ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

Detekcia sieťových útokov vo vysoko-rýchlostných počítačových sieťach

Dizertačná práca

Evidenčné číslo: 28360020193008

Študijný program: Aplikovaná informatika
Študijný odbor: 9.2.9 aplikovaná informatika
Pracovisko: Katedra informačných sietí
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline
Školiteľ: doc. Mgr. Ondrej Šuch, PhD.
Školiteľ špecialista: doc. Ing. Pavel Segeč, PhD.

Žilina, 2019 **Ing. Jakub Hrabovský**

Poďakovanie

Moje poďakovanie patrí školiteľovi práce doc. Mgr. Ondrejovi Šuchovi, PhD. a školiteľovi špecialistovi doc. Ing. Pavlovi Segečovi, PhD. za ich pomoc, odborné usmernenie, cenné rady, trpezlivosť a čas pri tvorbe tejto práce. Špeciálne chcem poďakovať Ing. Oldřichovi Kovářovi, PhD. za jeho čas a trefné pripomienky pri tvorbe textu ako aj praktickej časti práce, ktoré viedli k zlepšeniu kvality práce a jej výsledkov. Chcel by som poďakovať aj celému kolektívu katedry informačných sietí za podporu, rady a technické vybavenie, bez ktorých by práca nemohla vzniknúť. Veľká vďaka patrí najmä mojej rodine, priateľom ale aj všetkým, ktorí ma podporovali počas celého štúdia a neumožnili mi ukončiť štúdium predčasne.

Abstrakt

JAKUB HRABOVSKÝ: Detekcia sieťových útokov vo vysoko-rýchlostných počítačových sieťach [Dizertačná práca] - Žilinská univerzita v Žiline. Fakulta riadenia a informatiky. Katedra informačných sietí. - Školiteľ: doc. Mgr. Ondrej Šuch, PhD. - Stupeň odbornej kvalifikácie: Doktor filozofie v študijnom odbore 9.2.9 aplikovaná informatika. - Žilina, FRI ŽU, apríl 2019, 152 s.

Nedostatočná kvalita zabezpečenia súčasných sieťových služieb, spôsobená najmä výskytom masívnych sieťových útokov typu DoS/DDoS, vedie často k nedostupnosti týchto služieb. Detekcia sieťových útokov spadá do oblasti bezpečnosti počítačových sietí a predstavuje problém, ktorému sa venuje aj predkladaná práca. Cieľom práce je vytvorenie metodiky návrhu detektora DoS/DDoS útokov s použitím strojového učenia vo vysokorýchlostnej 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 obvodov. V návrhu tohto systému je použitý systémový prístup, ktorý viedol ku špecifikácii jednotlivých subsystémov. Najväčšia pozornosť je venovaná návrhu originálnej štruktúry 2D konvolútora, ako kľúčového výpočtového prvku konvolučnej siete. 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, distribuované odopretie služby, systém detekcie sieťových prienikov, hlboké učenie, konvolučná neurónová sieť, programovateľné hradlové polia

Abstract

JAKUB HRABOVSKÝ: Network-based Intrusion Detection in High-Speed Computer Networks [Dissertation thesis] - The University of Žilina in Žilina. Faculty of Management Science and Informatics. Department of InfoComm Networks. - Supervisor: doc. Mgr. Ondrej Šuch, PhD. - Qualification level: Philosophiae doctor in the study field 9.2.9 Applied Informatics. - Žilina, FRI ZU, april 2019, 152 p.

Unsatisfactory quality of security in current network services, caused primarily by massive computer network intrusions such as DoS/DDoS attacks, leads often to an unavailability of these services. Network intrusion detection is a part of computer network security field and represents a problem that is also addressed in this thesis. 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 circuits. The systematic approach used in this system design helped in a specification of the individual subsystems. The most attention is given to the novel structure of 2D convolver as a key processing element of the convolutional network. 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.

Keywords: network intrusion, denial of service, distributed denial of service, network-based intrusion detection system, deep learning, convolutional neural network, field programmable gate arrays

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

ACAP Adaptive Compute Acceleration Platform

ANN Artificial Neural Network (umelá neurónová sieť)

ASIC Application-Specific Integrated Circuit (aplikačne-špecifický integrovaný obvod)

BMU Best-Match Unit

BN Bayesian Network (Bayesovská sieť)

CC Cloud Computing

CE Clock Enable (povoľovací signál hodín)

CLB Configurable Logic Block (konfigurovateľný logický blok)

CSG Cell Splitting Grid (mriežka deliacich buniek)

DDoS Distributed Denial of Service (distribuované odopretie služby)

DNS Domain Name System

DoS Denial of Service (odopretie služby)

DSOM Distributed Self-Organizing Map (distribuovaná samo-organizujúca mapa)

DSP Digital Signal Processing (číslicové spracovanie signálov)

ESOM Emergent Self-Organizing Map

FFT Fast Fourier Transform (rýchla Fourierová transformácia)

FMA Fused Multiply Add

FN False Negative

FNR False Negative Rate

FP False Positive

FPGA Field Programmable Gate Arrays (programovateľné hradlové polia)

FPR False Positive Rate

FSM Finite State Machine (konečný stavový automat)

GA Genetic Algorithm (genetický algoritmus)

GHSOM Growing Hierarchical Self-Organizing Map (rastúca hierarchická samo-organizujúca mapa)

GPU Graphics Processing Unit (grafický procesor)

HDL Hardware Description Language

HIDS Host-based Intrusion Detection System

HMM Hidden Markov Model

HSOM Hierarchical Self-Organizing Map (hierarchická samo-organizujúca mapa)

ICMP Internet Control Message Protocol

IDS Intrusion Detection System (systém detekcie prienikov)

IKT Informačno Komunikačné Technológie

IoT Internet of Things (internet vecí)

IP Internet Protocol

LUT Look-Up Table

MAC Multiply And Accumulate

ML Machine Learning (strojové učenie)

MSE Mean Squared Error (kvadratická účelová funkcia)

NB Naive Bayes (naivný Bayesov klasifikátor)

NIDS Network-based Intrusion Detection System (systém detekcie sieťových prienikov)

NTP Network Time Protocol

OSI Open Systems Interconnection

PCA Principal Component Analysis (analýza hlavných komponentov)

PE Processing Element (výpočtový prvok)

PSO Particle Swarm Optimization

PSVM Plane-based one-class Support Vector Machine (rovinovo-založené one-class SVM)

R2L Remote-to-Local

RBF Radial Basis Function (radiálna bázová funkcia)

ReLU Rectified Linear Unit (rektifikovaná lineárna jednotka)

RMSE Root Mean Square Error (stredná kvadratická chyba)

RPC Remote Procedure Call (Vzdialené volanie procedúry)

RTL Register-Transfer Level

SE Systolic Element (systolický prvok)

SGD Stochastic Gradient Descent (stochastická metóda najstrmšieho spádu)

SIMD Single Instruction Multiple Data

SOM Self-Organizing Map (samo-organizujúca mapa)

SSDP Simple Service Discovery Protocol

SVDD Support Vector Data Description (Popis dát podporných vektorov)

SVM Support Vector Machine (Podporné vektory)

TCP/IP Transmission Control Protocol/Internet Protocol

TN True Negative

TP True Positive

U2R User-to-Root

UDP User Datagram Protocol

VHDL Very High Speed Integrated Circuit HDL

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. Dostupnosť je jednou z troch základných vlastností **bezpečnej** počítačovej siete (popri dôvernosti a integrite). 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. V súčasnosti sa bezpečnosť počítačových sietí zaoberá otázkami viacerých problémových oblastí, na ktoré sa v práci pozeráme z niekoľkých pohľadov [1]:

- **Implementačný pohľad**: Návrh bezchybných systémov je nereálnou výzvou kvôli vysokým požiadavkám a nákladom, ktoré takýto návrh vyžaduje. To znamená, že reálny systém vždy obsahuje bezpečnostné slabiny a zraniteľnosti, ktoré útočník nájde a zneužije, dnes alebo v budúcnosti.

- **Pohľad útočníka**: Útočník neustále zvyšuje úroveň svojich schopností a rozvíja nové útoky rovnakým tempom ako vznikajú nové ochranné metódy. Priestor hrozieb je tak veľmi veľký vzhľadom na otvorenosť a rôznorodosť počítačovej siete a jej služieb.

- **Pohľad obete**: Sieťová architektúra TCP/IP a súvisiaca globálna sieť, Internet, sú založené na princípe otvorenosti, ktorý vyžaduje od sieťových zariadení spracovať prichádzajúcu sieťovú prevádzku bez ohľadu na jej pôvod. Princíp otvorenosti umožňuje tiež skryť zdrojovú identitu odosielateľa úpravou a odoslaním paketov

s vlastným obsahom. Príkladom je technika falšovania IP adries (angl. *IP spoofing*). Zložitosť overenia pravosti sťažuje identifikáciu a sledovanie útočníka.

- **Pohľad úplnosti**: Napriek veľkému a neustále sa zvyšujúcemu počtu existujúcich ochranných metód neexistuje univerzálna metóda, ktorá poskytuje úplné zabezpečenie a ochranu počítačovej siete pred všetkými možnými hrozbami.
- **Pohľad sieťovej infraštruktúry**: Infraštruktúra súčasných sietí je zložitý a veľmi dynamický systém. S príchodom nových technológií sa prostredie sietí rýchlo mení – dochádza k nárastu počtu pripojených zariadení, rýchlosti liniek, a objemu prenášaných dát (narastajúci objem celkovej sieťovej prevádzky umožňuje útokom ľahšie sa maskovať v rámci normálnej prevádzky). To má za následok oveľa vyššie nároky na ochranné metódy a ich výkon v porovnaní s nárokmi v minulosti.

Aktuálne dostupné ochranné metódy neriešia spomenuté problémy a tak je potreba navrhnúť nové metódy prevencie, detekcie a reakcie na sieťové útoky s cieľom udržať krok s útočníkmi. Preto sa predložená práca zameriava na detekčné metódy a možnosti ich zdokonaľovania s cieľom odstrániť niektoré z uvedených problémov zabezpečenia počítačových sietí. 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).

Štruktúra práce

Predkladaná práca pozostáva z piatich kapitol. Aktuálny stav zvolenej problematiky detekcie sieťových prienikov s použitím metód strojového učenia je popísaný v **prvej kapitole**. Popis problematiky zdôrazňuje nebezpečenstvo existujúcich DoS/DDoS útokov a uvádza metódy detekcie sieťových prienikov vrátane trendov v návrhu detekčných mechanizmov. Súčasťou problematiky sú základné princípy v práci zvolenej metódy hlbokého učenia - konvolučnej neurónovej siete. Hlavný cieľ práce je uvedený v **druhej kapitole**. Súčasťou kapitoly je stanovená postupnosť krokov pre dosiahnutie zvoleného cieľa. Hlavný text práce, popisujúci metodiku návrhu detekčnej metódy, sa nachádza v **tretej kapitole**. Štruktúra metodiky je daná etapami, z ktorých metodika pozostáva. Každej z etáp je pridelená samostatná podkapitola s jej detailnejším popisom. V **štvrtej kapitole** práca popisuje architektúry výpočtových subsystémov konvolučnej neurónovej siete, ktorých návrh odpovedá jednej z etáp. Pre každý zo základných typov vrstiev konvolučnej siete je navrhnutý subsystém, ktorý poskytuje funkcie daného typu vrstvy a je implementovateľný do obvodu programovateľných hradlových polí (angl. *Field Programmable Gate Array*; FPGA). K pochopeniu štruktúry a správania subsystémov, popísaných v práci,

prispievajú viaceré schémy. Priebeh a výsledky experimentálneho overenia navrhnutých subsystémov sú popísané v **piatej kapitole**. Prínosy predkladanej práce do oblasti detekcie sieťových prienikov sú zhrnuté v závere.

Kapitola 1

Aktuálny stav zvolenej problematiky

Vzhľadom na dôležitosť komunikačných sietí, narastajúce nebezpečenstvo DoS/DDoS útokov, a nedostatky súčasných detekčných metód, je návrh nových, účinnejších metód nutným krokom pre zachovanie dostupnosti IKT. V tejto kapitole sa venujeme aktuálnemu stavu problematiky v oblasti bezpečnosti počítačových sietí. Najprv sú analyzované prejavy DoS/DDoS útokov a ich základné vlastnosti, aby sme zdôraznili dôvody ich efektívnosti. Ďalej sa venujeme základnej klasifikácii detekčných metód. Na základe súčasného stavu v oblasti detekcie sieťových prienikov sme vybrali niektoré z metód strojového učenia, pričom sa primárne venujeme ich použitiu v úlohe detektora sieťových prienikov. Pre účely porovnania rôznych metód sú uvedené ich základné kvalitatívne parametre. Na základe analýzy viacerých detekčných metód, dostupných vo vedeckých článkoch, je vytvorený prehľad trendov pre návrh podľa nášho názoru účinnejšej detekčnej metódy. Vychádzajúc z uvedených trendov a zvoleného zamerania predloženej práce je v závere tejto kapitoly popísaná konvolučná neurónová sieť a trendy jej hardvérovej implementácie.

