a ich spôsob spracovania vzoriek majú tak oveľa väčšiu váhu tie príznaky, ktoré nadobúdajú väčšie hodnoty. Pre odstránenie nerovnosti medzi rôznymi príznakmi sa ich hodnoty transformujú do intervalu $\mathbf{(0, 1)}$. Tento proces sa nazýva *Min-Max* škálovanie (5).

$$\boldsymbol{X}_{\text{norm}} = \frac{\boldsymbol{X} - \boldsymbol{X}_{\mathbf{min}}}{\boldsymbol{X}_{\mathbf{max}} - \boldsymbol{X}_{\mathbf{min}}} \tag{5}$$

Alternatívou je štandardizácia alebo tzv. *Z-score* normalizácia (6), kde $\bar{\boldsymbol{X}}$ je stredná hodnota príznaku $\boldsymbol{X}$ a $\boldsymbol{\sigma_X}$ je jeho štandardná odchýlka.

$$\boldsymbol{X}_{\text{z-score}} = \frac{\boldsymbol{X} - \bar{\boldsymbol{X}}}{\boldsymbol{\sigma_X}} \tag{6}$$

3.2.4 Redukcia rozmerov

Podľa [41] je možné vykonať redukciu rozmerov cez extrakciu alebo výber príznakov. V prípade extrakcie príznakov ide o lineárnu alebo nelineárnu transformáciu typu N-to-M, kde platí $\boldsymbol{N} \gg \boldsymbol{M}$. V prípade veľarozmerných priestorov typických pre oblasť detekcie sieťových prienikov sa osvedčila metóda analýzy hlavných komponentov (angl. *Principal Component Analysis*; PCA). Pri výbere príznakov len vyberáme M z N dostupných príznakov bez vykonania transformácie. Výber prebieha na základe informačnej hodnoty, ktorá je pridelená každému z príznakov v procese ohodnotenia ich dôležitosti, tzv. feature ranking.

3.2.5 Diskretizácia symbolických príznakov

Pre prevod symbolických príznakov na číselné sa používajú najčastejšie indikátorové premenné. Ide o množinu umelo vytvorených binárnych príznakov, z ktorých každému symbolu prislúcha práve jeden indikátor. Výhodou indikátorových premenných v porovnaní s priamym priradením čísla každému symbolu je zachovanie vzdialenosti medzi každou dvojicou symbolov. Táto vlastnosť je dôležitá najmä v prípade metód, ktoré vyjadrujú rozdiely medzi vstupnými vzorkami cez vzájomnú vzdialenosť k nim prislúchajúcich bodov v N-rozmernom priestore.

3.2.6 Náš pohľad na sieťovú prevádzku

Vzhľadom na prístupy publikované vo viacerých vedeckých článkoch a zhrnutých vyššie, zvolili sme ako základný objekt pozorovania tok. Pre výpočet štatistických charakteristík je použitá metóda posuvných okien. Pri tejto metóde je jednou z úloh voľba vhodnej šírky okna. Malé okno je okamžite dostupné, čo minimalizuje oneskorenie spôsobené odchytením okna a spracovaním jeho obsahu. Dlhé okno je schopné zachytiť aj dlhšie priebehy útokov. V prípade detekčnej metódy potrebujeme schopnosti oboch prípadov. Riešením je použitie rôzne dlhých časových okien súčasne.

Pre zachytenie a popis aktuálneho stavu sieťovej prevádzky sme zvolili nasledujúci 3D objekt s rozmermi $\boldsymbol{N} \times \boldsymbol{M} \times \boldsymbol{K}$, kde $\boldsymbol{N}$ je počet prvkov v jednom riadku, $\boldsymbol{M}$ je počet rôznych šírok okien, a $\boldsymbol{K}$ je počet rôznych tokov, ktoré sledujeme. X-os v takto definovanom objekte vyjadruje indexy po sebe idúcich časových okien s rovnakou šírkou, y-os vyjadruje dĺžku časového okna, z-os vyjadruje tok a samotná hodnota daného prvku vyjadruje niektorú zo štatistických charakteristík, napr. intenzita vyjadrená cez počet paketov alebo bajtov, priemerná veľkosť paketu, a pod. Inými slovami, každý rez takéhoto objektu predstavuje stav konkrétneho toku, ktorý je definovaný päticou kľúčových príznakov: zdrojová a cieľová IP adresa, zdrojový a cieľový port, a protokol. Každý riadok (daný indexom na y-ovej osi) reprezentuje postupnosť susedných neprekrývajúcich okien s pevne stanovenou šírkou danou vzťahom (7),

$$\boldsymbol{W_y} = \boldsymbol{W_0} \times (\boldsymbol{1} + \boldsymbol{\alpha} \times \boldsymbol{y}) \tag{7}$$

kde $\boldsymbol{W_0}$ je základná šírka okna a $\boldsymbol{\alpha}$ je základný prírastok. *Y-tý* riadok, braný ako postupnosť časových okien, zachytáva časový interval s dĺžkou danou vzťahom (8),

$$\boldsymbol{\Delta T\,(y) = N \times W_y} \tag{8}$$

kde $\boldsymbol{N}$ je pevne stanovený počet okien v riadku, a $\boldsymbol{W_y}$ je šírka jedného okna. Keďže dĺžka tohto časového intervalu závisí od indexu na y-ovej osi, rôzne riadky zachytávajú rôzne dlhé časové intervaly sieťovej prevádzky.

3.3 Architektúra modelu

Ďalším krokom je návrh architektúry modelu, ktorý na základe vstupných predspracovaných dát vykoná detekciu sieťových prienikov. Vzhľadom na súčasný trend nasadenia hlbokého učenia [42] sme zvolili model konvolučnej neurónovej siete (kap. 1.3).

Aktuálnou otázkou je možnosť použiť metódy hlbokého učenia v oblasti detekcie sieťových prienikov. Sieťová prevádzka vykazuje všeobecné vlastnosti hierarchickej štruktúry – sieťová prevádzka pozostáva z tokov, tok pozostáva z postupnosti paketov, paket pozostáva z hlavičiek, a každá hlavička pozostáva z polí. Sieťový prienik predstavuje v tomto kontexte špecifický tok alebo skupinu tokov, ktoré sa značne odlišujú od ostatných, normálnych tokov. Schopnosť modelov hlbokého učenia automaticky extrahovať skryté príznaky zjednodušuje výber a extrakciu vhodných príznakov zo sieťovej prevádzky pre detekciu sieťových prienikov. Navyše, hlboké učenie umožňuje objaviť a použiť aj príznaky, ktoré človek nie je schopný manuálne odhaliť. Preto predpokladáme, že hlboké učenie je vhodným prístupom pre detekciu sieťových prienikov. Túto myšlienku podporujú aj viaceré príklady detektorov anomálií v sieťovej prevádzke, ktorých úspešné nasadenie umožnili hlboké neurónové siete [41], [43], [44].

3.4 Optimalizácia modelu - trénovanie

Úlohou optimalizácie modelu je hľadanie takej kombinácie parametrov modelu, pri ktorých sa model správa podľa vopred definovaných požiadaviek. Z tohto pohľadu vnímame trénovanie ako proces riešenia optimalizačnej úlohy. Cieľom trénovania je minimalizácia chybovej funkcie (angl. *error function*) pre danú množinu vzoriek a prislúchajúce značky. Okrem chybovej funkcie je dôležitá voľba optimalizačnej metódy, ktorá špecifikuje spôsob, akým sú parametre modelu upravované.

3.4.1 Optimalizačné metódy

Príkladom optimalizačných metód, používaných v strojovom učení, sú gradientové metódy. Najpoužívanejšou z gradientových metód je metóda najstrmšieho

spádu. Konkrétne ide o stochastickú metódu najstrmšieho spádu (angl. *Stochastic Gradient Descent*; SGD), ktorá pracuje v rámci iterácie len s malou skupinou vzoriek (angl. *minibatch*). V prípade zložitejších hierarchických modelov, napr. viacvrstvová neurónová sieť, predstavuje výpočet gradientu na niektorej zo skrytých vrstiev problém. Riešením je metóda spätného šírenia chyby (angl. *back-propagation*), ktorá popisuje algoritmus na systematický výpočet a šírenie gradientov chybovej funkcie vrstvami modelu umelej neurónovej siete. V praxi sa okrem SGD používajú aj ďalšie optimalizačné metódy, napr. AdaGrad, RMSProp, a ADAM.

3.4.2 Typy trénovania

Hlavným faktorom pre výber konkrétneho typu učenia je dostupnosť, resp. nedostupnosť značiek pre vzorky v dátovej množine. Podľa dostupnosti trénovacej dátovej množiny a prislúchajúcich značiek sú k dispozícii štyri typy učenia [42]:

(a) **Riadené učenie** (angl. *supervised learning*)

(b) **Neriadené učenie** (angl. *unsupervised learning*)

(c) **Čiastočne riadené učenie** (angl. *semi-supervised learning*)

(d) **Učenie odmeňovaním** (angl. *reinforcement learning*)

3.5 Implementácia modelu

3.5.1 Výber platformy technických prostriedkov

V súčasnosti sú dostupné viaceré technické prostriedky, ktoré je možné použiť pre implementáciu konvolučnej siete. Patria sem: *integrované obvody navrhnuté pre špecifickú aplikáciu (ASIC)*, *neurónové čipy*, *programovateľné hradlové polia (FPGA)*, *grafické procesorové jednotky (GPGPU)*, a rôzne *sekvenčné procesory*.

Pre návrh a implementáciu inferencie konvolučnej siete bola v práci zvolená platforma **FPGA**. Inferencia konvolučnej siete je implementovaná v mnohých zariadeniach, prebieha pre každú vstupnú vzorku a na každom nasadenom zariadení opakovane, a tak je jej optimalizácia smerom k efektivite podstatná. Vzhľadom na rozvoj oblasti Internetu vecí sú aktuálne primárnou cieľovou skupinou energeticky obmedzené zariadenia, a preto je všeobecným cieľom optimalizácie inferencie nájsť vhodný pomer medzi **efektivitou** (počet operácií v pomere ku spotrebe), **výkonom** (oneskorenie a počet operácii za čas), a **presnosťou** výstupov [45].