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 [2], [3]. Napriek intenzívnemu výskumu v oblasti sieťovej bezpečnosti, účinok týchto útokov narastá každý rok, čo potvrdzujú mnohé oficiálne správy [4]–[6]. Ako príklad uvádzame zhrnutie jednej z týchto správ [5] v niekoľkých bodoch, ktoré zdôrazňujú súčasný stav zabezpečenia (resp. nezabezpečenia) počítačovej siete v známych medzinárodných spoločnostiach. Správa stručne uvádza skúsenosti s útokmi typu DoS/DDoS viac ako tisíc spoločností v Severnej Amerike, Európe a Ázii v roku 2015. Viac ako 73% spoločností priznávajú, že sa stali obeťami útokov DoS/DDoS. Navyše, napadnuté spoločnosti ďalej priznávajú, že:

- 82% zaznamenali opakované napadnutia,
- 57% uviedli stratu zákazníckych údajov, finančného alebo intelektuálneho vlastníctva,
- 42% detegovali útoky DDoS vo vlastnej sieťovej infraštruktúre až po troch alebo viac hodinách,
- 45% majú okrem DoS/DDoS skúsenosti aj s inými formami útokov, ako sú vírusy a malvér,
- 44% boli o útoku informovaní zo strany zákazníkov a iných, tretích strán.

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**.

DDoS DDoS útok je hromadná synchronizovaná verzia DoS útoku, vykonaná sieťou infikovaných zariadení – *botov*. DDoS prebieha v štyroch fázach. V prvej fáze útočník prehľadáva sieť so zámerom odhaliť systémy s neriešenými hrozbami a otvorenými zraniteľnosťami, ktoré predstavujú potenciálne obete. V druhej fáze útočník rozšíri v sieti infekciu, aby získal kontrolu nad čo najväčšou skupinou zariadení. Útočník tak vytvorí svoju sieť infikovaných zariadení – *botnet*. Infekcia má formu škodlivého programu (angl. *malware*), ktorý zneužíva v minulosti nájdené bezpečnostné slabiny v systéme. V tretej fáze útočník založí skrytý komunikačný kanál, cez ktorý útočník riadi činnosť infikovaných zariadení. V štvrtej fáze je realizovaný samotný útok na obeť. Útočník posiela príkazy zariadeniam pod svojou kontrolou. Infikované zariadenia, bez svojho vedomia, vykonávajú rôzne akcie podľa prijatých príkazov, ktorých výsledkom je synchronizovaný DDoS útok. Skutočné nebezpečenstvo DDoS útokov spočíva v objeme umelo generovanej sieťovej prevádzky, ktorá zaplavuje obeť vrátane okolitého prostredia.

1.1.1 Typy DoS/DDoS útokov

Dostupné zdroje nie sú jednotné v klasifikácii DoS a DDoS útokov, používajú rôzne atribúty pre zadelenie rôznych útokov do tried. V práci uvádzame dve delenie DoS/DDoS útokov podľa [1], [7]. Podľa modelu OSI/ISO, jeho vrstiev a funkcií, rozlišujeme primárne **útoky sieťovej vrstvy** (angl. *network-layer*) a **útoky aplikačnej vrstvy** (angl. *application-layer*). Útoky sieťovej vrstvy pracujú na sieťovej a čiastočne na transportnej vrstve.

Ide prevažne o záplavové (volumetrické) útoky, ktoré zaplavia obeť veľkým objemom náhodne generovanej sieťovej prevádzky, napr. TCP-Syn záplava, UDP záplava, a ICMP záplava. Na druhej strane, útoky aplikačnej vrstvy sa spoliehajú na chyby a zraniteľnosti, prítomné v rôznych implementáciách aplikačných služieb. Podstatou aplikačných útokov je posielanie paketov s precízne vytvoreným obsahom, zameraným priamo na konkrétnu chybu alebo zraniteľnosť implementácie. Z tohto dôvodu je realizácia aplikačného útoku zložitejšia než v prípade útokov na sieťovej vrstve. Ako cieľ aplikačných útokov sa stávajú rôzne programy pre poskytovanie služieb, napr. HTTP, SIP, a DNS protokolov.

Volumetrické útoky (podkategória DoS/DDoS útokov), sa delia do troch tried podľa spôsobu, akým pristupujú k tvorbe záplavy [8]:

- **Priama záplava** (angl. *direct flooding*) je najjednoduchší typ záplavového útoku, založený na priamom generovaní veľkého objemu dát bez ich skrytého významu. Tento prúd dát zahltí obeť do takej miery, že obeťou poskytované služby už nie sú viac dostupné pre klientov. Príkladom tohto typu útokov sú SYN a DNS záplavy. SYN záplava sa sústredí na vyčerpanie maximálneho povoleného počtu otvorených TCP relácií. DNS záplava zneužíva nerovnomerné zaťaženie zariadenia počas generovania žiadosti na strane klienta a jej spracovania na strane servera.

- **Zosilnenie** (angl. *amplification*) zdokonaľuje priamu záplavu a zosilňuje jej vplyv na výkon obete. Primárnym zámerom je efektívne využitie dostupných zdrojov a základných princípov protokolu, kde platí "odpoveď je oveľa väčšia ako žiadosť". Pomer medzi veľkosťou odpovede a veľkosťou žiadosti sa nazýva *zosilňujúci faktor* a určuje efektivitu útoku. Ako príklad poslúži DNS protokol a jeho rozšírenia: EDNS a DNSSEC.

- **Odraz** (angl. *reflection*) zdokonaľuje zosilnenie vzdialeným zneužitím iných zariadení, pripojených do siete, bez ich vedomia. Princíp útoku spočíva v odosielaní upravených žiadostí na servery – *reflektory*, ktoré neúmyselne zahltia obeť generovanými odpoveďami. Požadované správanie reflektorov je zaručené použitím techniky falšovania IP adresy, konkr. vložením IP adresy obete do poľa zdrojovej adresy v IP hlavičke každej žiadosti.

Volumetrické útoky sú častou voľbou útočníkov z dôvodu ich jednoduchej realizácie a silného účinku, a predstavujú tak aj dnes veľkú hrozbu pre súčasné komunikačné siete. Popri DNS existujú aj iné protokoly, ktoré sú vhodnými kandidátmi pre použitie volumetrického útoku, napr. Network Time Protocol (NTP), Simple Service Discovery Protocol (SSDP), BitTorrent, NetBIOS, a Remote Procedure Call (RPC) portmap.

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 [8]. Detekcia je prvým krokom k vyriešeniu problému potlačenia sieťových útokov. Preto má detekcia vysokú prioritu nasadenia v reálnom prostredí. 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 [9], [10].

1.2.1 Klasifikácia detektorov sieťových prienikov podľa zdrojov dát

Podľa zdrojov, z ktorých pochádzajú odchytené dáta, a ich umiestnenia sú známe dve triedy detekčných metód: **host-based** a **network-based** systémy detekcie prienikov (angl. *Intrusion Detection System*; IDS) [11].

Host-based IDS

Host-based IDS (HIDS) [12] je nasadzovaný na koncových staniciach. HIDS pracuje s údajmi, ktoré zozbieral operačný systém a služby, bežiace na koncových staniciach (klientský počítač, server, tlačiareň, atď.) alebo na medziľahlých uzloch (prepínač, smerovač, a firewall). Údaje sú uchované vo forme záznamov (angl. *log*) a popisujú aktuálny stav systému [13]. Detekčné metódy tohto typu sa zameriavajú na detekciu útokov, ktoré pôsobia len na samotný monitorovaný systém. Všeobecne ide o útoky **Remote-to-Local** (R2L) a **User-to-Root** (U2R) [2]. Antivírusové systémy, firewally, a anti-spyware programy sú príkladmi HIDS.

Network-based IDS

Network-based IDS (NIDS) používa údaje, zozbierané zo sieťového prostredia vo forme odchytených paketov. Primárnymi úlohami NIDS sú separácia útokov od normálnej prevádzky a následná identifikácia paketov, resp. tokov, ktoré sú súčasťou útoku. Činnosť NIDS spočíva v tvorbe modelu, ktorý reprezentuje správanie konkrétneho typu sieťovej prevádzky – **profil**. Metóda musí vykonať celý proces detekcie rýchlo, keďže potenciálny

útok do značnej miery degraduje kvalitu služby (šírka pásma sieťového spojenia, latencia odpovede sieťových služieb, a pod.).

Druhá spomenutá operácia, identifikácia, je rovnako dôležitá až nevyhnutná pre vykonanie protiopatrení, ako sú filtrovanie a limitovanie prevádzky. V prípade pozitívnej detekcie útoku musí metóda zistiť podozrivú časť prevádzky a určiť jej parametre.

Dominantným typom útokov, ktoré sú detegovateľné metódami NIDS, sú **DoS/DDoS** a rôzne formy **prieskumu siete** (skenovanie siete a vyhľadávanie informácii o pripojených zariadeniach) [2].

1.2.2 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) [9]. Model, vytvorený signatúrnymi metódami, popisuje správanie podozrivej prevádzky. Na druhej strane, anomálne metódy sa pozerajú na model ako na dostatočného reprezentanta normálnej prevádzky. Kategórie sú v nasledujúcich podkapitolách popísané z pohľadu NIDS.

Signatúrne metódy

Signatúrne metódy sú silno závislé od znalosti o existujúcich útokoch, ktorých model je uložený v databáze. Databáza obsahuje profily všetkých známych útokov, popísaných formou signatúr. Signatúra má tvar usporiadaného zoznamu parametrov alebo príznakov s ich konkrétnymi hodnotami. Táto postupnosť hodnôt jednoznačne odlišuje prislúchajúci útok od ostatných. V prípade R2L a U2R útokov je napríklad možné použiť ako unikátny príznak postupnosť systémových volaní. Iným príkladom sú špecifické hodnoty polí v IP hlavičke paketov, odchytené zo sieťovej prevádzky.

Kľúčovou myšlienkou detekcie signatúrnymi metódami je analýza úsekov dátového toku, extrahovaných počas monitorovania sieťovej prevádzky. Metóda porovnáva vzorky so štruktúrou signatúr, ktoré sú načítané z databázy. V prípade zhody je daný dátový úsek označený ako útok, ktorému patrí pozitívna signatúra. Keďže signatúry detailne popisujú jednotlivé útoky, presnosť detekcie a klasifikácie je vysoká v závislosti od množstva a kvality uložených signatúr. Rôzne metódy realizujú reprezentáciu, spracovanie a vyhodnotenie signatúr rôzne.

Výkonnosť metódy závisí od databázy signatúr, a tak z dôvodu dosiahnutia čo najlepších výstupov vyžaduje jej pravidelnú aktualizáciu. Aktualizáciu vykonáva bezpečnostný

expert, ktorý je zodpovedný za pridávanie signatúr nových typov útokov. Kvalita obsahu databázy a dosiahnuté výsledky tak silno závisia od teoretických vedomostí a praktických skúseností bezpečnostných expertov. Táto skutočnosť je považovaná za vážnu nevýhodu signatúrnych metód, a preto je dôvodom prechodu na iný prístup k detekcii prienikov – anomálne metódy.

Anomálne metódy

Anomálne metódy sú považované za komplement signatúrnych metód, lebo 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. Model je pravidelne aktualizovaný, aby zodpovedal dynamickému správaniu súčasnej sieťovej infraštruktúry.

Metóda identifikuje anomáliu ako výraznú odlišnosť od normálnej sieťovej aktivity, určenej modelom. Príkladom sieťovej anomálie sú zariadenia s poruchou a záplavové sieťové útoky. Tvar a technika tvorby modelu závisia od zvolenej metódy.

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.