FPGA prirodzene podporuje návrh vysoko paralelných architektúr výpočtových systémov vo forme špeciálnych aritmetických subsystémov (DSP) a pamäťových blokov (BRAM) umiestnených priamo v štruktúre obvodu FPGA. Vzájomná nezávislosť prístupu k BRAM a DSP blokom v FPGA podporuje paralelný prístup spracovania dát konvolučnou sieťou, a to na viacerých úrovniach:

po vrstvách, po príznakových mapách, po výstupných príznakoch v rámci príznakovej mapy, a po numerických operáciach v rámci výpočtu jedného príznaku [46].

Naším cieľom je použiť platformu s vhodným pomerom medzi flexibilitou a výkonom z pohľadu inferencie konvolučnej siete. Vzhľadom na výhody FPGA, uvedené v [45]–[48] považujeme túto platformu za vhodnú pre riešenie zadanej úlohy. V návrhu systému konvolučnej siete, popísaného v tejto práci, boli použité techniky **prúdového spracovania** [49] a **systolických polí** [50], [51].

4 Architektúra subsystémov konvolučnej siete

Vzhľadom na modulárnu štruktúru konvolučnej siete, popísanej v kap. 1.3, a náš cieľ aplikovať FPGA obvod pozostáva navrhnutá architektúra z viacerých funkčne odlišných blokov, odpovedajúcich rôznym typom vrstiev.

4.1 Subsystém konvolučnej vrstvy

Vzhľadom na popis 2D konvolúcie (kap. 1.3.1) a prislúchajúci matematický vzťah (1), je možné rozdeliť jej výpočet do viacerých krokov. 2D konvolúcia je rozložiteľná na skupinu 1D konvolúcii, ktoré používajú len vektorové operácie a predstavujú všeobecný problém číslicového spracovania signálov.

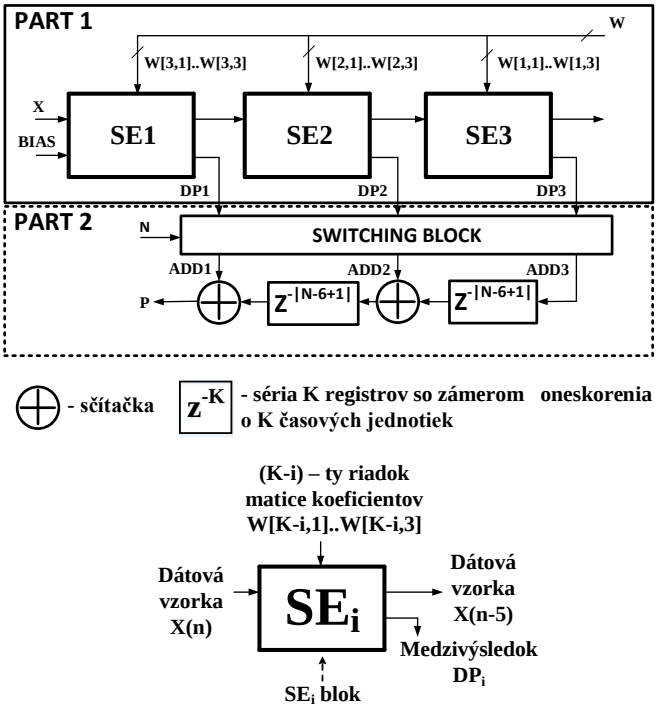
Predpokladáme, že vstupná príznaková mapa má pravouhlý tvar s rovnakou konštantnou výškou a šírkou. V tejto práci je veľkosť vstupnej mapy vyjadrená parametrom N. Rovnaký predpoklad o konštantnej vopred definovanej šírke platí aj pre maticu koeficientov (angl. *kernel*). Šírka matice koeficientov je označená parametrom K. Všetky v práci uvedené príklady predpokladajú šírku matice koeficientov $\boldsymbol{K = 3}$. Maticu s touto šírkou považujeme za **základný formát**, ktorý poskytuje jednoduchosť a jasnú zrozumiteľnosť, pričom zachováva hlavné princípy modelu.

4.1.1 Architektúra modelu

Navrhovaný model pozostáva z dátovej a riadiacej štruktúry.

Popis dátovej štruktúry Dátová štruktúra pozostáva z dvoch častí, ktoré v práci označujeme **PART1** a **PART2**. Ich adaptovaný model je zobrazený na obr. 3.

Úlohou prvej časti (*PART1*) je výpočet jednotlivých vnútorných súčtov, ktorý pozostáva z opakovaného násobenia dvojíc – vstupná vzorka, koeficient – a následného sčítania súčinov (angl. *Multiply-And-Accumulate*; MAC). Výstupmi z tejto časti modelu sú čiastočné skalárne súčiny $\mathrm{DP}_{\boldsymbol{i}}$ (9) a (10),



Obr. 3: Adaptovaný model dátovej štruktúry 2D konvolútora so zámerom následnej efektívnej implementácie do FPGA obvodu (Zdroj: [51]). Adaptovaný model (pre $\boldsymbol{K = 3}$) pozostáva z troch SE blokov – **SE1**, **SE2**, a **SE3**, prepínacieho bloku, sčítačiek a vyrovnávacích pamätí vo forme registrov. Vstupom do modelu je tok dát **X**, reprezentujúci body vstupnej mapy, a **bias**. Vstupné dáta prechádzajú sekvenčne cez SE bloky. Okrem vstupných dát používa každý SE blok trojicu váhových koeficientov pre výpočet prislúchajúceho medzivýsledku, t. j. blok **SEi** pracuje s váhovými koeficientami **W[3-i+1,1]**, **W[3-i+1,2]** a **W[3-i+1,3]**. Výstupom z bloku **SEi** je medzivýsledok **DPi**. Prepínací blok pripojí vhodným spôsobom výstupy z SE blokov na sčítačky a vyrovnávacie pamäte, pričom berie do úvahy aktuálny rozmer vstupnej mapy N. Výstupom z modelu 2D konvolútora je finálny súčet medzivýsledkov **P**.

$$\mathrm{DP}_{\mathbf{1}} = \mathrm{BIAS} + \sum_{j=1}^{\boldsymbol{K}} \boldsymbol{X}\,(\boldsymbol{n} - \boldsymbol{j} + \mathbf{1}) \times \boldsymbol{W}\,[\boldsymbol{K}, \boldsymbol{j}] \tag{9}$$

$$\mathrm{DP}_{i=\mathbf{2}\ldots\boldsymbol{K}} = \sum_{j=\mathbf{1}}^{\boldsymbol{K}} \boldsymbol{X}\,(\boldsymbol{n}-\boldsymbol{j}+\mathbf{1}) \times \boldsymbol{W}\,[\boldsymbol{K}-\boldsymbol{i}+\mathbf{1}, \boldsymbol{j}] \tag{10}$$

ktoré musia byť vhodne spojené do finálneho výsledku ($\boldsymbol{P}$) (11).

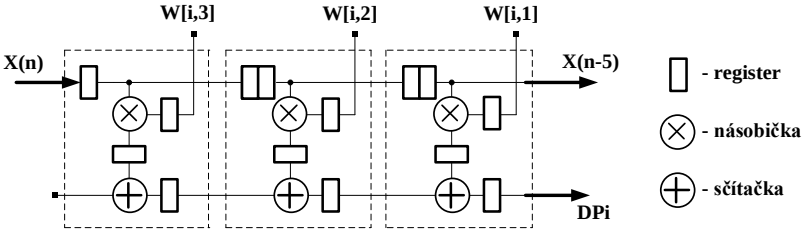
$$\boldsymbol{P} = \sum_{i=\mathbf{1}}^{\boldsymbol{K}} \mathrm{DP}_{\boldsymbol{i}} \tag{11}$$

Druhá časť (*PART2*) určuje časovanie súvisiace so spájaním jednotlivých medzivýsledkov, operáciou sčítania, keďže rôzne medzivýsledky pre výpočet jedného konkrétneho výstupu sú platné v rôznych časových intervaloch.

Prvá časť pozostáva zo systolických prvkov, usporiadaných do tvaru reťaze. Každý systolický prvok (SE_i) tvorí 1D konvolútor s K koeficientami. Pri jeho implementácii vychádzame z odporúčaných metód [52], [53]. DSP bloky umožňujú optimálnu implementáciu konvolútora. Vstavaná násobička a sčítačka podporujú symetrické zaokrúhľovanie a kvantizáciu výsledkov numerických operácií. Ich použitím je možné predísť pretečeniu a udržať tak validnosť výstupov. Vďaka týmto funkciám je možné vytvoriť model kaskádovo zapojených DSP blokov so zámerom efektívne počítať súčiny a postupne ich spájať do finálneho výstupu cez reťaz sčítačiek bez použitia ďalších zdrojov FPGA.

Po zvážení nárokov vysokej vzorkovacej frekvencie a malého počtu koeficientov sme zvolili konkrétny typ paralelnej implementácie konvolútora – **číslicový systolický výpočtový systém 2D konvolúcie**. Dôvodom sú výhody, ktoré tento model vykazuje: **vysoký výkon**, **efektívne mapovanie** operácií na DSP bloky, a **bez požiadaviek na externé zdroje** (zdroje nezahrnuté v DSP blokoch) [52]. Systolický konvolútor je všeobecne považovaný za optimálny model s paralelným spracovaním pre obvody FPGA. Kaskádový model zásadne znižuje spotrebu energie a zvyšuje rýchlosť spracovania tým, že odzrkadľuje pravidelné usporiadanie a priame prepojenia medzi DSP, BRAM, a CLB. Ilustračný model systolického konvolútora s tromi koeficientami je zobrazený na obr. 4.

Druhá časť dátovej štruktúry preusporiadava medzivýsledky s cieľom ich následného spojenia do požadovaného výstupu. Preusporiadanie je realizované oneskorením rôznych medzivýsledkov takým spôsobom, aby boli vždy skombinované (sčítané) len medzivýsledky prislúchajúce k spoločnému finálnemu výsledku. K tomuto účelu slúžia oneskorovacie vyrovnávacie pamäte (angl. *delay buffer*). Dĺžka oneskorenia je zvolená so zámerom zachovať platnosť výstupov a synchronizovaný stav modelu. Vzdialenosť medzi susednými medzivýsledkami toho istého výstupu je rovná N časových jednotiek. Navyše, prvá časť dátovej štruktúry už prináša oneskorenie $\mathbf{2} \times \boldsymbol{K} - \mathbf{1}$ časových jednotiek. Odčítaním oneskorenia z prvej časti od celkového požadovaného oneskorenia sú tak oneskorovacie pamäte zodpovedné za zostávajúce oneskorenie dlhé $\boldsymbol{N} - \mathbf{2} \times \boldsymbol{K} + \mathbf{1}$ časových jednotiek.



Obr. 4: Model SE$_i$ bloku (Zdroj: [51]). $\boldsymbol{SE_i}$ pozostáva z registrov, násobičiek a sčítačiek. Registre slúžia ako dočasná pamäť pre vstupné dáta v rôznych časových okamihoch – $\boldsymbol{X_n, X_{n-1}, \ldots, X_{n-5}}$, aj pre dáta z operácií násobenia a sčítania. Druhým operandom násobenia je niektorý z váhových koeficientov $\boldsymbol{W\,[i, 3]}$, $\boldsymbol{W\,[i, 2]}$ a $\boldsymbol{W\,[i, 1]}$. Oneskorený vstupný údaj $\boldsymbol{X_{n-5}}$ vychádza z bloku a je priamo použitý ako vstup do susedného SE$_{i+1}$ bloku. Ďalším výstupom z bloku je medzivýsledok **DPi**.

Popis riadiacej štruktúry Riadiaca štruktúra je realizovaná konečným stavovým automatom (angl. *Finite State Machine*; FSM). Rozhranie FSM pozostáva z jedného vstupu a jedného výstupu, ktoré reprezentujú valídnosť aktuálneho vstupného a výstupného príznaku. Náš návrh FSM využíva dostupné informácie o pevnej, vopred známej šírke a výške výstupnej príznakovej mapy pre pravidelné nastavovanie a sledovanie interných binárnych počítadiel (angl. *counter*). FSM nepretržite počíta príchod platných vstupných vzoriek, generuje signál indikujúci platnosť výstupov a prechádza medzi stavmi podľa aktuálneho stavu počítadiel. Stavy FSM reprezentujú rôzne prípady umiestnenia aktuálne spracovaného okna vo vstupnej mape.

FSM rozlišuje tri prípady prechodu okna vstupnou mapou:

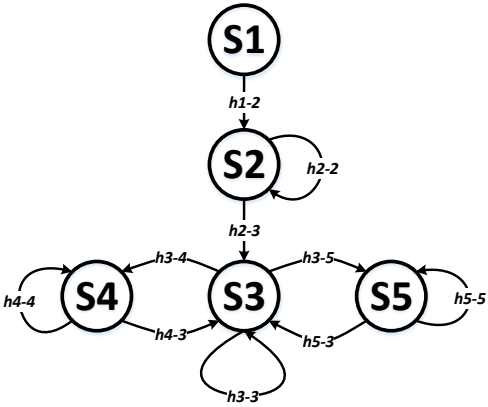
(a) prechod **vo vnútri** vstupnej mapy,

(b) prechod **cez vertikálne okraje** – prechod na nový riadok v rámci tej istej vstupnej mapy,

(c) prechod **cez horizontálne okraje** – prechod na nasledujúcu vstupnú mapu.

Výsledky získané počas (b) a (c) sú považované za neplatné a musia byť ignorované v ďalšom spracovaní. Stavový diagram FSM je uvedený na obr. 5. Diagram pozostáva z piatich stavov, ktoré mapujú okrem troch vyššie uvedených prípadov ešte fázy inicializácie (init) a rozbehnutia (startup). Všeobecne platí, že všetky výstupy získané kombináciou neplatných medzivýsledkov sú tiež považované za neplatné. Z dôvodu konkrétnych, konštantných hodnôt parametrov N a K je možné priamo vyjadriť rozmery platných a neplatných oblastí

na výstupe. Numerické vzťahy pre tieto rozmery sú použité v FSM riadiacej štruktúry pre identifikáciu platnosti výstupov.



Obr. 5: Stavový diagram FSM 2D konvolútora. Riadiaca časť (FSM) pozostáva z piatich stavov *S1*, . . . , *S5* (tab. 1). Hrany medzi stavmi reprezentujú podmienené prechody *h1-2*, . . . , *h5-5*, kde prvá cifra označuje číslo zdrojového stavu a druhá cifra označuje číslo cieľového stavu.

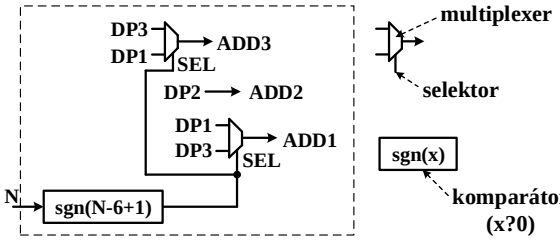
Tabuľka 1: Prehľad stavov v automate riadiacej časti FSM 2D konvolútora (obr. 5).

Stav	Názov
S1	Inicializačný stav (INIT)
S2	Začiatok výpočtov (STARTUP)
S3	Vo vnútri mapy (INSIDE_IMAGE)
S4	Vertikálna hrana (VERTICAL_BORDER)
S5	Horizontálna hrana (HORIZONTAL_BORDER)

Prepínací blok V prípade záporného oneskorenia (platí $\boldsymbol{N < 2 \times K - 1}$) sú medzivýsledky, odchádzajúce z SE blokov, presmerované do jednotlivých sčítačiek tak, že medzivýsledok z prvého SE bloku ($\mathrm{DP}_{\mathbf{1}}$) je pripojený na poslednú sčítačku ($\mathrm{ADD}_{\boldsymbol{K}}$) a medzivýsledok z posledného SE bloku ($\mathrm{DP}_{\boldsymbol{K}}$) je pripojený na prvú sčítačku ($\mathrm{ADD}_{\mathbf{1}}$). Príklad prepínacieho bloku pre $\boldsymbol{K = 3}$ je uvedený na obr. 6.

4.1.2 Originálny prínos nášho návrhu

Nami navrhovaný model 2D konvolútora predstavuje originálnu štruktúru, pričom originalita návrhu spočíva v unikátnom umiestnení oneskorovacích pamätí.



Obr. 6: Schéma prepínacieho bloku pre $\boldsymbol{K = 3}$. Vstupmi do bloku sú šírka vstupnej mapy N, a medzivýsledky **DP1**, **DP2** a **DP3**, získané z SE blokov. Blok pozostáva zo skupiny multiplexerov so spoločným selektorom **SEL**. Hodnota selektora je daná znamienkom výrazu $\boldsymbol{N - 6 + 1 - sgn\,(N - 6 + 1)}$. Každý medzivýsledok je pripojený do dvoch multiplexerov. Výstupmi z bloku sú výstupy z multiplexerov **ADD1** a **ADD3**. Výnimkou z dôvodu symetrie je medzivýsledok z prostredného SE bloku **DP2**, ktorý je priamo napojený na výstup **ADD2**. Tieto hodnoty sú následne použité ako vstupy do sčítačiek v druhej časti 2D konvolútora.

Štandardným umiestnením oneskorovacích pamätí je ich vloženie pred výpočtovú časť alebo častejšie používané vloženie pamätí do vnútra výpočtovej časti medzi jednotlivé výpočtové bloky ([54], kap. 8). Príklady modelov sú uvedené v článkoch [27], [55]. Efektívne riešenie predstavuje vloženie oneskorovacích pamätí až za výpočtové bloky. Pamäte sú prepojené so sčítačkami spôsobom, ktorý nevyžaduje aplikáciu súčtového stromu. Použitím tejto konštrukcie je dosiahnuté zníženie celkového oneskorenia výstupov a možnosť efektívne využiť štruktúru FPGA obvodu. Konkrétne ide o interné rýchle prepojenia medzi susednými DSP blokmi. Tieto prepojenia môžu slúžiť na šírenie vstupných pixelov reťazou násobičiek, ktoré sú súčasťou DSP blokov.

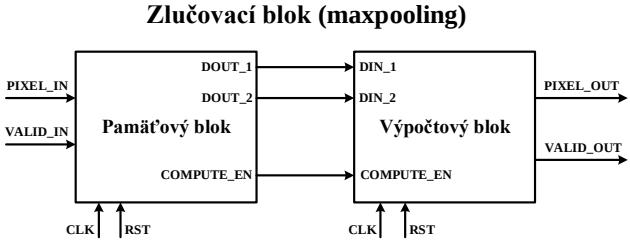
4.2 Subsystém zlučovacej vrstvy

V práci je použitý vlastný návrh s najčastejšie používanými hodnotami parametrov: operácia **maximum**, rozmery okna $\boldsymbol{2 \times 2}$ pixelov a dĺžka kroku **2** pixely. Nami navrhnutý subsystém zlučovacej vrstvy pozostáva z dvoch blokov: **pamäťový** blok a **výpočtový** blok, zobrazené na obr. 7.