Datasety hrajú významnú úlohu počas testovania, kedy je hodnotená kvalita modelu. Väčší dataset všeobecne znamená presnejšie natrénovaný model bez ohľadu na použitý učiaci algoritmus. Vzorky v datasete môžu byť označené ako normálna prevádzka alebo konkrétny typ anomálie. Správne značkovanie datasetov je náročný proces, ktorý vyžaduje čas a skúsenosti bezpečnostných expertov. V praxi sa tak používajú neoznačkované, čiastočne označkované, a úplne označkované datasety podľa potreby danej úlohy. Práca sa podrobnejšie venuje dôležitosti datasetov a procesu trénovania v kap. 3.1 a 3.4.

Schopnosť detegovať popri známych útokoch aj ich modifikácie a úplne nové "zero-day" útoky je v súčasnosti dôležitou výhodou anomálnych metód. Na druhej strane, presnosť anomálnych metód je nižšia v porovnaní so signatúrnymi metódami, lebo sa orientujú na detekciu anomálií a nie priamo útokov. Keďže útoky sú len podmnožinou anomálií, často dochádza ku nesprávnemu vyhodnoteniu ľubovoľnej dynamickej zmeny v správaní používateľov alebo zariadení ako útoku. Následkom tohto správania generujú anomálne metódy časté falošné hlásenia (angl. *false positive*).

Anomálna metóda ponúka priestor pre možnosť voľby mnohých parametrov, ktoré môžu mať veľký vplyv na kvalitu jej výsledkov. Z tohto pohľadu sú známe dva prístupy:

štatistická analýza a **strojové učenie** (angl. *machine learning*; ML).

Štatistická analýza Metódy štatistickej analýzy vytvárajú stochastický model, určený rozdelením pravdepodobnosti [14]. Model predstavuje štatisticky popísané správanie normálnej prevádzky. Proces učenia hľadá rozdelenie pravdepodobnosti, ktoré najpresnejšie popisuje vzorky z použitého datasetu. Dataset obsahuje len vzorky normálnej prevádzky. Následná detekcia spočíva v spúšťaní testov s cieľom overiť, či vstupné vzorky aktuálnej sieťovej prevádzky patria do zvoleného rozdelenia pravdepodobnosti. Inými slovami, či pochádzajú štatistické charakteristiky vzoriek z platného rozsahu. V inom prípade, sú vzorky označené ako anomálie.

V procese hľadania vhodného rozdelenia existujú dva prístupy: **parametrický** a **bezparametrický**. Parametrické metódy začínajú prednastavením počiatočných podmienok a voľbou konkrétneho typu rozdelenia (napr. Gaussovo rozdelenie). Bezparametrické metódy nepoužívajú žiadne predpoklady. Z dôvodu absencie ľubovoľnej informácie o tvare modelu sa tieto metódy spoliehajú len na známe štatistické charakteristiky, napr. stredná hodnota, rozptyl, štandardná odchýlka a iné.

Výhodou prístupu štatistickej analýzy je jednoduchšia implementácia s vyššou presnosťou v porovnaní s metódami strojového učenia. Na druhej strane, metódy štatistickej analýzy nie sú schopné detegovať útoky, ktorých štatistické charakteristiky sú zhodné s charakteristikami normálnej prevádzky. Navyše, tieto metódy vyžadujú oveľa väčšiu množinu vzoriek a dlhší čas na tvorbu modelu s požadovanými výsledkami. Modely týchto metód konvergujú oveľa pomalšie ako v prípade metód strojového učenia. Pomalá konvergencia je dôvodom, prečo tento prístup nevyhovuje aplikáciám v súčasných vysokorýchlostných sieťach.

Strojové učenie Vysoká zložitosť súčasných sieťových systémov neumožňuje vytvoriť ich exaktný model. Metódy strojového učenia (ML) prinášajú možnosť vytvoriť približný model len na základe vstupných vzoriek bez znalosti interného správania systému. Všeobecná štruktúra modelu je daná vopred zvolenými hyper-parametrami, konkrétne správanie modelu je adaptované dynamicky v procese učenia. 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.

Každá metóda ML je definovaná svojim modelom, parametrami a chybovou funkciou. Proces trénovania upravuje parametre s cieľom minimalizovať celkovú chybu všetkých vzoriek, vyjadrenú prostredníctvom hodnoty chybovej funkcie. V oblasti ML sú známe dve

kategórie metód: **klasifikácia/regresia** (angl. *classification/regression*) a **zhlukovanie** (angl. *clustering*).

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) [15], umelé neurónové siete (angl. *Artificial Neural Networks*; ANN), a samo-organizujúce mapy (angl. *Self-Organizing Maps*; SOM) [16].

1.2.3 Vybrané klasifikátory založené na strojovom učení

V tejto kapitole uvádzame prehľad štyroch známych klasifikátorov založených na ML, ktoré dosahujú dobré výsledky v úlohách detekcie sieťových prienikov. Popis každej metódy obsahuje okrem teoretického popisu základného princípu aj prehľad vedeckých článkov, ktorých autori úspešne použili príslušnú metódu v oblasti detekcie sieťových prienikov.

Bayesovská sieť

Bayesovská sieť [17], [18] je klasifikátor založený na princípoch ML, ktorý používa riadené učenie. Model BN je vizuálne vyjadrený ako acyklický spojovo-orientovaný graf. Každý vrchol reprezentuje jeden z atribútov modelu (trieda, príznak), štatisticky považovaný za diskrétnu náhodnú premennú. BN podporuje len kategorické dáta.

Pre vyčíslenie miery vzájomnej závislosti medzi susednými vrcholmi, prepojenými orientovanou hranou, je použitá podmienená pravdepodobnosť. Princíp BN spočíva vo vyjadrení relácie medzi *apriórnymi* a *aposteriórnymi* udalosťami. Vychádzajúc z teórie pravdepodobnosti a Bayesovho pravidla (angl. *Bayes Rule*), sa pre odhad pravdepodobnosti každého vrcholu používa združená pravdepodobnosť (angl. *joint probability*) s odvolaním sa na pravdepodobnosti všetkých susedných vrcholov. Bayesovo pravidlo vyjadruje vzťah medzi pravdepodobnosťou výskytu apriórnej a aposteriórnej udalosti v (1.1),

$$\boldsymbol{P}\left(\boldsymbol{y}|\boldsymbol{X}\right) = \frac{\boldsymbol{P}\left(\boldsymbol{y} \wedge \boldsymbol{X}\right)}{\boldsymbol{P}\left(\boldsymbol{X}\right)} = \frac{\boldsymbol{P}\left(\boldsymbol{X}|\boldsymbol{y}\right) \times \boldsymbol{P}\left(\boldsymbol{y}\right)}{\sum_{j=1}^{k} \boldsymbol{P}\left(\boldsymbol{X}|\boldsymbol{y} = \boldsymbol{y_j}\right) \times \boldsymbol{P}\left(\boldsymbol{y} = \boldsymbol{y_j}\right)} \tag{1.1}$$

kde $\boldsymbol{y}$ je náhodná premenná výstupnej triedy a $\vec{\boldsymbol{X}}$ je vektor náhodných premenných, reprezentujúcich príznaky vstupnej vzorky. Z pohľadu modelu **skrytých Markovových reťazcov** (angl. *Hidden Markov Model*; HMM) je BN považovaná za jeho zovšeobecnenú verziu – uzly, reprezentujúce rôzne stavy HMM, sú v grafe BN modelované ako vrchol.

Metóda BN pracuje počas trénovania s označkovanými vzorkami datasetu, metóda postupne vyčísluje podmienené pravdepodobnosti vrcholov. Následná klasifikácia je vykonaná vo fáze inferencie, kedy sú nové vstupné vzorky (zobrané z testovacieho datasetu

alebo z reálneho prostredia) rozdelené do tried vzhľadom na ich hodnoty združenej pravdepodobnosti. Vzorke je pridelená finálna trieda, prislúchajúca najväčšej pravdepodobnosti.

BN je citlivá na dostupnosť **počiatočných znalostí** z oblasti riešeného problému. Z tohto dôvodu je možné zlepšiť výsledky metódy použitím dodatočných informácií o vstupoch, špecifických pre danú aplikačnú doménu, a skombinovať ich s novými dostupnými údajmi. V doméne detekcie prienikov sú napríklad použité informácie o rozložení rôznych tried sieťovej prevádzky v reálnej sieti. Zvyčajne v sieti dominuje len malá podmnožina typov prevádzky – normálna prevádzka a niektoré bežné útoky. Ich pravdepodobnosť je tak vyššia než v prípade ostatných typov. Oproti tomu, niektoré špeciálne útoky sú tak sporadické, že sa v reálnej sieti ani nemusia vôbec prejaviť. Použitím týchto rozšírujúcich informácií môže model upraviť svoje správanie tak, aby lepšie zodpovedal profilu reálneho sieťového prostredia ešte pred samotným trénovaním.

Nerovnomerné rozloženie rôznych útokov v reálnej sieti alebo v sieťových datasetoch má negatívny účinok na model BN a jeho rozhodovanie. Veľmi sporadické udalosti, ako sú špeciálne typy útokov, často nemajú žiadne zastúpenie v datasete z dôvodu jeho obmedzenej veľkosti. Triedy, patriace týmto útokom, vykazujú v modeli nulovú pravdepodobnosť, aj keď sa môžu objaviť v reálnej sieti. Následkom toho je neschopnosť detegovať útoky, ktoré prislúchajú triedam s nulovou pravdepodobnosťou. V praxi existujú viaceré metódy vyhladzovania, ktoré ponúkajú riešenie na spomenutý problém. Nenulová pravdepodobnosť pre všetky triedy je zaručená pridaním umelo vytvorených vzoriek do trénovacieho datasetu.

Detekcia sieťových prienikov založená na anomáliách je jednou z aplikačných domén, kde dosahujú metódy založené na BN priaznivé výsledky [19]–[22]. Všeobecne má model BN vysoké výpočtové nároky, ktoré bránia vo vykonaní klasifikácie v reálnom čase.

Odľahčenou verziou BN je **naivný Bayesov klasifikátor** (angl. *Naive Bayes*; NB), ktorý ponúka riešenie uvedeného problému s vysokými výpočtovými nárokmi a umožňuje použiť princípy BN aj v reálnom čase. Graf modelu NB pozostáva len z jedného koreňa (finálna trieda) a z množiny jeho priamych potomkov (jednotlivé príznaky). Hlavným predpokladom NB je **vzájomná nezávislosť** príznakov. Všetky hrany medzi potomkovými vrcholmi sú tak odstránené. Z toho vyplývajúca riedka štruktúra grafu značne zjednodušuje výpočet podmienených pravdepodobností pre jednotlivé triedy (1.2),

$$\boldsymbol{P}\left(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}|\boldsymbol{Y}\right) = \boldsymbol{P}\left(\boldsymbol{x_1}|\boldsymbol{Y}\right) \times \boldsymbol{P}\left(\boldsymbol{x_2}|\boldsymbol{Y}\right) \times \ldots \times \boldsymbol{P}\left(\boldsymbol{x_n}|\boldsymbol{Y}\right) \tag{1.2}$$

kde $\boldsymbol{Y}$ je náhodná premenná výstupnej triedy a $(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$ je vektor náhodných premenných pre príznaky vstupnej vzorky.

Experimenty analyzovaných článkov ukazujú, že metóda NB poskytuje dostatočne kvalitné výsledky aj v aplikáciach, kde predpoklad nezávislosti neplatí, napr. detekcia

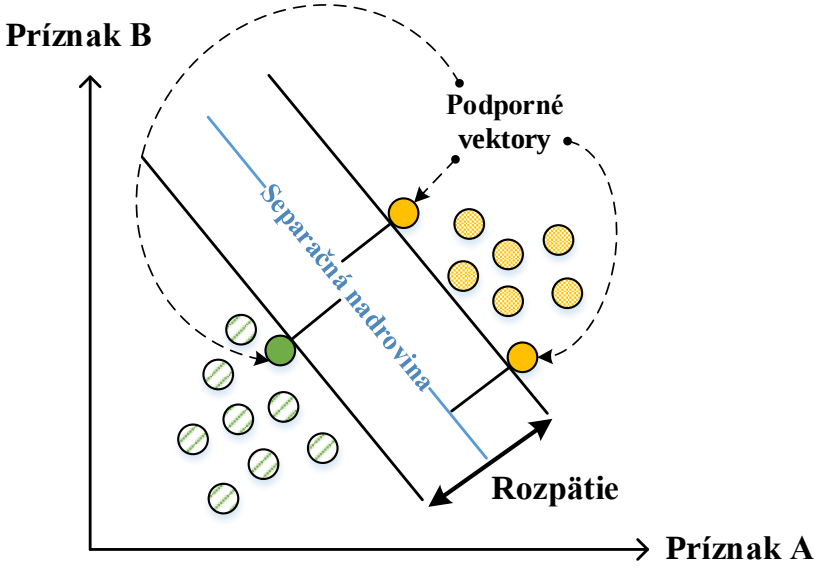
sieťových prienikov. Myšlienka detekcie prienikov a následnej klasifikácie je založená na výpočte pravdepodobnosti pre všetky možné triedy v závislosti od hodnôt príznakov v aktuálnej vstupnej vzorke. Ako finálnu značku vzorky zvolí metóda NB triedu s najväčšou pravdepodobnosťou. Týmto spôsobom je metóda schopná detegovať potenciálne sieťové útoky a taktiež určiť ich skutočné typy.