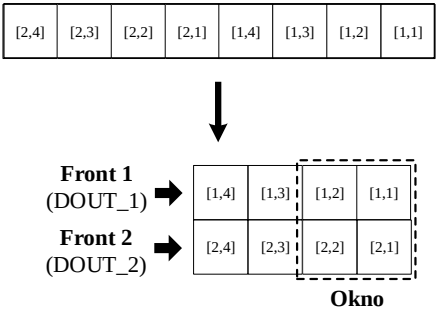
4.2.1 Pamäťový blok

Úlohou pamäťového bloku je preusporiadanie vstupných pixelov v čase. Obr. 8 ilustruje účinok použitia frontov na poradie pixelov.

Štruktúra pamäťového bloku, zobrazená na obr. 9, pozostáva z dvoch samostatných dvojíc frontov. Každý front má pevne zvolenú dĺžku, danú šírkou



Obr. 7: Schéma zlučovacieho bloku. Zlučovací blok pozostáva z pamäťového a výpočtového bloku. Vstupné signály *PIXEL_IN* a *VALID_IN* reprezentujú hodnotu a platnosť prichádzajúcich pixelov vstupnej mapy. Pamäťový blok preusporiada pixely do dvoch tokov, ktoré posiela do výpočtového bloku cez porty *DOUT_1* a *DOUT_2*. Výpočtový blok vvpočíta maximum zo štvorice susedných pixelov. Výslednú hodnotu posiela na výstupný port *PIXEL_OUT*, pričom jej platnosť indikuje cez výstupný signál *VALID_OUT*.

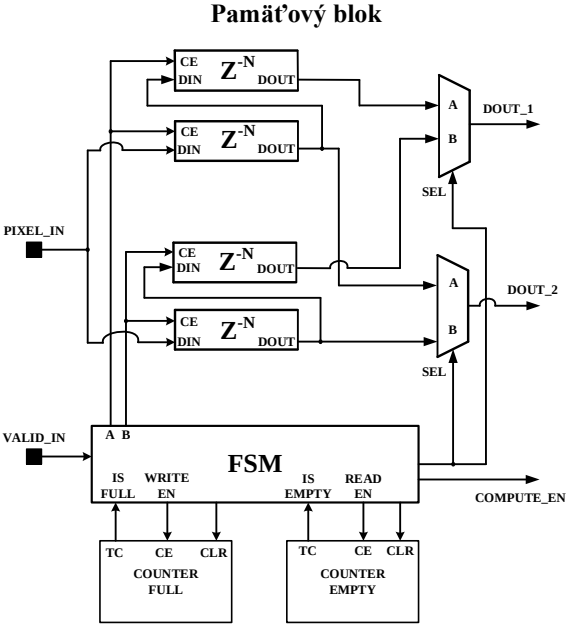


Obr. 8: Preusporiadanie pixelov cez pamäťový blok. Pixely vstupnej mapy na vstupnom rozhraní do pamäťového bloku tvoria jednorozmerný vektor, prúd dát. Použitím frontov dochádza k ich preusporiadaniu tak, aby bol následný výpočet vykonaný nad požadovanou dvojicou pixelov v spoločnom okne.

vstupnej mapy **N**. Zvolená šírka je stanovená tak, aby po naplnení každý z dvojice pasívnych frontov obsahoval pixely jedného riadku vstupnej mapy. Pasívna dvojica frontov prijíma platné vstupné pixely. Aktívna dvojica frontov vyberá pixely a posúva ich ďalej do výpočtového bloku alebo je neaktívna (čaká). Fronty si vzájomne vymieňajú úlohu aktívneho a pasívneho stavu. Ide o mechanizmus frontov, známy ako *Ping-Pong*, použitý napríklad v [56].

Vkladanie vstupných pixelov do pasívnych frontov, výber pixelov z aktív-

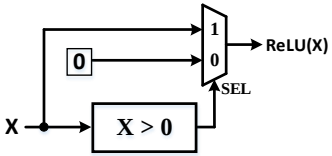
nych frontov a striedanie roly frontov vyžaduje riadenie. Pre tento účel je navrhnutý riadiaci automat pamäťového bloku, ktorý zabezpečuje generovanie korektných riadiacich signálov. Automat využíva vopred známe informácie o rozmeroch vstupnej mapy **N** a veľkosti okna **K**. Stavový automat mení stavy frontov podľa hodnôt interných počítadiel.



Obr. 9: Štruktúra pamäťového bloku. Pamäťový blok pozostáva z dvoch párov posuvných registrov s dĺžkou *N*, ktoré plnia úlohy frontov. Vkladanie pixelov zo vstupného portu *PIXEL_IN* a ich posúvanie v každom z dvojice frontov je riadené cez samostatný povoľovací signál *CE*. Súčasťou pamäťového bloku je riadiaci automat FSM, ktorého úlohou je regulácia posúvania pixelov vo frontoch na základe ich stavu (*pasívny* / *aktívny*) a hodnôt indikátorov z počítadiel *COUNTER_FULL* a *COUNTER_EMPTY*. Preusporiadané pixely vychádzajú z frontov cez výstupné porty *DOUT_1* a *DOUT_2*. Ich platnosť je daná výstupným signálom *COMPUTE_EN*. Systémové signály *CLK* a *RST* nie sú kvôli prehľadnosti v štruktúre zobrazené.

4.2.2 Výpočtový blok

Výpočtový blok (obr. 11) obsahuje dva kaskádovo zapojené operátory *maximum*. Medzi operátory sú vložené dva registre (*RA* a *RB*), ktoré slúžia na dočasné ulo-



Obr. 10: Model rektifikovanej lineárnej jednotky (ReLU). Model správaním zodpovedá aktivačnej funkcii ReLU (4), pozostáva z komparátora a multiplexera. Komparátor porovnáva vstupný pixel X s nulou. Multiplexer následne posiela na výstup hodnotu vstupného pixelu alebo nulu podľa riadiaceho signálu *SEL* z komparátora.

ženie dvojice po sebe idúcich operandov. Z registrov je vždy aktívny práve jeden, registre sa pravidelne striedajú. Výstupná hodnota je platná až po prijatí a spracovaní štyroch platných pixelov, ktoré tvoria okno $\mathbf{2 \times 2}$ pixelov (zvýraznené okno na obr. 8).

4.3 Subsystém plne-prepojenej vrstvy

V prípade plne-prepojenej vrstvy sme zvolili priamočiaru architektúru založenú na súčtovom strome (angl. *adder tree*).

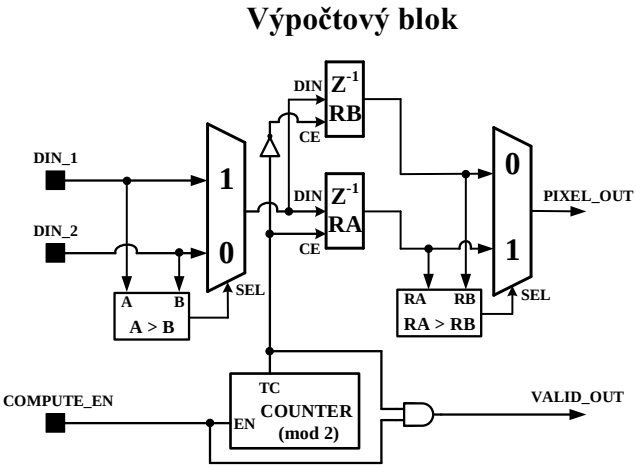
4.3.1 Súčtový strom

Súčtový strom vykonáva súčet N vstupných dát, pričom využíva štruktúru binárneho stromu. Uzly v strome sú operátory sčítania. Spracovanie vstupných hodnôt do súčtového stromu prebieha smerom od listov ku koreňu. Hrany, vstupujúce do listov, predstavujú hodnoty vstupných dát; hrana, vystupujúca z koreňa vyjadruje hodnotu súčtu všetkých vstupných dát, prijatých v rovnakom časovom okamihu.

Architektúra subsystému plne-prepojenej vrstvy rozširuje súčtový strom o ďalšiu úroveň uzlov, ktorá slúži na výpočet súčinov vstupných pixelov s prislúchajúcimi váhami. Schéma rozšíreného súčtového stromu je zobrazená na obr. 12.

4.4 Subsystém aktivačnej vrstvy

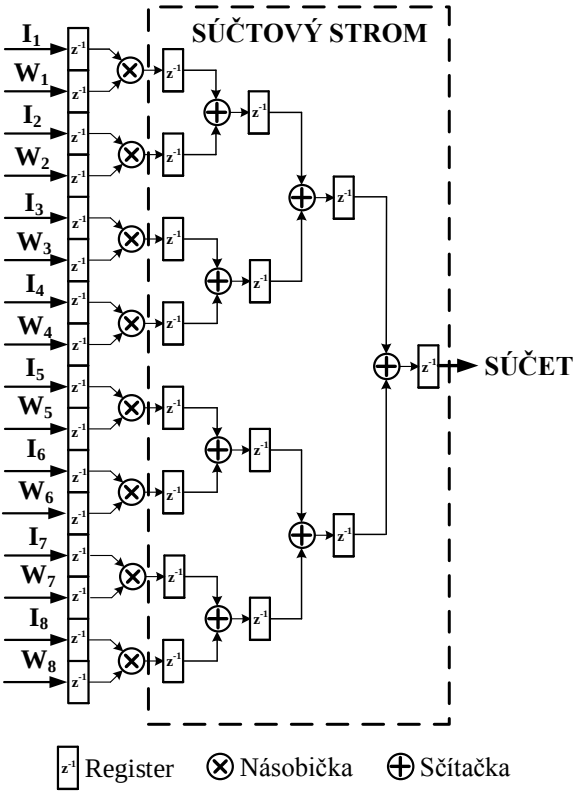
V subsystéme aktivačnej vrstvy sa výpočet vzťahuje na každý pixel vstupnej mapy samostatne, čím sa odlišuje od predchádzajúcich vrstiev. Návrh subsystému aktivačnej vrstvy tak spočíva len v návrhu subsystému aktivačnej funkcie. V práci je uvedený model aktivačnej funkcie $ReLU$ (4), zobrazený na obr. 10.



Obr. 11: Štruktúra výpočtového bloku. Dvojica pixelov susedných riadkov vstupuje do výpočtového bloku cez porty *DIN_1* a *DIN_2*. Vstupný signál *COMPUTE_EN* udáva ich platnosť. Následný výpočet bloku spočíva v aplikácii dvoch kaskádovo zapojených operácii *maximum*, realizovaných cez komparátor a multiplexer. Výstup z komparátora slúži ako selektor pre multiplexer z dôvodu výberu väčšej hodnoty z dvojice pixelov. Výstup z prvého multiplexeru je uložený do jedného z registrov *RA* a *RB*. Registre sa pravidelne striedajú, aby vždy obsahovali väčšie hodnoty z dvoch po sebe idúcich dvojíc vstupných pixelov. Striedanie zabezpečuje počítadlo *COUNTER (mod 2)* nastavením povoľovacieho signálu *CE*. Hodnoty z registrov sú použité ako operandy druhého operátora *maximum*. Signál *PIXEL_OUT*, vychádzajúci z druhého multiplexera, predstavuje maximum zo štvorice vstupných pixelov, tvoriacich spoločné okno. Jeho platnosť je daná hodnotou výstupného signálu *VALID_OUT*. Systémové signály *CLK* a *RST* nie sú kvôli prehľadnosti v štruktúre zobrazené.

5 Experimentálne overenie navrhnutej architektúry

Vzhľadom na cieľovú implementačnú platformu FPGA boli pre popis nami navrhnutého systému zvolené dva prístupy: **modelovo-riadený prístup** [57] v nástroji *Matlab/Simulink 2018a* a **popis systému s použitím popisného jazyka HDL** (angl. *Hardware Description Language*), konkr. *Very High Speed Integrated Circuit HDL* (VHDL) v nástroji *Vivado Design Suite 2017.4*. Zámerom prvého prístupu je **overenie funkčného správania** systému formou simulácie. Zámerom druhého prístupu je **overenie možnej implementácie navrhnutého**



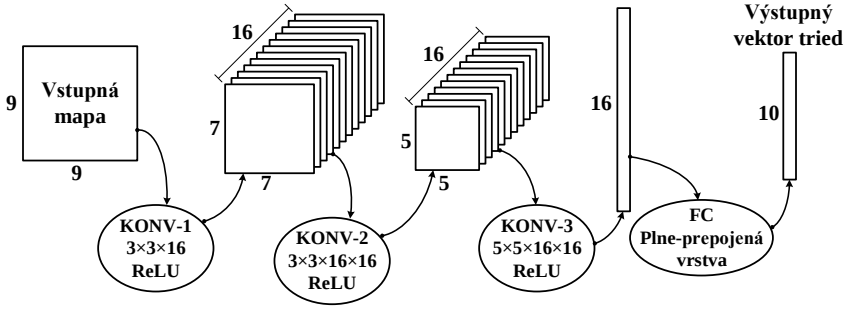
Obr. 12: Rozšírený súčtový strom v architektúre plne-prepojenej vrstvy. $\boldsymbol{I_1, \ldots, I_8}$ sú vstupné pixely a $\boldsymbol{W_1, \ldots, W_8}$ sú prislúchajúce váhy. Výstupom z bloku je *SÚČET*, daný vzťahom: $\sum_{i=1}^{8} \boldsymbol{I_i} \times \boldsymbol{W_i}$. Susedné úrovne stromu sú prepojené cez registre z dôvodu obmedzenia dĺžky kritickej cesty. Vložené registre tak umožňujú zvýšiť taktovaciu frekvenciu.

systému do FPGA obvodu, **analýza syntézy systému** (prevod HDL kódu do zdrojov FPGA) a **výsledné požiadavky systému na typ a množstvo zdrojov FPGA obvodu**.

5.1 Referenčná konvolučná sieť

Pre overenie korektnosti správania nami navrhnutých subsystémov v kap. 4 bola použitá konkrétna konvolučná sieť s architektúrou podľa obr. 13. Vstupné a výstupné hodnoty siete, v práci označované ako *referenčné hodnoty* použité počas

simulácie, pochádzajú z jej implementácie do platformy GPU. Popis implementácie siete do GPU ani overenie jej správnosti nie sú súčasťou práce.



Obr. 13: Architektúra referenčnej siete. Zvolená konvolučná sieť pozostáva zo štyroch vrstiev (okrem vstupnej vrstvy): *KONV-1*, *KONV-2*, *KONV-3* a *FC*. Čísla pri vstupných a výstupných mapách uvádzajú ich rozmery. Počet a rozmery váhových matíc sú uvedené pod označeniami jednotlivých vrstiev.

5.1.1 Testovací dataset

Upravená testovacia sada obrázkov z datasetu MNIST [28] bola použitá ako testovacia sada pre overenie správania navrhnutého modelu. Zvolená testovacia sada (obr. 14) pozostáva zo sto obrázkov. Rozlíšenie obrázkov bolo zmenšené z pôvodných $\mathbf{28 \times 28}$ pixelov na $\mathbf{9 \times 9}$ pixelov. Vstupné pixely sú kódované ako 9-bitové čísla z intervalu $\mathbf{(0, 1)}$. Váhy vo váhových maticiach všetkých vrstiev sú vyjadrené ako 5-bitové čísla z intervalu $\mathbf{(-1, 1)}$.

Kódovanie číselných hodnôt V práci vytvorených modeloch boli pre hodnoty vstupných máp do jednotlivých vrstiev siete použité rôzne parametre formátu s pevnou rádovou čiarkou (angl. *fixed-point*), uvedené v tab. 2. Voľba šírky celej a desatinnej časti hodnôt bola zvolená tak, aby nedošlo k pretečeniu.

Tabuľka 2: Parametre formátu s pevnou rádovou čiarkou pre hodnoty vstupných máp jednotlivých vrstiev siete. Hodnoty vyjadrujú počet vyhradených bitov pre danú časť.

	KONV-1	KONV-2	KONV-3	FC
Celá časť	1	3	4	6
Desatinná časť	8	7	6	4

Obr. 14: Upravená testovacia sada obrázkov z datasetu MNIST. Obrázky majú rozmery **9** × **9** pixelov. Nad každým obrázkom je uvedená dvojica čísel. Prvé číslo (umiestnené vľavo) reprezentuje triedu, predikovanú referenčným modelom. Druhé číslo (umiestnené vpravo a podčiarknuté) reprezentuje triedu, priradenú k obrázku počas **simulácie** modelu. Na základe porovnania týchto tried pre všetky zobrazené vstupné obrázky sa simuláciou modelu v nástroji Matlab/Simulink získané výstupné triedy vo všetkých prípadoch zhodujú s referenčnými triedami. Teda správanie modelu sa v simulácií úplne zhoduje so správaním referenčnej siete.

5.2 Verifikácia implementovaného systému konvolučnej siete

5.2.1 Modelovo-riadený popis testovaného modelu

Model, vytvorený v nástroji Matlab/Simulink, zodpovedá štruktúre referenčnej siete a zhoduje sa so schémou subsystémov, popísaných v kap. 4. Tento model slúži na **funkčnú** verifikáciu a validáciu navrhnutých subsystémov. Pre realizáciu modelu boli použité funkčné bloky z toolboxu *Xilinx Blockset*, dostupného

cez nástroj *System Generator* (rozšírenie nástroja *Vivado Design Suite - System Edition*) [58] (kap. 8). Pre funkčné overenie modelu sme zvolili viacero parametrov:

- *finálna klasifikácia vstupných obrázkov*, t.j. porovnanie tried, pridelených referenčným modelom, s triedami, ktoré vzorkám priradil nami navrhnutý model počas simulácie (obr. 14),
- *stredná kvadratická chyba* (angl. *Root Mean Square Error*; RMSE) modelu (obr. 15) pre každú z vrstiev siete,
- *absolútna hodnota rozdielov* medzi simuláciou získanými hodnotami a referenčnými hodnotami (obr. 16) pre každú z vrstiev siete.

Pre vyčíslenie priemernej chyby na jednotlivých vrstvách siete je na obr. 15 pre každú z vrstiev zachytená stredná kvadratická chyba (RMSE) s použitím vzťahu (12),

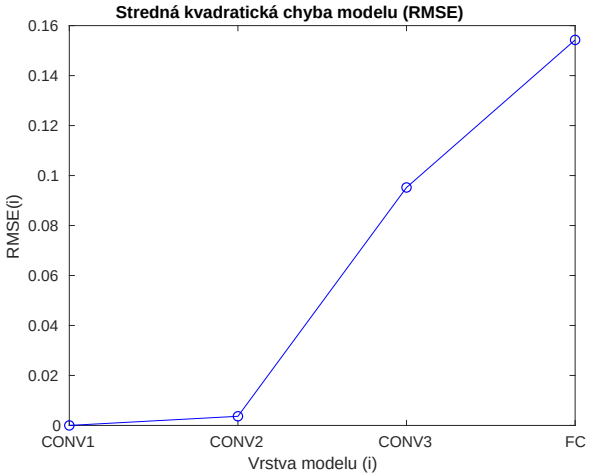
$$\mathrm{RMSE}\,(\boldsymbol{i}) = \sqrt{\frac{\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} \left(\boldsymbol{Y}_{\boldsymbol{k,sim}}^{\boldsymbol{i}} - \boldsymbol{Y}_{\boldsymbol{k,ref}}^{\boldsymbol{i}}\right)^{\boldsymbol{2}}}{\boldsymbol{N}}} \tag{12}$$

kde $\boldsymbol{i}$ je index vrstvy, $\boldsymbol{N}$ je počet všetkých výstupných príznakov na $\boldsymbol{i}$-tej vrstve, prislúchajúcich ku všetkým testovacím obrázkom, $\boldsymbol{Y}_{\boldsymbol{k,sim}}^{\boldsymbol{i}}$ je hodnota $\boldsymbol{k}$-teho výstupného príznaku $\boldsymbol{i}$-tej vrstvy získaná simuláciou nami navrhnutého modelu a $\boldsymbol{Y}_{\boldsymbol{k,ref}}^{\boldsymbol{i}}$ je hodnota prislúchajúceho $\boldsymbol{k}$-teho výstupného príznaku z referenčného modelu. Pre detailnejšie porovnanie presnosti výpočtov modelu je na obr. 16 uvedený okienkový diagram (angl. *boxplot*), ktorý zachytáva rozdiely medzi nami navrhnutým modelom získanými hodnotami výstupných máp a hodnotami z referenčného modelu samostatne pre každú vrstvu. Pre vyjadrenie rozdielu bol použitý vzťah (13).