Prehľad výskumnej činnosti BN v oblasti NIDS Aplikáciou metód založených na BN v doméne detekcie prienikov sa zaoberajú viaceré články. Ich stručný prehľad je uvedený v tomto odstavci. Patel a Buddhadev sa v [19] 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 [21] použil Bayesové učenie, konkrétne NB klasifikátor, pre vytvorenie modelu reálnej sieťovej prevádzky s cieľom detegovať sieťové útoky DDoS. Autor poukazuje na vplyv nesprávne označkovaných vzoriek na kvalitu procesu trénovania, primárne pri tvorbe profilu normálnej prevádzky. Ako potenciálne základné príznaky boli zvolené polia z IP a TCP hlavičky: IP adresy, TCP porty a TCP flagy. Navrhovaná metóda taktiež používa techniku posuvných okien pre výpočet odvodených štatistických príznakov. 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 [22]. 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. Na získanie jej riešenia bol použitý genetický algoritmus (GA) nad vzorkami z datasetov *KDD99* [23] a *ISCX-2012* [24]. Článok zdôrazňuje zlepšenie celkových výsledkov a univerzálnosť zloženej metódy vďaka spolupráci rôznych prístupov (GA a NB). Alkasassbeh a spol. [20] porovnávajú modely NB, ANN a náhodného lesu (angl. *Random Forest*) nad úlohou detekcie DDoS útokov. Napriek tomu, že sa článok zameriava primárne na predstavenie nového datasetu, prezentované výsledky zdôrazňujú niektoré dôležité vlastnosti BN, napr. jej citlivosť na nevyrovnaný dataset.

Podporné vektory

Metóda podporných vektorov (SVM) je lineárny klasifikátor, uvedený Vapnikom v [15]. Metóda triedi označkované vzorky datasetu (vnímané ako body v N-rozmernom priestore) do dvoch skupín. Úlohou SVM je špecifikácia oddeľovacej nadroviny (angl.

discrimination hyper-plane), ktorá separuje body s ich maximálnou vzdialenosťou od samotnej nadroviny. Body z každej skupiny, umiestnené najbližšie k nadrovine (najťažšie rozlíšiteľné vzorky), sú označené ako podporné vektory (angl. *support vectors*). SVM predstavuje optimalizačnú techniku, ktorá maximalizuje rozpätie medzi podpornými vektormi rôznych tried. Príklad modelu SVM je ilustrovaný na obr. 1.1. Počet podporných vektorov do značnej miery ovplyvňuje náročnosť výpočtov.



Obr. 1.1: Vizuálna reprezentácia modelu podporných vektorov (SVM)

Ako už bolo spomenuté pri predchádzajúcej metóde BN, kvôli náročnému značkovaniu veľkých datasetov je ich súčasťou aj skupina nesprávne označených vzoriek. Tieto vzorky môžu značne sťažiť vyhľadávanie ideálnej nadroviny. Metóda SVM rieši problém nekorektných značiek zahodením alebo ignorovaním osamotených vzoriek (angl. outlier), t. j. vzoriek, ktorých príznaky sa zásadne líšia od väčšiny. SVM nazýva tieto vzorky *soft-margins*.

Ak nie sú vstupné body lineárne separovateľné, je ešte pred ich spracovaním použitá **nelineárna jadrová funkcia**. Jadrová funkcia transformuje prislúchajúce body z pôvodného priestoru do viac-rozmerného príznakového priestoru. Transformáciou vznikne nový nelineárny model, ktorého body sú už lineárne separovateľné. Navyše, aplikácia jadrovej funkcie vedie k lepšej separácii bodov a spolu s aplikáciou Lagrangeových multiplikátorov sú ďalšie výpočty jednoduchšie. Lagrangeove multiplikátory sú výsledkom procesu trénovania a určujú váhy trénovacích vzoriek – všetky trénovacie vzorky s výnimkou podporných vektorov majú nulové váhy. Preto je správanie klasifikátora určené len aktuálnymi hodnotami podporných vektorov.

Ako rozhodovacia funkcia pre klasifikáciu vzoriek vo fáze inferencie slúži marginálna funkcia (1.3),

$$\text{sign}\left(\left[\sum_{i=0}^{n} \boldsymbol{\alpha_i} \times \boldsymbol{y_i} \times (\boldsymbol{x_i} \times \boldsymbol{x})\right] + \boldsymbol{b}\right) \tag{1.3}$$

kde $[\boldsymbol{x_i}, \boldsymbol{y_i}]$ je i-ta trénovacia vzorka, $\boldsymbol{\alpha_i}$ je jej Lagrangeho multiplikátor, $\boldsymbol{b}$ je bias, a $\boldsymbol{x}$ je nová vzorka.

Variantom SVM s podporou neriadeného učenia je metóda **one-class SVM**, ktorá predpokladá len jednu triedu pre všetky vzorky v trénovacom datasete. Preto je hlavným cieľom tejto metódy nájsť lineárnu hranicu, ktorá oddeľuje všetky trénovacie vzorky od počiatku. Ostatné vlastnosti, princípy a kroky sú rovnaké ako v prípade pôvodnej metódy SVM s riadeným učením. Keďže metóda *one-class* SVM používa neriadené učenie, zdieľa všetky súvisiace výhody, napr. nevyžaduje označkovaný dataset. Vďaka tomu je možné trénovať model priamo v reálnom prostredí. Niektoré varianty SVM sú schopné odlíšiť viac ako dve triedy definovaním samostatnej oddeľovacej nadroviny pre každú triedu. Princíp sa pre tieto varianty nemení, len narastá ich výpočtová zložitosť. Všeobecnou výhodnou metód založených na SVM je ich presnosť a schopnosť pracovať s viac-rozmernými dátami. Na druhej strane, proces trénovania má v oboch variantoch SVM (s riadeným aj neriadeným učením) veľkú výpočtovú zložitosť.

Prehľad výskumnej činnosti SVM v oblasti NIDS Chandola a spol. [25] navrhli vizuálny model SVM regiónov, prislúchajúcich k profilom rôznych typov sieťovej prevádzky. Oddeľovacia nadrovina v tomto prípade reprezentuje hranice, ktoré oddeľujú rôzne vzory prevádzky. Patel a Buddhadev [19] poukazujú na využitie SVM v rôznych aplikačných doménach zvyčajne ako finálny klasifikátor v kombinácii s inými metódami strojového učenia. Osareh a Shadgar [26] porovnávajú modely SVM a ANN pri riešení úlohy detekcie sieťových prienikov. Článok vyzdvihuje dobrú schopnosť zovšeobecnenia (angl. *generalization*) modelu SVM a priebeh jeho implementácie v reálnom čase. Autori rovnako tak zdôrazňujú citlivosť SVM na podiel vzoriek normálnej prevádzky v použitom datasete *KDD99*. 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. [27] 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

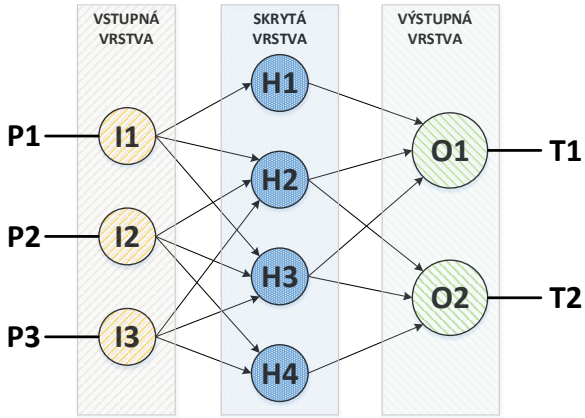
zdôrazňuje výhody hierarchickej hybridnej metódy v porovnaní s konvenčnými metódami, ktoré vedú k vyššej detekčnej presnosti, a k rýchlejšej fáze trénovania a testovania. Iný príklad hybridnej metódy je uvedený v [28]. Autori v uvedenom článku zdôrazňujú problém spracovania veľa-rozmerných vektorov, typických pre doménu detekcie sieťových prienikov. Erfani a spol. prichádzajú s navrhovanou metódou, ktorá používa hlboký model BN s neriadeným učením pre dynamický výber príznakov a one-class SVM model pre detekciu anomálií. Model hlbokej BN eliminuje nedostatky metódy SVM, ktoré sa viažu na redukciu veľa-rozmerného príznakového priestoru a extrakciu skrytých závislostí medzi príznakmi. Erfani a spol. popisujú dve konkrétne modifikácie modelu SVM: popis dát podporných vektorov (angl. *Support Vector Data Description*; SVDD) a rovinovo-založené one-class SVM (angl. *Plane-based One-class SVM*; PSVM). SVDD a PSVM sa líšia len v tvare oddeľovacej hranice. SVDD používa guľu s minimálnym polomerom; PSVM používa štandardnú nadrovinu. Autori porovnávajú navrhovanú metódu s viacerými odlišnými metódami (klasické aj hybridné), pričom navrhnutá metóda vykazuje vyššiu presnosť a menší čas trénovania/testovania. She a spol. [29] 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.

Umelá neurónová sieť

Umelá neurónová sieť (ANN) je všeobecne používaným systémom ML s mnohými existujúcimi variantmi. Podporuje riadené aj neriadené učenie a je použiteľná v rôznych aplikačných doménach. Inšpiráciou pre štruktúru modelu ANN je ľudský mozog a správanie sa jeho nervového systému. Model pozostáva z neurónov zoskupených do vrstiev. Spojenia medzi neurónmi určujú ich vzájomné vzťahy; spojenia závisia od konkrétneho typu ANN. Pôvodný model ANN používa tri základné typy vrstiev: **vstupná**, **výstupná**, a **skrytá** vrstva. Každý model ANN obsahuje vždy práve jednu vstupnú a jednu výstupnú vrstvu. Tieto vrstvy poskytujú rozhranie modelu pre komunikáciu s okolitým prostredím. Počet skrytých vrstiev je voliteľný. Ich správanie je nejasné, lebo sú tieto vrstvy zakódované do štruktúry ANN (váhy spojení a bias). Architektúra modelu je daná hyper-parametrami.

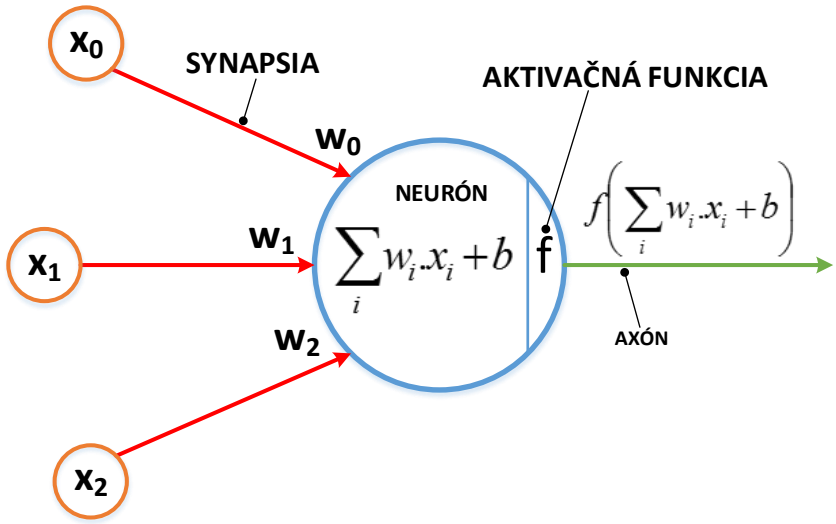
Vrstvy ANN sú umiestnené v riadku za sebou v špecifickom poradí. Každá vrstva obsahuje jeden alebo viacero neurónov, ktoré môžu byť pripojené len k neurónom susedných vrstiev (obr. 1.2). Podľa povoleného smeru spojení medzi vrstvami sú známe dva typy ANN: **dopredná** a **rekurentná** neurónová sieť.

Základným objektom modelu ANN je umelý neurón - **primitívna výpočtová jednotka**. Vstupy vchádzajú do neurónu cez **synapsie**. Každá synapsia má pridelenú **váhu**, ktorá ovplyvňuje intenzitu vstupného signálu. Celkový potenciál vstupných signálov je



Obr. 1.2: Vizuálna reprezentácia umelej neurónovej siete (ANN) s jednou skrytou vrstvou (Zdroj: http://cs231n.github.io). I1, I2, I3 reprezentujú neuróny vstupnej vrstvy – vstupné dáta majú tvar trojprvkových vektorov (P1, P2, P3). Neuróny, označené H1, H2, H3 a H4, tvoria skrytú vrstvu. O1, O2 sú neurónmi výstupnej vrstvy a reprezentujú triedy T1 a T2.

daný hodnotou **integračnej funkcie** – skalárny súčin. Integrovaný signál je následne upravený nelineárnou **aktivačnou funkciou**. Po spracovaní vstupných signálov posiela neurón finálny výsledok von ako výstupný signál k pripojeným susedným neurónom. Týmto spôsobom sa signál šíri celou sieťou. Obr. 1.3 popisuje abstraktnú štruktúru umelého neurónu.



Obr. 1.3: Štruktúra umelého neurónu. Na obrázku uvedené symboly $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}$ sú príznaky vstupných dát, $\boldsymbol{w_0}, \boldsymbol{w_1}, \boldsymbol{w_2}$ sú prislúchajúce váhy a $\boldsymbol{b}$ je bias vstupnej vzorky.

Počas trénovania sú vzorky z trénovacieho datasetu postupne vkladané do neurónovej siete so zámerom pozorovať reakciu siete (výstupy). V prípade chybného rozhodnutia sú váhy modelu automaticky upravené s cieľom zlepšiť budúce výstupy a spresniť rozhodovanie siete. Na vyčíslenie odchýlky od požadovaného výstupu slúži chybová funkcia (angl. *error function*), ktorá tak poskytuje pre trénovanie nevyhnutnú spätnú väzbu.

Výber chybovej funkcie priamo závisí od použitej aktivačnej funkcie. Dve najznámejšie funkcie sú kvadratická účelová funkcia (angl. *Mean Squared Error*; MSE) a krížová entropia (angl. *Cross-Entropy*). Učiaci algoritmus špecifikuje proces modifikácie váh. Celkové správanie ANN konverguje k požadovanému stavu použitím veľkého množstva trénovacích vzoriek a opakovanou úpravou váh počas trénovania. Medzi známe učiace algoritmy patria spätné šírenie chyby (angl. *Back-propagation*), Manhattanské aktualizačné pravidlo (angl. *Manhattan update rule*), rýchle šírenie (angl. *Quick propagation*), a odolné šírenie (angl. *Resilient propagation*), ktoré poukazujú na rôzne spôsoby úpravy váh. Časový interval procesu trénovania musí byť zvolený vhodne, aby nedošlo k pretrénovaniu modelu (angl. *over-fitting*). Tento stav sa prejavuje značným rozdielom medzi presnosťou vo fáze trénovania a vo fáze testovania. Príčinou okamžitej straty schopnosti zovšeobecňovať je skutočnosť, že si model do svojej štruktúry siete uložil všetky použité trénovacie vzorky. Trénovanie nastáva pri použití nedostatočne veľkého trénovacieho datasetu pre danú úlohu. Po skončení optimalizácie váh je model použitý pre spracovanie vstupných vzoriek z reálneho prostredia.

Vo všeobecnosti dokáže model ANN aproximovať ľubovoľnú funkciu alebo mapovanie. Výhodami ANN v porovnaní s ostatnými metódami ML sú: schopnosť extrahovať skryté závislosti vo vstupných dátach, a dobrá schopnosť zovšeobecnenia. Na druhej strane, kvalita správania modelu ANN priamoúmerne závisí od veľkosti dostupného trénovacieho datasetu. Správanie modelu ANN je skryté v štruktúre siete, preto nie je možné garantovať správne výsledky pre všetky možné vstupné vzorky. Aplikácia ANN tak vykazuje mieru neistoty v kritických doménach, ktoré nepripúšťajú žiadne chyby.

Prehľad výskumnej činnosti ANN v oblasti NIDS ANN predstavuje techniku ML, ktorá je vďaka vlastnostiam jej všeobecného modelu často používaná v rôznych oblastiach, napr. detekcia sieťových prienikov. Viaceré články potvrdzujú túto skutočnosť a prinášajú rôzne návrhy nových detekčných metód, založených na princípoch ANN. Alfantookh [30] predstavil jeden z prvých modelov ANN so spätným šírením chyby (BP-ANN) v úlohe detektora sieťových prienikov nad datasetom *DARPA98*. 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. Alfantookh vytvoril vo svojej práci niekoľko modelov s rôznymi hodnotami hyper-parametrov (metóda inicializácie váh, počet epóch, a rozmery skrytých vrstiev). Výkon tradičnej BP-ANN v úlohe detekcie sieťových prienikov je vyhodnotený v prácach [20], [26]. Osareh a Shadgar [26] porovnali model BP-ANN s modelom SVM nad datasetom *KDD99*. Podobným spôsobom, Alkasassbeh a spol. [20] porovnali model

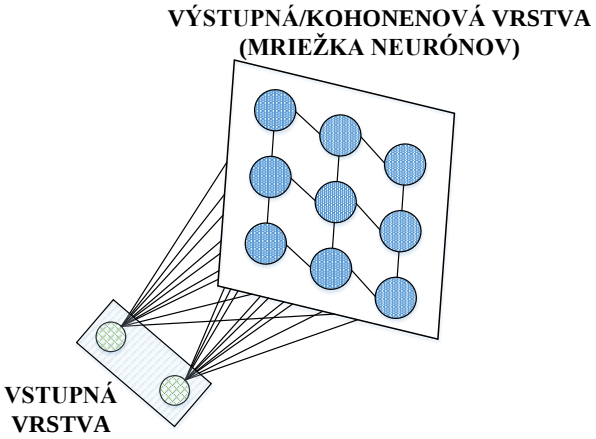
ANN s modelmi náhodného lesu a NB. Výsledky, publikované v oboch článkoch, poukazujú na jedinú výhodu ANN, vyššiu presnosť. Na druhej strane, ANN a jej varianty vyžadujú dlhší časový interval trénovania pre dosiahnutie dostatočných výsledkov. Patel a Buddhadev [19] uviedli popri tradičným typom ANN aj radič pre cerebelarný model artikulácie (angl. *Cerebellar Model Articulation Controller*) ako použiteľnú metódu detekcie DoS útokov. Vzhľadom na rovnakú grafickú reprezentáciu modelov BN a ANN, autori článku vyzdvihujú využitie skupiny jednoduchých špecializovaných modelov BP-ANN v hierarchickej hybridnej metóde. Článok [31] priniesol hybridnú signatúrnu metódu založenú na hierarchickej multi-agentovej štruktúre – každý agent, BP-ANN model, reprezentuje profil jedného alebo dvoch konkrétnych typov DoS útokov. Tang [32] popisuje model hlbokej ANN s vyladenými hyper-parametrami pre detekciu DDoS útokov. Metóda využíva výhody hlbokého učenia, ktoré umožňujú spájať skupiny jednoduchých príznakov do skrytých zložených príznakov. Garcia a Trinh [33] prišli s modelom ANN, natrénovaným pomocou algoritmu odolného šírenia. Článok porovnáva osem rôznych modelov a vyzdvihuje zvolený učiaci algoritmus, ktorý nevyžaduje výber hyper-parametrov vopred. Wei a spol. [34] zlepšili presnosť a rýchlosť konvergencie pôvodného modelu BP-ANN použitím špecifického optimalizačného algoritmu – *Particle Swarm Optimization* (PSO). Metóda LVQ, uvedená v [35], 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.

Samo-organizujúca mapa

Samo-organizujúca mapa (SOM), známa aj ako **Kohonenová sieť** [16], patrí medzi metódy ANN s plytkou štruktúrou. SOM používa neriadené učenie, aby rozdelila vstupné vzorky do zhlukov. SOM transformuje vzorky z veľa-rozmerného priestoru do jedno-/dvoj-/trojrozmerného priestoru – **topologickej mapy**. Napriek veľkej redukcii rozmerov dát je rozhodujúca informácia zachovaná.

Štruktúra modelu SOM pozostáva len z dvoch vrstiev. Výstupná vrstva obsahuje množinu prepojených neurónov, ktoré sú usporiadané do symetrického tvaru podľa rozmeru výstupnej hodnoty – reťaz, mriežka, a kocka (obr. 1.4). Počas trénovania sa model učí nové vzory, ktoré nájde v trénovacích vzorkách. Podobné vzory sú zoskupené do zhlukov. Segmentácia vzorov do zhlukov zachytáva rôzne vzťahy medzi príznakmi vstupných dát.

V prvom kroku spracovania vstupnej vzorky, model SOM nájde neurón topologickej mapy, ktorý sa najviac podobá vstupnej vzorke – najbližší neurón. Pre vyčíslenie vzdialenosti medzi vektorom vstupnej vzorky a váhovými vektormi jednotlivých neurónov je možné použiť ľubovoľnú viacrozmernú funkciu. Medzi najpoužívanejšie patria *Euklidová*



Obr. 1.4: Vizuálna reprezentácia Kohonenovej siete (SOM)

a *Manhattanská* vzdialenosť. V druhom kroku, model upraví váhové vektory výherného neurónu (angl. *Best Match Unit*; BMU) a všetkých jeho susedných neurónov s cieľom posilniť ich vzájomnú podobnosť. Miera zmeny váh klesá s narastajúcou vzdialenosťou neurónov od BMU. Po vykonaní predchádzajúcich krokov pre všetky trénovacie vzorky obsahuje topologická mapa niekoľko zhlukov. Ich počet je daný počtom rôznych vzorov, dostupných v trénovacom datasete. Vzory sa v topologickej mape prejavujú ako vznikajúce regióny. Ďalšia vizualizácia vyžaduje priemet váhových vektorov do 3D grafu. V praxi sa používajú tri vizualizačné techniky, ktoré sa odlišujú v spôsobe stanovenia intenzity pixelov: *U-Matrix* (vzdialenosť vektora k jeho susedom), *P-Matrix* (hustota okolia vektora), a *U*-Matrix* (kombinácia predchádzajúcich dvoch techník).

Model SOM podporuje aj čiastočne riadené učenie (kap. 3.4), ktoré prináša určitú nezávislosť modelu od dostupných datasetov. Prostredníctvom modelu SOM sú dosahované veľmi dobré výsledky v úlohách zhlukovania, segmentácie a vizualizácie. Na druhej strane, metóda SOM nájde len povrchné vlastnosti vstupných dát bez skrytých detailnejších informácií a nie je schopná klasifikovať vstupné vzorky. Vzhľadom na uvedené vlastnosti je SOM v praxi často používaná v kombinácií s inými metódami ML, ktoré pomáhajú zlepšiť jej výsledky. Preto existujú rôzne modifikácie metódy SOM, ktoré riešia nedostatky pôvodnej metódy, napr. Emergent SOM, HSOM, GHSOM, a Distributed SOM.