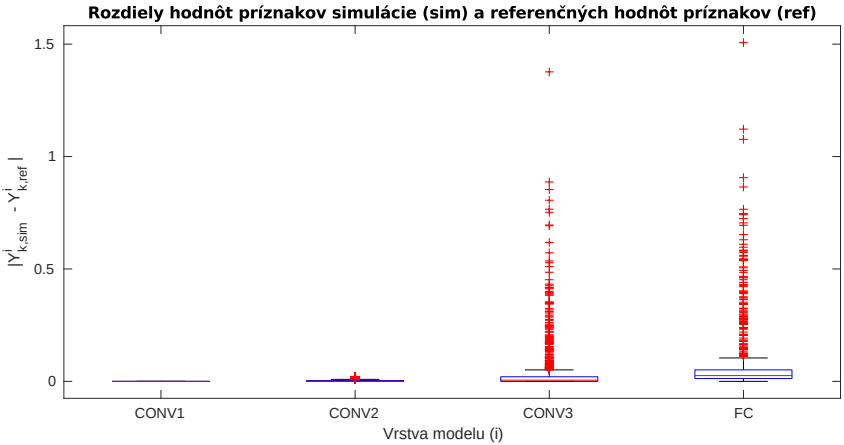
$$\boldsymbol{\Delta}_{\boldsymbol{i,k}} = \left|\boldsymbol{Y}_{\boldsymbol{k,sim}}^{\boldsymbol{i}} - \boldsymbol{Y}_{\boldsymbol{k,ref}}^{\boldsymbol{i}}\right| \tag{13}$$

V prípade prvej konvolučnej vrstvy sú rozdiely minimálne. Prechodom cez ďalšie vrstvy siete sa rozdiely mierne zväčšujú, pričom sa objavujú osamotené prípady s väčšou hodnotou chyby. Nárast chyby je spôsobený zmenšovaním počtu bitov určených pre vyjadrenie desatinnej časti hodnôt. Ďalším dôvodom sú nepresnosti vo výpočtoch jednej vrstvy, ktoré vedú k narastajúcej nepresnosti aj vo výpočtoch ďalších vrstiev.

Ďalším parametrom modelu je **oneskorenie** (angl. *delay*), ktoré je v práci vyjadrené ako dĺžka časového intervalu medzi prijatím prvého pixelu vstupného obrázku a odovzdaním modelom pridelenej výstupnej triedy. Vzhľadom na diskrétnu povahu nástroja Matlab/Simulink, dĺžka časového intervalu je vyjadrená ako počet periód synchronizačného *hodinového* signálu. Celkové oneskorenie modelu je dané ako súčet oneskorení, spôsobených prechodom vstupného signálu postupne cez jednotlivé subsystémy v poradí podľa obr. 13. Oneskorenia subsystémov sú zhrnuté v tab. 3, kde čas príchodu prvého platného údaju do subsystému $\boldsymbol{i}$-tej vrstvy $\boldsymbol{T}_{\boldsymbol{1}}^{\boldsymbol{i}}$ je vyjadrený ako počet periód hodinového signálu od



Obr. 15: Stredná kvadratická chyba modelu. X-os vyjadruje jednotlivé vrstvy referenčnej siete. Y-os vyjadruje *RMSE(i)* danú vzťahom (12).



Obr. 16: Rozdiely hodnôt príznakov simulácie modelu a referenčných hodnôt príznakov. X-os vyjadruje jednotlivé vrstvy referenčnej siete. Y-os vyjadruje $\boldsymbol{\Delta_{i,k}}$.

začiatku simulácie. Oneskorenie subsystému $\boldsymbol{i}$-tej vrstvy $\boldsymbol{\Delta T^i}$ je vyjadrené ako rozdiel časov príchodu prvého platného údaju $\boldsymbol{(i+1)}$-tej vrstvy $\boldsymbol{T_1^{i+1}}$ a $\boldsymbol{i}$-tej vrstvy $\boldsymbol{T_1^i}$ ($\boldsymbol{\Delta T^i = T_1^{i+1} - T_1^i}$). Z dôvodu počiatočnej inicializácie modelu vchá-

dza prvý platný vstupný pixel do prvej vrstvy modelu (KONV-1) až v čase $\boldsymbol{T_1^1 = 4}$.

Tabuľka 3: Oneskorenia subsystémov navrhnutého modelu

	KONV-1	**KONV-2**	**KONV-3**	**FC**	**VÝSTUP**
$\boldsymbol{T_1^i}$	4	35	72	219	226
$\boldsymbol{\Delta T^i}$	31	37	147	7	-

5.2.2 Popis testovaného modelu s použitím HDL

Po overení správania jednotlivých subsystémov navrhnutého modelu v predchádzajúcej podkapitole 5.2.1, bol vytvorený popis tohto modelu v jazyku VHDL. Pre napísanie VHDL kódu, vykonanie jeho simulácie a následnej syntézy bol použitý nástroj *Vivado Design Suite v2017.4*.

Spotreba FPGA zdrojov Spotreba zdrojov FPGA pre jednotlivé subsystémy je zhrnutá v tab. 4, pričom vyjadruje ich **plne-paralelnú** implementáciu. Hodnoty vyjadrujú počet použitých jednotiek daného zdroja pre implementáciu subsystému danej vrstvy do FPGA obvodu. Spotreba zdrojov v prípade konvolučných vrstiev zahŕňa aj použitie aktivačnej funkcie *ReLU*.

Tabuľka 4: Spotreba zdrojov FPGA jednotlivých vrstiev navrhnutého modelu. Pre vyjadrenie závislosti medzi parametrami vrstvy a použitými FPGA zdrojmi pre implementáciu prislúchajúceho subsystému sú v tabuľke uvedené parametre vrstvy: **H** (počet vstupných príznakových máp), **L** (počet výstupných príznakových máp) a v prípade konvolučnej vrstvy aj **K** (rozmer matice váh).

	H	**L**	**K**	**LUT**	**LUTRAM**	**FF**	**DSP**	**IO**
KONV-1	1	16	3	750	320	818	144	173
KONV-2	16	16	3	7627	0	17784	2304	324
KONV-3	16	16	5	22989	10240	25460	6400	324
FC	16	10	-	4312	0	3760	0	263
RELU	-	-	-	5	0	0	0	20

Vzhľadom na hodnoty tabuľky predstavujú DSP bloky kritický zdroj FPGA obvodu v prípade konvolučnej siete, a preto sú použité len v implementácii numerických operácií prvej časti 2D konvolútora. Ostatné numerické operácie sú implementované prostredníctvom LUT. Spotreba DSP blokov pre konkrétny subsystém konvolučnej vrstvy je tak daná výrazom $\boldsymbol{H \times L \times K^2}$, kde $\boldsymbol{H}$ a $\boldsymbol{L}$ vyjadrujú počet vstupných a výstupných príznakových máp, a $\boldsymbol{K}$ je šírka váhových matíc.

Vychádzajúc z hodnôt tab. 4 je ideálny úplne paralelný návrh implementovateľný len do najvýkonnejších FPGA obvodov z dôvodu vysokých požiadaviek na DSP bloky. Preto je jedným z riešení úprava úplne paralelného návrhu na **čiastočne paralelný návrh**. Ide o náhradu viacerých funkčne rovnakých

výpočtových blokov za jeden, napr. 2D konvolútory v subsystéme konvolučnej vrstvy. Tento funkčný blok sekvenčne vykonáva výpočty, ktoré boli pôvodne pridelené celej skupine blokov. Tento prístup je kompromisom medzi spotrebou FPGA zdrojov a oneskorením systému.

Výkon systému je v práci vyjadrený jednotkou **FMA** (angl. *Fused Multiply Add*). Ide o počet kombinovaných operácií, pozostávajúcich z po sebe idúcich operácií násobenia a sčítania. V prípade FPGA obvodov od výrobcu *Xilinx* môžeme vyjadriť výkon systému vzťahom $\boldsymbol{P}_{\mathrm{DSP}} \times \boldsymbol{F}_{\mathrm{max}}$ ako súčin počtu DSP blokov, dostupných v FPGA obvode a použitých v implementácii systému, a ich maximálnej operačnej frekvencie. $\boldsymbol{F}_{\mathrm{max}}$ DSP blokov v FPGA obvode závisí od typu obvodu a jeho rýchlostnej triedy (angl. *speed grade*). Podľa tab. 4 používa navrhnutý systém spolu 8848 DSP blokov, teda jeho teoretický výkon je $\mathbf{8848} \times \boldsymbol{F}_{\mathrm{max}}$. Teoretický výkon systému a maximálny výkon FPGA obvodu pre zvolené príklady FPGA obvodov sú uvedené v tab. 5.