Prehľad výskumnej činnosti SOM v oblasti NIDS Metóda SOM má bohatú históriu v implementácii NIDS, kde dosahuje najlepšie výsledky spomedzi všetkých tradičných metód ML. V tejto časti sú spomenuté niektoré vedecké články, ktoré sa venujú využitiu SOM v úlohe detektora sieťových prienikov. Patel a Buddhadev [19] považujú neriadené učenie v metóde SOM ako podstatnú výhodu vzhľadom na zložitosť, chybovosť a časovú náročnosť procesu značkovania sieťových vzoriek. Autori zdôrazňujú vhodnosť

skombinovať SOM s inými metódami kvôli už poznamenanej neschopnosti SOM klasifikovať dáta. Článok [36] 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. Zväčšenie rozmerov topologickej mapy vedie k vyššiemu rozlíšeniu prislúchajúceho 3D grafu. Pan a Li [37] popisujú hybridný model vylepšenej verzie SOM a radiálnej bázovej funkcie (angl. *Radial Basis Function*; RBF). Spolupráca uvedených metód je viditeľná priamo v štruktúre výsledného modelu; model SOM nastavuje parametre RBF metódou neriadeného učenia namiesto menej presných tradičných zhlukovacích algoritmov ako je *k-means*. Vylepšená verzia SOM – Mriežka deliacich buniek (angl. *Cell Splitting Grid*; CSG) – dynamicky hľadá vhodné centrá pre neuróny RBF. Podľa výsledkov troch odlišných modelov vykazuje kombinácia RBF a CSG najrýchlejší tréning a najlepšiu presnosť. Výhody metódy SOM, samoučenie a detekcia nových útokov, sa v [38] 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. Všetky tri modely majú rovnaké rozmery topologickej mapy, odlišujú sa len vo výbere príznakov podľa zvolenej úrovne. Jiang a Yang [39] prichádzajú s viacerými zlepšeniami. Ich navrhnuté úpravy trénovania modelu SOM obmedzujú negatívne následky kompetitívneho učenia – existencia úplne neaktívnych neurónov v topologickej mape. Neaktívne neuróny zodpovedajú veľmi sporadickým udalostiam (občasné typy útokov), ktoré sú v trénovacom datasete zastúpené len veľmi malým počtom vzoriek. Tieto neuróny nikdy nevyhrajú, a tak sa časom postupne izolujú od ostatných. Osamotené neuróny majú negatívny účinok na presnosť detekcie prislúchajúcich občasných útokov. Riešením je náhrada pôvodného algoritmu úpravy váh jeho modifikovanou verziou, ktorá berie do úvahy aj vzťahy medzi susednými neurónmi a zaručuje aktualizáciu všetkých neurónov. Problému neaktívnych neurónov, konkrétne ich existencii a vplyvu na zvýšenú výpočtovú zložitosť, sa venuje aj článok [40], ktorý porovnáva rôzne existujúce varianty SOM. Choksi a Shah implementovali modely dvoch pokročilých metód SOM v úlohe IDS: Hierarchická SOM (HSOM) a Rastúca hierarchická SOM (angl. *Growing Hierarchical SOM*; GHSOM). Obe metódy umožňujú dynamický rast modelu (zmena rozmerov topologickej mapy) počas trénovania v závislosti od aktuálnych vstupných vzoriek. Vďaka tomu je zamedzený vznik izolovaných neurónov v topologickej mape a zvýšený celkový výkon metódy. Kim a Park [41] 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. Keďže sú z dôvodu odlišného umiestnenia modelov použité rôzne trénovacie vzorky, modely následne reagujú na tú istú vzorku rôzne. Narastajúca odlišnosť medzi topologickými mapami modelov a ich správaním je vyriešená ich pravidelnou synchronizáciou. Topologické mapy modelov sú najprv zlúčené a následne nahradené novou spoločnou mapou. 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.

1.2.4 Ohodnotenie kvality metód detekcie sieťových prienikov

Vzhľadom na veľký počet rôznych detekčných metód je nevyhnutné vykonať ich vzájomné porovnanie a zvoliť najvhodnejšiu z nich. Pre tento účel slúžia viaceré metriky a numerické ukazovatele, ktoré exaktne ohodnotia výkonnosť, kvalitu a efektivitu zvolenej metódy [7], [9], [10], [42]. Ukazovatele sú vyjadrené pomocou štyroch základných premenných, ktoré zaznamenávajú všetky možné situácie podľa rozhodnutia metódy a jeho správnosti: *True Negative* (TN), *True Positive* (TP), *False Negative* (FN), a *False Positive* (FP). Tieto počítadlá sú numericky vyjadrené cez chybovú maticu (angl. *error matrix*).

Presnosť (angl. *accuracy*) (1.4) a **miera chybnej klasifikácie** (angl. *misclassification*) (1.5) berú do úvahy všetky štyri premenné. Presnosť je miera, s akou je metóda schopná správne vyhodnotiť vstupnú vzorku. Presnosť je definovaná ako pomer počtu správnych rozhodnutí ku počtu všetkých rozhodnutí metódy. Miera chybnej klasifikácie je doplnok ku presnosti, reprezentuje mieru nesprávnych rozhodnutí.

$$\text{Presnosť} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1.4}$$

$$\text{Chyba} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \mathbf{1} - \text{Presnosť} \tag{1.5}$$

Citlivosť (angl. *sensitivity*) (1.6) odhaľuje reakciu metódy na výskyt anomálie. Citlivosť je definovaná ako pomer počtu správne označených anomálií ku počtu všetkých anomálnych vzoriek. **Špecifickosť** (angl. *specificity*) (1.7) je podobná, ale zachytáva reakciu metódy na vzorky normálnej prevádzky.

$$\text{Citlivosť} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.6}$$

$$\text{Špecifickosť} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{1.7}$$

Miera FP (angl. *False Positive Rate*; FPR) (1.8) a **miera FN** (angl. *False Negative*

Rate; FNR) (1.9) odzrkadľujú nesprávne správanie metódy. FPR je pomer počtu chybných alarmov ku všetkým alarmom. Vysoká hodnota FPR degraduje dôveryhodnosť metódy. V kritických oblastiach, ako je detekcia sieťových prienikov, je preto rozhodovanie podľa výstupov metódy s vysokou mierou FP riskantné a vyvoláva zvýšené náklady, spôsobené opakovaným vykonávaním opatrení na riešenie falošných alarmov. Na druhej strane, FNR je pomer neodhalených anomálií ku všetkým vzorkám, ktoré metóda zaradila do normálnej prevádzke.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} \tag{1.8}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} \tag{1.9}$$

Vyjadrením hodnoty vymenovaných ukazovateľov je možné rozhodnúť o prevahe jednej metódy nad ostatnými. Tieto hodnoty ale nepokrývajú všetky vlastnosti metód a v čase sa dynamicky menia. Výstupy metódy závisia aj od iných parametrov, ktoré do značnej miery ovplyvňujú kvalitu jej správania, napr. voľba datasetov a umiestnenie v sieťovej infraštruktúre. Okrem správnosti rozhodovania metódy je potrebné zvážiť aj iné z jej vlastností ako sú výpočtová zložitosť, oneskorenie, a škálovateľnosť.

Problematika ohodnotenia NIDS V praxi je dostupných veľa metód pre modelovanie a implementáciu NIDS (viď. predchádzajúce časti práce). Jasné, jednoznačné ohodnotenie týchto metód čelí viacerým otázkam, ktoré súvisia najmä s ich vzájomnou odlišnosťou. Príkladom sú rôzne prístupy implementácie (simulácia, emulácia, a reálne nasadenie) a rôzne účely metód vzhľadom na ich kategorizáciu. Druhý z uvedených príkladov poukazuje na dôležitosť výberu vhodného datasetu. Datasety použité pre ohodnotenie aktuálnych detekčných metód sú odlišné a často zastaralé. Navyše, pomer vzoriek normálnej prevádzky ku anomálnym vzorkám je v týchto datasetoch otázny vzhľadom na skutočnosť, že detekčné metódy vyžadujú zvyčajne vyvážené datasety pre natrénovanie všetkých typov prevádzky za rovnakých podmienok. Súčasné, verejne dostupné datasety (*KDD99* [23], *CAIDA-DDOS-2007*, *DARPA-DDOS-2009*, *TUIDS-2012*, *BOOTER-DNS-2014*) trpia týmito nedostatkami, nevhodne ovplyvňujú výsledky testovania, a nezodpovedajú správaniu súčasnej reálnej sieťovej infraštruktúre. Správne vyhodnotenie výstupov detekčnej metódy vyžaduje zváženie viacerých otázok, ktoré prinášajú dodatočné požiadavky na techniky testovania:

- **Spoľahlivosť** - Test dáva relevantné informácie bez ohľadu na typ testovanej metódy.

- **Reprodukovateľnosť** - Výskumníci sú schopní opakovať experimenty za rovnakých podmienok a s rovnakými výstupmi.
- **Flexibilita** - Test umožňuje meniť svoje parametre pre dosiahnutie určitej úrovne variability.
- **Škálovateľnosť** - Test je použiteľný v reálnom prostredí, napr. v prostredí rozľahlej siete s veľkým objemom sieťovej prevádzky.
- **Prezentácia výsledkov** - Test poskytuje výstupy v jasnej, zrozumiteľnej forme. Vizualizácia hrá dôležitú úlohu pri analýze sieťovej prevádzky.

Trendom v testovaní NIDS je použitie dát, extrahovaných z reálnej sieťovej prevádzky so zámerom čo najvernejšie napodobniť správanie moderných počítačových sietí.

1.2.5 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** - Trénovanie nad vzorkami reálnej sieťovej prevádzky rieši záležitosti, ktoré úzko súvisia s vlastnosťami dostupných sieťových datasetov, popísaných v predchádzajúcej podkapitole.

- **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. Do tejto online aktualizácie je okrem zmeny parametrov modelu zahrnutá aj dynamická rekonfigurácia jeho štruktúry.

Aplikáciou vymenovaných trendov na návrh detekčnej metódy očakávame zlepšenie celkovej výkonnosti a vyriešenie viacerých problémov v súčasných NIDS.

1.3 Konvolučná neurónová sieť

Konvolučná neurónová sieť [43]–[46] vychádza z klasickej doprednej umelej neurónovej siete a preto preberá väčšinu jej základných princípov štruktúry, trénovania aj inferencie. Jej stavebnými prvkami sú umelé neuróny, ktoré sú usporiadané do vrstiev. Neuróny susedných vrstiev sú vzájomne prepojené synapsiami. Ich prostredníctvom sa šíria signály medzi neurónmi. Vzhľadom na vyššie uvedený dopredný charakter konvolučnej siete sú vrstvy usporiadané sekvenčne za sebou, a tak aj spracovanie vstupných dát prebieha v tomto smere.

Motiváciou pre aplikáciu konvolučnej siete bol veľký rozmer vstupných dát v oblasti spracovania obrazu [47], [48]. Keďže každý pixel vstupného obrazu predstavoval samostatný príznak, počet vstupných príznakov sa ďalej postupne prejavil aj na ďalších skrytých vrstvách a sieť tak vyžadovala natrénovať veľké množstvo voľných parametrov (parametre upravované niektorou z adaptívnych metód počas trénovania). Tento efekt spôsobený narastajúcim počtom vstupných príznakov predstavoval problém pri návrhu siete, jej trénovaní a následnej implementácii na báze efektívnych technických prostriedkov.

Ak sieť disponuje veľkým množstvom voľných parametrov (v prípade klasickej umelej neurónovej siete všeobecne platí, že počet voľných parametrov je väčší než počet synapsií medzi neurónmi), má sieť veľkú pamäťovú kapacitu [49], teda dokáže si presne zapamätať veľké množstvo rôznych vzoriek. Ak je v takom prípade použitá malá sada trénovacích vzoriek, sieť sa “nabifľuje” tieto vzorky bez pochopenia samotnej podstaty úlohy. Inak

povedané, sieť stráca schopnosť generalizácie – schopnosť siete klasifikovať aj vzorky, ktoré nie sú súčasťou trénovacej sady. Tento stav siete označujeme ako pretrénovanie (overfitting) a prejavuje sa značnými rozdielmi medzi chybou počas trénovania a chybou počas testovania. Riešením pretrénovania je:

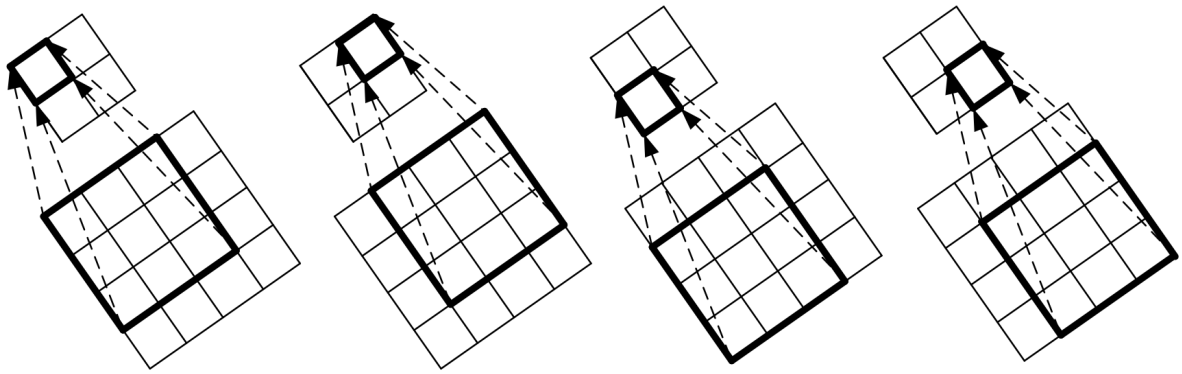
- zväčšiť veľkosť trénovacej sady
- zmenšiť počet voľných parametrov

Úlohy z oblasti spracovania obrazu si vynútili z dôvodu možného pretrénovania veľké množstvo trénovacích vzoriek. Konvolučná sieť znižuje riziko pretrénovania znížením počtu voľných parametrov priamo v návrhu štruktúry siete. K tomuto účelu využíva všeobecne známe fakty z oblasti rozpoznávania vzorov v obraze: priestorová nemennosť (vzor sa môže vyskytovať v ľubovoľnom mieste vstupného obrazu) a hierarchická štruktúra vzorov (zložitejšie objekty pozostávajú z viacerých jednoduchších objektov).