Tabuľka 5: Prehľad výkonu podľa FPGA obvodu

	XC7A100T-1	**XC7K325T-1**
Počet DSP blokov	240	840
$\boldsymbol{F}_{\mathrm{max}}$	464,25 Mhz [59]	547,95 Mhz [60]
Teoretický výkon	4107,6 GFMA/s	4848,3 GFMA/s
Maximálny výkon	111,4 GFMA/s	460,3 GFMA/s

Záver

Predkladaná práca sa venovala problematike detekcie sieťových útokov. Rešeršou aktuálneho stavu v oblasti detekcie sieťových útokov typu DoS/DDoS a nových prístupov založených na strojovom učení sme uviedli trendy v zdokonaľovaní detekčných metód (kap. 1.2.2). Vychádzajúc z týchto informácií bol stanovený primárny cieľ práce, ktorý je zameraný na *vytvorenie metodiky návrhu detektora DoS/DDoS útokov na báze analýzy paketových tokov s použitím strojového učenia* (kap. 2). Ako metóda pre návrh detektora bola zvolená konvolučná neurónová sieť z oblasti hlbokého učenia.

Na základe výsledkov vyššie uvedenej analýzy sme sformulovali metodiku návrhu detektora sieťových útokov (kap. 3). Metodika bola vybudovaná na piatich primárnych krokoch. Predlohou špecifikácie jednotlivých krokov sa stala oblasť rozpoznávania vzorov. Každý krok metodiky bol navrhnutý so zámerom dosiahnuť čo najúčinnejšiu detekciu DoS/DDoS útokov v počítačovej sieti. Metodika a mapovanie všeobecných krokov rozpoznávania vzorov do oblasti detekcie sieťových útokov predstavujú jeden z prínosov nami predkladanej práce.

Z dôvodu náročnosti jednotlivých krokov metodiky sa práca detailnejšie zaoberala len návrhom systému konvolučnej siete a jeho implementáciou do obvodu FPGA (kap. 4). V návrhu systému konvolučnej siete bol použitý systémový prí-

stup, ktorý viedol ku špecifikácii subsystémov. Najväčšia pozornosť bola venovaná subsystému konvolučnej vrstvy. Originálny prístup k návrhu 2D konvolútora (kap. 4.1) považujeme za jeden z kľúčových prínosov predkladanej práce. Pre popis návrhu subsystémov boli použité dva prístupy: modelovo-riadený prístup v nástroji Matlab/Simulink a RTL prístup v jazyku VHDL. Funkčnosť subsystémov bola overená formou simulácie. Zhoda v správaní referenčného modelu a simulačných modelov potvrdila korektnú funkciu navrhnutých subsystémov. Ich správanie je možné meniť zmenou hodnoty generických parametrov. Okrem rozmerov dátových vstupov sú nastaviteľné aj počty a rozmery vstupných a výstupných príznakových máp.

Metodika návrhu detektora tvorí základ riešenia, ale nepokrýva úplne problém so sieťovými útokmi typu DoS/DDoS v počítačovej sieti. Návrh a implementácia špecifickej architektúry konvolučnej siete s použitím v práci uvedených subsystémov by viedli k rozšíreniu tretieho a štvrtého kroku metodiky. Pokrok nastal aj v používaní bezdrôtových sietí, ktoré prinášajú ďalšie, doposiaľ neriešené problémy zabezpečenia počítačových sietí. Hlbší rozbor problematiky detekcie sieťových útokov z pohľadu bezdrôtových sietí by rozšíril prvý a druhý krok metodiky. Vyššie uvedené body rozšírenia práce otvárajú možnosti ďalšieho výskumu.

Zoznam použitej literatúry

[1] M. Handley a E. Rescorla, "Internet denial-of-service considerations", RFC Editor, RFC 4732, dec. 2006.

[2] V. Zlomislić, K. Fertalj a V. Sruk, "Denial of service attacks: An overview", in *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, jún 2014, s. 1–6. DOI: 10.1109/CISTI.2014.6876979.

[3] Neustar, "Worldwide DDoS Attacks & Protection Report - A Steady State of Threats in the Connected World", tech. spr. October, 2016.

[4] ——, "The threatscape widens: DDoS aggression and the evolution of IoT risks", tech. spr. April, 2016.

[5] D. Holmes, "2016 DDoS Attack Trends", F5 Networks, Inc., tech. spr. August, 2016, s. 1–7.

[6] M. Geva, A. Herzberg a Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses", *IEEE Security Privacy*, roč. 12, č. 1, s. 54–61, jan. 2014, ISSN: 1540-7993. DOI: 10.1109/MSP.2013.55.

[7] S. Dua a X. Du, *Data Mining and Machine Learning in Cybersecurity*, 1st. Boston, MA, USA: Auerbach Publications, 2011, ISBN: 9781439839423.

[8] D. K. Bhattacharyya a J. K. Kalita, *Network Anomaly Detection: A Machine Learning Perspective*. Chapman & Hall/CRC, 2013, ISBN: 9781466582088.

[9] C. Cortes a V. Vapnik, "Support-vector networks", *Machine Learning*, roč. 20, č. 3, s. 273–297, sept. 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018. url: https://doi.org/10.1007/BF00994018.

[10] T. Kohonen, "The self-organizing map", *Proceedings of the IEEE*, roč. 78, č. 9, s. 1464–1480, sept. 1990, ISSN: 0018-9219. DOI: 10.1109/5.58325.

[11] K. Patel a B. Buddhadev, "Machine learning based research for network intrusion detection: A state-of-the-art", roč. 3, s. 31–50, jún 2014.

[12] R. Vijayasarathy, S. V. Raghavan a B. Ravindran, "A system approach to network modeling for ddos detection using a naìve bayesian classifier", in *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, jan. 2011, s. 1–10. DOI: 10.1109/COMSNETS.2011.5716474.

[13] G. Kumar a K. Kumar, "Design of an evolutionary approach for intrusion detection", *The Scientific World Journal*, roč. 2013, 2013, ISSN: 1537744X. DOI: 10.1155/2013/962185.

[14] V. Chandola, A. Banerjee a V. Kumar, "Anomaly detection: A survey", *ACM Comput. Surv.*, roč. 41, č. 3, 15:1–15:58, júl 2009, ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. url: http://doi.acm.org/10.1145/1541880.1541882.

[15] A. Osareh a B. Shadgar, "Intrusion detection in computer networks based on machine learning algorithms", in *International Journal of Computer Science and Network Security, VOL.8 No.11*, 2008.

[16] G. Kim, S. Lee a S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection", *Expert Syst. Appl.*, roč. 41, č. 4, s. 1690–1700, mar. 2014, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.08.066. url: http://dx.doi.org/10.1016/j.eswa.2013.08.066.

[17] C. She, W. Wen, Z. Lin a K. Zheng, "Application-Layer DDOS Detection Based on a One-Class Support Vector Machine", *International Journal of Network Security & Its Applications*, roč. 9, č. 1, s. 13–24, jan. 2017, ISSN: 09752307. DOI: 10.5121/ijnsa.2017.9102. url: http://www.aircconline.com/ijnsa/V9N1/9117ijnsa02.pdf.

[18] Alfantookh a A. A., "Dos attacks intelligent detection using neural networks", *J. King Saud Univ. Comput. Inf. Sci.*, roč. 18, s. 31–51, jan. 2006, ISSN: 1319-1578. DOI: 10.1016/S1319-1578(06)80002-9. url: http://dx.doi.org/10.1016/S1319-1578(06)80002-9.

[19] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi a M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking", in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, okt. 2016, s. 258–263. DOI: 10.1109/WINCOM.2016.7777224.

[20] M. Wei, J. Su, J. Jin a L. Wang, "Research on intrusion detection system based on BP neural network", roč. 270 LNEE, č. VOL. 1, s. 657–663, 2014, ISSN: 18761119. DOI: 10.1007/978-3-642-40618-8_85.

[21] J. Li, Y. Liu a L. Gu, "Ddos attack detection based on neural network", in *2010 2nd International Symposium on Aware Computing*, nov. 2010, s. 196–199. DOI: 10.1109/ISAC.2010.5670479.

[22] A. Mitrokotsa a C. Douligeris, "Detecting denial of service attacks using emergent self-organizing maps", in *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.*, dec. 2005, s. 375–380. DOI: 10.1109/ISSPIT.2005.1577126.

[23] C.-d. Wang, H.-f. Yu, H.-b. Wang a K. Liu, "Som-based anomaly intrusion detection system", in *Embedded and Ubiquitous Computing*, T.-W. Kuo, E. Sha, M. Guo, L. T. Yang a Z. Shao, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, s. 356–366, ISBN: 978-3-540-77092-3.

[24] M. Kim, S. Jung a M. Park, "A distributed self-organizing map for dos attack detection", in *2015 Seventh International Conference on Ubiquitous and Future Networks*, júl 2015, s. 19–22. DOI: 10.1109/ICUFN.2015.7182487.

[25] Y. LeCun a Y. Bengio, "Convolutional networks for images, speech, and time-series", English (US), in *The handbook of brain theory and neural networks*, M. Arbib, ed. MIT Press, 1995.

[26] Y. LeCun, Y. Bengio a G. Hinton, "Deep learning", *Nature*, roč. 521, s. 436, máj 2015. DOI: 10.0.4.14/nature14539.

[27] V. Sze, Y. H. Chen, T. J. Yang a J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey", *Proceedings of the IEEE*, roč. 105, č. 12, s. 2295–2329, dec. 2017, ISSN: 0018-9219. DOI: 10.1109/JPROC.2017.2761740.

[28] Y. LeCun, L. Bottou, Y. Bengio a P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, roč. 86, č. 11, s. 2278–2324, nov. 1998, ISSN: 0018-9219. DOI: 10.1109/5.726791.

[29] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn a D. Yu, "Convolutional neural networks for speech recognition", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, roč. 22, č. 10, s. 1533–1545, okt. 2014, ISSN: 2329-9290. DOI: 10.1109/TASLP.2014.2339736.

[30] X. Yang, V. De Andrade, W. Scullin, E. L. Dyer, N. Kasthuri, F. De Carlo a D. Gürsoy, "Low-dose x-ray tomography through a deep convolutional neural network", *Scientific Reports*, roč. 8, č. 1, s. 2575, 2018, ISSN: 2045-2322. DOI: 10.1038/s41598-018-19426-7. url: https://doi.org/10.1038/s41598-018-19426-7.