Konvolučná sieť je postavená na nasledujúcich troch princípoch: zdieľanie váh, lokálne receptívne pole (angl. *receptive field*) a podvzorkovanie [49]. Zdieľanie váh spočíva v doplnení obmedzujúcich podmienok pre návrh synapsií medzi neurónmi – synapsie v rámci skupiny neurónov musia použiť rovnaké váhy. Keďže mnohé synapsie zdieľajú rovnaké váhy, následkom tohto 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*). Receptívne pole pre konkrétny neurón zodpovedajúci bodu na výstupnej príznakovej mapy predstavuje skupinu susedných neurónov umiestnených v malom okolí na vstupnej mape. Z pohľadu výpočtov zodpovedá receptívne pole oknu, ktoré je použité na výpočet prislúchajúceho bodu výstupnej mapy. Keďže všetky okná jednej vstupnej mapy používajú rovnaké váhy, teda vykonávajú rovnakú operáciu ale nad rôznymi jej časťami, môžeme si celý proces výpočtu predstaviť ako skenovanie vstupnej mapy metódou posuvného okna s cieľom nájsť všetky výskyty lokálneho príznaku. Obrázok 1.5 ilustruje metódu použitia posuvného okna na vstupnú mapu.

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. Z toho vyplýva, že všetky vstupné mapy, ktoré získame permutáciou (preusporiadaním) pixelov v rámci jedného okna, budú prislúchať jednej a tej istej výstupnej mape. Použitím tejto myšlienky na oblasť spracovania obrazov dosiahneme operáciou zlučovania to, že všetky lokálne rozdiely v natočení konkrétneho vzoru a jeho skreslenie budú namapované na jednu spoločnú mapu.



Obr. 1.5: Skenovanie mapy pomocou filtra – posuvné okno o veľkosti **3 × 3** (silne zvýraznené políčka) postupne prechádza vstupnou mapou (skupina políčok tvoriacich štvorec s rozmermi **4 × 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 **2 × 2** umiestnená vyššie).

Z pohľadu všeobecných úloh, ktoré rieši oblasť rozpoznávania vzorov, je konvolučná sieť schopná s použitím adaptívnych metód súčasne vykonať extrakciu príznakov aj následnú klasifikáciu [49]. Navyše, každú z týchto úloh rieši samostatná časť siete, ktorá je v modeli jasne identifikovateľná. Keďže mnohé úlohy sú veľmi podobné, môžeme vybrať zo siete naučenej nad konkrétnou úlohou len časť zodpovednú za extrakciu príznakov a vložiť ju následne do modelu zodpovedného za riešenie inej, podobnej úlohy.

Zatiaľ čo klasická neurónová sieť pracovala prevažne s jednorozmernými dátami, konvolučná sieť je schopná pracovať aj s viacrozmernými dátami, najčastejšie ide o trojrozmerné polia (šírka, výška a hĺbka). Prvé dva rozmery určujú rozmery príznakových máp. Tretí rozmer udáva ich počet na príslušnej vrstve, ktorý všeobecne označujeme ako počet kanálov. Konvolučná sieť zohľadňuje hodnoty vstupných príznakov ako aj ich rozloženie – **topológiu**, t. j. výstup z vrstvy nezávisí len od hodnôt vstupných príznakov ale aj od ich poradia, napr. umiestnenie pixelov vo vstupných obrazoch alebo poradie prvkov v časových radoch.

1.3.1 Typy vrstiev

Na rozdiel od klasickej umelej neurónovej siete, ktorá používa štrukturálne identické vrstvy so synapsiami medzi každou dvojicou neurónov susedných vrstiev, konvolučná sieť preukázala praktický význam a prínos v použití rôznych druhov špecializovaných vrstiev. Typy vrstiev vychádzajú z pôvodného zámeru použiť neurónovú sieť nielen ako klasifikátor ale súčasne aj ako extraktor príznakov. Nasadenie každej z vrstiev mení správanie siete iným spôsobom. V praxi sú zaužívané striedajúce sekvencie konvolučných a zlučovacích vrstiev v prvej fáze (extrakcia príznakov) a sekvencie plne-prepojených vrstiev v druhej

fáze (klasifikácia) siete. Aktivačná vrstva nie je spravidla uvádzaná v popise architektúry, ale býva umiestnená takmer za každou konvolučnou a plne-prepojenou vrstvou. Správanie vrstiev je definované ich typom, nastavením hyperparametrov a natrénovaním voľných parametrov. Hyperparameter vrstvy je oproti voľnému parametru nastavený vopred a počas trénovania sa už nemení.

Konvolučná vrstva

Konvolučná vrstva vychádza z princípov uvedených vyššie a vykonáva výpočtovo náročnú, kľúčovú operáciu konvolučnej siete [50]. Základom je priestorová, resp. časová súvislosť medzi susednými vstupnými príznakmi (lokálna korelácia medzi susednými pixelmi vo vstupných obrazoch). Úlohou konvolučnej vrstvy je extrakcia rôznych príznakov zo vstupnej príznakovej mapy.

$$\boldsymbol{Y}^{\boldsymbol{l}}_{\boldsymbol{i,j}} = \boldsymbol{b}^{\boldsymbol{l}} + \sum_{\boldsymbol{h=1}}^{\boldsymbol{H}} \sum_{\boldsymbol{m=1}}^{\boldsymbol{K}} \sum_{\boldsymbol{n=1}}^{\boldsymbol{K}} \boldsymbol{X}^{\boldsymbol{h}}_{\boldsymbol{i+m,j+n}} \times \boldsymbol{W}^{\boldsymbol{h}}_{\boldsymbol{m,n}} \tag{1.10}$$

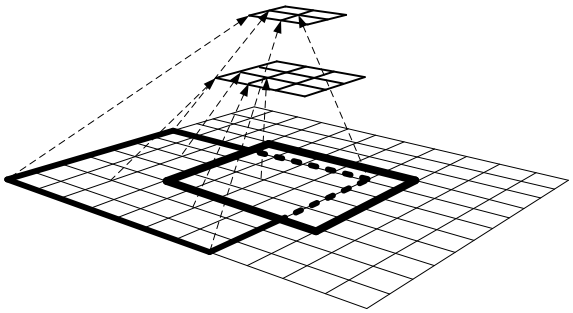
Matematický výraz uvedený v (1.10) určuje spôsob spracovania konvolučnej vrstvy, kde $\boldsymbol{X}^{\boldsymbol{h}}_{\boldsymbol{i+m,j+n}}$ je bod na pozícii $\boldsymbol{(i+m, j+n)}$ v $\boldsymbol{h}$-tej vstupnej mape, podobne $\boldsymbol{Y}^{\boldsymbol{l}}_{\boldsymbol{i,j}}$ je bod na pozícii $\boldsymbol{(i, j)}$ v $\boldsymbol{l}$-tej výstupnej mape, $\boldsymbol{W}^{\boldsymbol{h}}_{\boldsymbol{m,n}}$ je koeficient na pozícii $\boldsymbol{(m, n)}$ v kerneli s rozmermi $\boldsymbol{K \times K \times H}$ použitom pre $\boldsymbol{h}$-tú vstupnú mapu a $\boldsymbol{b}^{\boldsymbol{l}}$ je bias pre $\boldsymbol{l}$-tú výstupnú mapu. Spôsob spracovania vstupných máp konvolučnou vrstvou je tak porovnateľný s 3D konvolúciou známou pri aplikácii filtrov v oblasti spracovania signálov.

Myšlienka použitia viacrozmerných filtrov na obraz bola známa už pred vznikom konvolučnej vrstvy, napr. ako súčasť manuálnej extrakcie príznakov. Ale tieto klasické filtre používali vopred zvolené a pevne nastavené koeficienty, ktoré boli zamerané na detekciu konkrétnych vzorov, napr. Sobelov filter. Nevýhodou takýchto filtrov je ich vhodnosť len pre konkrétnu podmnožinu úloh. Prínos konvolučnej vrstvy spočíva v nahradení pevných koeficientov vo filtroch za voľné parametre, ktoré môžeme upravovať rovnako ako ľubovoľné iné parametre počas trénovania a tak automaticky prispôsobiť filter presne danej úlohe.

Koeficienty filtra v prípade konvolučnej vrstvy tvoria kernel. Keďže kernel slúži na detekciu konkrétneho vzoru (konkrétny vzor určený použitými váhami), môžeme sa na príznakovú mapu pozerať ako na mapu výskytov daného vzoru na rôznych miestach vstupnej mapy. Počet výstupných máp na vrstve určuje počet rôznych príznakov, ktoré je vrstva schopná extrahovať. Postupným prechodom filtra (skupina kernelov určená svojou šírkou, výškou a hĺbkou) cez vstupnú sadu máp získame výstupné mapy.

Veľkosť filtra obmedzuje rozmery príznaku, ktorý môžeme na danej vrstve extrahovať. Preto pre extrakciu zložitejších príznakov, ktoré sa vo vstupnom obrázku rozprestierajú na

väčšom počte pixelov (teda receptívne pole pre zaznamenanie príznaku je väčšie), musíme do architektúry konvolučnej siete pridať viacero vrstiev za sebou. Postupným vrstvením dosiahneme schopnosť siete extrahovať príznaky rôznej zložitosti od najjednoduchších (hrany a rohy extrahované na prvých vrstvách) až po zložité (tváre, osoby, dopravné prostriedky a iné objekty extrahované na posledných vrstvách). Tento princíp rozširovania efektívnej šírky receptívneho poľa je vizuálne znázornený na obr. 1.6 ako pyramída.



Obr. 1.6: Rozširovanie receptívneho poľa vrstvením – efektívna veľkosť receptívneho poľa (zobrazená na spodnej mape) narastá postupným pridávaním ďalších vrstiev uvedených v smere zdoľa nahor.

Konvolučná vrstva podporuje nasledujúce hyperparametre:

- rozmery filtrov - šírka a výška sú zvyčajne rovnaké, štandardne malé nepárne čísla, napr. $\mathbf{3 \times 3}$ a $\mathbf{5 \times 5}$ pixelov (v práci označené $\boldsymbol{K}$),
- počet výstupných máp (v práci označené $\boldsymbol{H}$),
- šírka kroku (stride) - určuje medzeru medzi susednými oknami (v práci označené $\boldsymbol{S}$),
- výplň okrajov (padding) - určuje počet bodov umelo doplnených po každom z okrajov vstupnej mapy, slúži na reguláciu rozmerov výstupnej mapy (v práci označené $\boldsymbol{P}$),
- zväčšenie vzdialenosti (dilation) medzi bodmi vstupnej mapy, ktoré tvoria spoločné okno.

Vychádzajúc zo vzťahu (1.10), konvolučná vrstva používa nasledujúce parametre:

- váhy kernelov ($\boldsymbol{W^l_{m,n}}$, kde usporiadana dvojica $\boldsymbol{(m, n)}$ je pozícia váhy v kerneli, $\boldsymbol{h}$ je index vstupnej mapy, a $\boldsymbol{l}$ je index výstupnej mapy),
- bias výstupných máp ($\boldsymbol{b^l}$ - bias pre $\boldsymbol{l}$-tú výstupnú mapu).

Rozmery výstupnej mapy závisia od viacerých hyperparametrov: šírka a výška vstupnej mapy, šírka kroku a výplne. Ak predpokladáme rovnakú šírku a výšku vstupnej mapy ($\boldsymbol{N}_{\text{in}}$), rovnakú šírku a výšku kernelov ($\boldsymbol{K}$), potom môžeme vyjadriť rozmery výstupnej mapy ($\boldsymbol{N}_{\text{out}}$) cez (1.11).

$$\boldsymbol{N}_{\text{out}} = \frac{\boldsymbol{N}_{\text{in}} - \boldsymbol{K} + \mathbf{2} \times \boldsymbol{P}}{\boldsymbol{S}} + \mathbf{1} \tag{1.11}$$

Unikátnym prípadom konvolučnej vrstvy je použitie kernelu so šírkou $\mathbf{1} \times \mathbf{1}$ pixelov, ktorý bol prvýkrát uvedený v [51]. Na prvý pohľad je aplikácia jednotkového kernelu zbytočná, lebo predstavuje len škálovanie bodov vstupnej mapy. Jej prínos ale spočíva v redukcii tretieho rozmeru, teda v znížení počtu príznakových máp. V porovnaní so zlučovacou vrstvou má takto navrhnutá konvolučná vrstva rovnaký efekt redukcie len nad iným rozmerom.