[31] J. D. Dormer, M. Halicek, L. Ma, C. M. Reilly, E. Schreibmann a B. Fei, "Convolutional neural networks for the detection of diseased hearts using ct images and left atrium patches", zv. 10575, 2018, s. 10 575–10 575. DOI: 10.1117/12.2293548. url: https://doi.org/10.1117/12.2293548.

[32] D. Gibert Llauradó, "Convolutional neural networks for malware classification", diz. pr., Universitat Politecnica de Catalunya, 2016.

[33] E. Carabez, M. Sugi, I. Nambu a Y. Wada, "Convolutional neural networks with 3d input for p300 identification in auditory brain-computer interfaces", *Computational Intelligence and Neuroscience*, roč. 2017, s. 9, 2017. DOI: 10.1155/2017/8163949. url: https://www.hindawi.com/journals/cin/2017/8163949/.

[34] C. Leondes, *Image Processing and Pattern Recognition*, ed. Neural Network Systems Techniques and Applications. Elsevier Science, 1998, ISBN: 9780080551449. url: https://books.google.sk/books?id=oDewAeVxr-4C.

[35] U. of California. (1999). Kdd cup 1999 data, url: http://kdd.ics.uci.edu/databases/kddcup99/task.html (cit. 19. 04. 2018).

[36] MIT Lincoln Laboratory. (1998). DARPA intrusion detection evaluation, url: https://www.ll.mit.edu/ideval/docs/index.html (cit. 19. 04. 2018).

[37] M. Tavallaee, E. Bagheri, W. Lu a A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set", in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, júl 2009, s. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[38] A. Shiravi, H. Shiravi, M. Tavallaee a A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", *Computers and Security*, roč. 31, č. 3, s. 357–374, 2012, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2011.12.012. url: http://www.sciencedirect.com/science/article/pii/S0167404811001672.

[39] J. J. Davis a A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review", *Comput. Secur.*, roč. 30, č. 6-7, s. 353–375, sept. 2011, ISSN: 0167-4048. DOI: 10.1016/j.cose.2011.05.008. url: http://dx.doi.org/10.1016/j.cose.2011.05.008.

[40] S. Brugger, "Data mining methods for network intrusion detection", 2004.

[41] D. Kwon, H. Kim, J. Kim, S.-c. Suh, I. Kim a K. Kim, "A survey of deep learning-based network anomaly detection", sept. 2017.

[42] F. Chollet, *Deep Learning with Python*, 1st. Greenwich, CT, USA: Manning Publications Co., 2017, ISBN: 9781617294433.

[43] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat a P. V. Krishna, "A deep learning based artificial neural network approach for intrusion detection", in *Mathematics and Computing*, D. Giri, R. N. Mohapatra, H. Begehr a M. S. Obaidat, ed., Singapore: Springer Singapore, 2017, s. 44–53, ISBN: 978-981-10-4642-1.

[44] G. Roudière a P. Owezarski, "A lightweight snapshot-based ddos detector", in *2017 13th International Conference on Network and Service Management (CNSM)*, nov. 2017, s. 1–7. DOI: 10.23919/CNSM.2017.8256014.

[45] G. Lacey, G. W. Taylor a S. Areibi, "Deep learning on fpgas: Past, present, and future", *CoRR*, roč. abs/1602.04283, 2016.

[46] K. Abdelouahab, M. Pelcat, J. Sérot a F. Berry, "Accelerating cnn inference on fpgas: A survey", 2018.

[47] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra a G. Boudoukh, "Can fpgas beat gpus in accelerating next-generation deep neural networks?", in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ed. FPGA '17, Monterey, California, USA: ACM, 2017, s. 5–14, ISBN: 978-1-4503-4354-1. DOI: 10.1145/3020078.3021740. url: http://doi.acm.org/10.1145/3020078.3021740.

[48] M. Jahre, "Using fpgas to accelerate neural network inference", Invited presentation at RC4DL, Ghent, Belgium, sept. 2017, url: http://www.ece.ucy.ac.cy/labs/easoc/RC4DL/Magnus%20Jahre.pdf.

[49] P. P. Chu, *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*. Wiley-IEEE Press, 2006, ISBN: 0471720925.

[50] H. T. Kung, "Why systolic architectures?", *Computer*, roč. 15, č. 1, s. 37–46, jan. 1982, ISSN: 0018-9162. DOI: 10.1109/MC.1982.1653825. url: https://doi.org/10.1109/MC.1982.1653825.

[51] J. Hrabovsky, P. Segec, M. Moravcik a J. Papan, "Systolic-based 2d convolver for cnn in fpga", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, okt. 2017, s. 1–7. DOI: 10.1109/ICETA.2017.8102485.

[52] Xilinx a G. C. Hawkes, *Dsp solutions – advanced design guide edition 1.0 dsp: Designing for optimal results high-performance dsp using virtex-4 fpgas*, 2005.

[53] Xilinx, *Ug479 7 series dsp48e slice user guide*, 2016.

[54] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. Wiley-IEEE Press, 2011, s. 416, ISBN: 978-0470828496. DOI: 10.1002/9780470828519. url: http://onlinelibrary.wiley.com/book/10.1002/9780470828519.

[55] J. Chang a J. Sha, "An efficient implementation of 2d convolution in cnn", *IEICE Electronics Express*, roč. 14, č. 1, s. 1–8, 2017. DOI: 10.1587/elex.13.20161134.

[56] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao a J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks", in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ed. FPGA '15, Monterey, California, USA: ACM, 2015, s. 161–170, ISBN: 978-1-4503-3315-3. DOI: 10.1145/2684746.2689060. url: http://doi.acm.org/10.1145/2684746.2689060.

[57] R. Aarenstrup, *Managing Model-Based Design*, 1st. CreateSpace Independent Publishing Platform, aug. 2015, s. 100, ISBN: 978-1512036138.

[58] S. Churiwala, *Designing with Xilinx FPGAs: Using Vivado*, 1st. Springer Publishing Company, Incorporated, 2016, ISBN: 978-3319424378. DOI: 10.1007/978-3-319-42438-5.

[59] Xilinx, *Ds181 (v1.25) - artix-7 fpgas data sheet: Dc and ac switching characteristics*, 2018. url: https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf.

[60] ——, *Ds182 (v2.16.1) - kintex-7 fpgas data sheet: Dc and ac switching characteristics*, 2018. url: https://www.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf.

Publikácie

[1] M. Moravčík, P. Segeč, P. Palúch, J. Hrabovský a J. Papán, "Clouds in educational process", in *2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov. 2015, s. 1–7. DOI: 10.1109/ICETA.2015.7558500.

[2] J. Papán, M. Drozdová, P. Segeč, Ľ. Mikuš a J. Hrabovský, "The new pim-sm ipfrr mechanism", in *2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov. 2015, s. 1–7. DOI: 10.1109/ICETA.2015.7558504.

[3] M. Moravčík, P. Segeč, J. Hrabovský, J. Papán a J. Uramová, "Survey of real-time multimedia security mechanisms", in *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov. 2016, s. 233–238. DOI: 10.1109/ICETA.2016.7802097.

[4] J. Papán, P. Segeč, M. Drozdová, Ľ. Mikuš, M. Moravčík a J. Hrabovský, "The ipfrr mechanism inspired by bier algorithm", in *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov. 2016, s. 257–262. DOI: 10.1109/ICETA.2016.7802053.

[5] J. Hrabovsky, P. Segec, P. Paluch, M. Moravcik a J. Papan, "Usability of the sip protocol within smart home solutions", *Communications - Scientific letters of the University of Zilina*, roč. 18, č. 1A, s. 4–12, mar. 2016. url: http://komunikacie.uniza.sk/index.php/communications/article/view/352.

[6] J. Papán, P. Segeč, M. Moravčík, J. Hrabovský, Ľ. Mikuš a J. Uramova, "Existing mechanisms of ip fast reroute", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, okt. 2017, s. 1–7. DOI: 10.1109/ICETA.2017.8102516.

[7] M. Moravčík, P. Segeč, J. Papán a J. Hrabovský, "Overview of cloud computing and portability problems", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, okt. 2017, s. 1–6. DOI: 10.1109/ICETA.2017.8102511.

[8] P. Segeč, M. Moravčík, J. Hrabovský, J. Papán a J. Uramová, "Securing sip infrastructures with pki — the analysis", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, okt. 2017, s. 1–8. DOI: 10.1109/ICETA.2017.8102525.

[9] J. Hrabovsky, P. Segec, M. Moravcik a J. Papan, "Systolic-based 2d convolver for cnn in fpga", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, okt. 2017, s. 1–7. DOI: 10.1109/ICETA.2017.8102485.

[10] J. Hrabovsky, P. Segec, M. Moravcik a J. Papan, "Trends in application of machine learning to network-based intrusion detection systems", in *Innovations for Community Services*, M. Hodon, G. Eichler, C. Erfurth a G. Fahrnberger, ed., Cham: Springer International Publishing, 2018, s. 218–228, ISBN: 978-3-319-93408-2.

[11] J. Hrabovský, M. Kontšek, P. Segeč a O. Šuch, "Influence of positive additive noise on classification performance of convolutional neural networks", in *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, aug. 2018, s. 175–180. DOI: 10.1109/DISA.2018.8490611.

[12] M. Klimo, O. Škvarek, P. Tarábek, O. Šuch a J. Hrabovsky, "Nearest neighbor classification in minkowski quasi-metric space", in *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, aug. 2018, s. 227–232. DOI: 10.1109/DISA.2018.8490622.

[13] J. Uramová, P. Scgeč, M. Moravčík, J. Papán, M. Kontšek a J. Hrabovský, "Infrastructure for generating new ids dataset", in *2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov. 2018, s. 603–610. DOI: 10.1109/ICETA.2018.8572201.