Zlučovacia vrstva

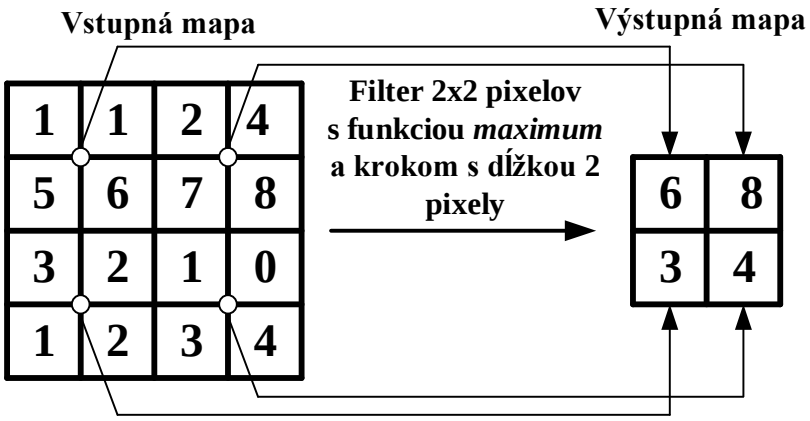
Zlučovacia vrstva (angl. *pooling*) vykonáva operáciu zlučovania (1.12). Táto operácia je vo svojej podstate rovnaká ako v prípade konvolučnej vrstvy. Rozdiel spočíva len vo funkcii, ktorú použijeme nad skupinou bodov v lokálnom okolí (v okne). V prípade zlučovacej vrstvy sú najpoužívanejšie funkcie: priemer a maximum. Okrem posilnenia generalizácie siete, teda potlačenia jej pretrénovania, má zlučovanie aj ďalšiu výhodu vo forme podstatnej redukcie rozmerov máp. Aplikáciou zlučovania je značná časť informácie odstránená bez straty schopnosti siete úspešne klasifikovať vstupný obraz. Naopak, redukciou informácií je odstránený šum v mapách, čím sa zlepšia celkové výsledky. Zlučovanie vedie ku zmenšeniu rozmerov príznakových máp na ďalších vrstvách, zníženiu počtu synapsií a voľných parametrov, a následne aj k zníženiu pamäťových a výpočtových nárokov. Príklad zlučovania s použitím maxima je zobrazený na obrázku 1.7.

$$\boldsymbol{Y}^{\boldsymbol{l}}_{\boldsymbol{i},\boldsymbol{j}} = \boldsymbol{f}(\boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i},\boldsymbol{j}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i}+\mathbf{1},\boldsymbol{j}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i},\boldsymbol{j}+\mathbf{1}}, \boldsymbol{X}^{\boldsymbol{l}}_{\boldsymbol{i}+\mathbf{1},\boldsymbol{j}+\mathbf{1}}) \tag{1.12}$$

Vzhľadom na principiálnu zhodu medzi konvolučnou a zlučovacou vrstvou majú tieto vrstvy zhodné aj niektoré hyperparametre:

- rozmery okna (štandardne $\mathbf{2} \times \mathbf{2}$),
- šírka kroku (štandardne zhodná s rozmermi okna, t. j. bez prekrývania susedných okien).

Zlučovacia vrstva nedisponuje štandardne žiadnymi voľnými parametrami, teda trénovaním sa jej správanie nemení.



Obr. 1.7: Zlučovanie aplikáciou filtra $\mathbf{2 \times 2}$ pixelov s použitím funkcie *maximum* a s dĺžkou kroku 2 pixely

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 (1.13)

$$\boldsymbol{Y_i} = \vec{\boldsymbol{X}} \cdot \vec{\boldsymbol{W}}_{\boldsymbol{i}}^{\mathsf{T}} + \boldsymbol{b_i} \tag{1.13}$$

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. Keďže numericky sa operácie použité v konvolučnej a plne-prepojenej vrstve zhodujú, môžeme chápať plne-prepojenú vrstvu ako konvolučnú vrstvu s filtrom o šírke 1 a výške rovnej počtu neurónov na tejto vrstve. Plne-prepojená vrstva slúži ako klasifikátor, kde vstupný vektor reprezentuje vektor zložitých príznakov (extrahovaných v predchádzajúcich vrstvách).
Plne-prepojená vrstva používa jediný hyperparameter - počet neurónov na vrstve, a voľné parametre:

- váhy synapsií (v práci označené ako $\mathbf{W}$),
- bias (v práci označené ako $\boldsymbol{b_i}$)

Aktivačná vrstva

Operácie konvolučnej a plne-prepojenej vrstvy nad vstupnými mapami môžeme vyjadriť ako súčin matíc, ide teda o **lineárnu** transformáciu. Pre zachytenie aj nelineárnych

závislostí vo vstupnom obraze musíme pridať do transformácie aj určitú úroveň nelinearity. Pre tento účel slúži aktivačná vrstva. Ide o aplikáciu vhodnej nelineárnej funkcie na body príznakových máp. Funkcia je aplikovaná samostatne na každý bod mapy, pričom jej výber závisí od typu úlohy. V praxi sa osvedčili funkcie: hyperbolický tangens (1.14) a sigmoid (1.15). Tieto dve funkcie boli považované za štandard. V článku [52] autor ukázal, že efektívnejšou aktivačnou funkciou v porovnaní s uvedenými je rektifikovaná lineárna jednotka (angl. *Rectified Linear Unit*; ReLU) (1.16). ReLU zrýchľuje trénovanie siete zjednodušením výpočtov a súčasne prináša lepšie výsledky.

$$\mathbf{tanh}\,(\boldsymbol{x}) = \frac{\mathbf{e}^{\boldsymbol{x}} - \mathbf{e}^{-\boldsymbol{x}}}{\mathbf{e}^{\boldsymbol{x}} + \mathbf{e}^{-\boldsymbol{x}}} \tag{1.14}$$

$$\boldsymbol{S}\,(\boldsymbol{x}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{e}^{-\boldsymbol{x}}} \tag{1.15}$$

$$\mathbf{ReLU}\,(\boldsymbol{x}) = \mathbf{max}\,(\mathbf{0}, \boldsymbol{x}) \tag{1.16}$$

Špeciálny prípad predstavuje posledná vrstva siete – výstupná vrstva, ktorá je zodpovedná za finálne pridelenie triedy k danej vstupnej vzorke. Každý neurón na tejto vrstve reprezentuje jednu konkrétnu triedu. V praxi je preto dôležité použiť takú aktivačnú funkciu, ktorá vyjadruje pravdepodobnosť zaradenia vstupnej vzorky do každej z tried. Pre tento účel slúži funkcia *softmax* (1.17), ktorá spĺňa podmienky $\sum_{i=1}^{N} \boldsymbol{\sigma}\,(\boldsymbol{X_i}) = \mathbf{1}$ a $\boldsymbol{\sigma}\,(\boldsymbol{X_i}) \geq \mathbf{0}$. Vďaka tomu je možné vnímať jej výstup ako pravdepodobnosť.

$$\boldsymbol{\sigma}\,(\boldsymbol{X_i}) = \frac{\mathbf{e}^{\boldsymbol{X_i}}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} \mathbf{e}^{\boldsymbol{X_k}}} \tag{1.17}$$

Aktivačná vrstva je v praxi štandardne umiestnená za každou konvolučnou a plne-prepojenou vrstvou. Jediným hyperparametrom v prípade tejto vrstvy je typ aktivačnej funkcie. Podobne ako zlučovacia, ani aktivačná vrstva štandardne nedisponuje žiadnymi parametrami.

Normalizačná vrstva

Pre zvýšenie efektivity a zrýchlenie procesu trénovania pomôže okrem iného aj zaradenie normalizačnej vrstvy do štruktúry modelu konvolučnej siete. Normalizácia vstupných dát vykonaná ešte počas ich predspracovania, t. j. pred ich vstupom do konvolučnej siete, pozitívne ovplyvňuje priebeh učenia siete. Vykonaním normalizácie aj vo vnútri siete, napr. pred aktivačnou vrstvou, sú zásadne zredukované veľké rozdiely medzi hodnotami bodov na vstupných mapách. V praxi existuje viacero spôsobov, ako vykonať normalizáciu. Jedným z nich je tzv. **Local Response Normalization** [52]. Tento typ normalizácie

posilňuje súťaživosť neurónov, ktoré sú buď umiestnené na rovnakej pozícii ale v rôznych vstupných mapách a vyjadrujú tak výskyt rozdielných príznakov, alebo sú umiestnené v lokálnej oblasti na spoločnej mape. Oba prístupy normalizácie sú matematicky vyjadrené v (1.18) a (1.19), kde dvojica (i, j) je pozícia aktuálneho bodu, l je index jeho mapy, N (počet máp, resp. šírka okolia bodu, ktoré sú zahrnuté do normalizácie). Symboly k, α a β sú hyperparametre vrstvy. Tento typ normalizácie sa už ale viac v súčasných architektúrach konvolučných sietí neobjavuje, bol nahradený iným typom normalizácie – **Batch Normalization** [53]. Normalizácia po dávkach prebieha nad skupinou máp získaných v za sebou idúcich časových okamihoch a matematicky sa veľmi podobá normalizácii vykonanej vo fáze predspracovania vstupných vzoriek.

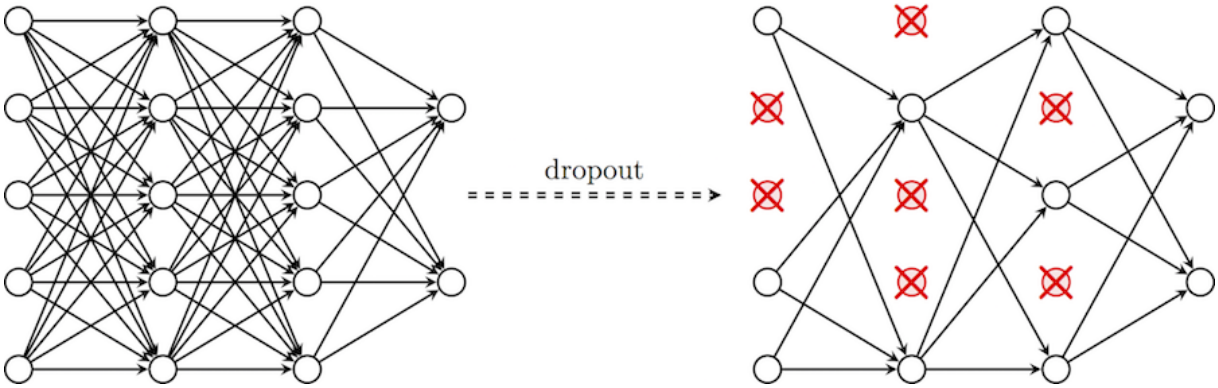
$$Y^l_{i,j} = \frac{X^l_{i,j}}{\left(k \ + \alpha \times \sum_{m=\max\left(0,l-\frac{N}{2}\right)}^{\min\left(H-1,l+\frac{N}{2}\right)} \left(X^m_{i,j}\right)^2\right)^{\beta}}, \tag{1.18}$$

$$Y^l_{i,j} = \frac{X^l_{i,j}}{\left(k \ + \frac{\alpha}{N^2} \times \left(\sum_{i'=i \ -\frac{N}{2}}^{i+\frac{N}{2}} \sum_{j'=j-\frac{N}{2}}^{j+\frac{N}{2}} \left(X^l_{i',j'}\right)^2\right)\right)^{\beta}}, \tag{1.19}$$

Dropout

Jednou z metód, ktorá zrýchľuje trénovanie siete a súčasne znižuje riziko jej pretrénovania, je tzv. **dropout** vrstva [54], [55]. Princíp fungovania tejto vrstvy je jednoduchý - každý neurón je doplnený o pravdepodobnosť, s akou je jeho výstup potlačený (nulovaný). Proces dočasného potlačenia rôznych neurónov sa nanovo vyhodnocuje pre každú trénovaciu vzorku zvlášť. Ak je neurón v danom behu vyhodený z vrstvy ("dropped out"), nepodieľa sa na doprednom vyhodnotení ani následnom spätnom dotrénovaní. Inak povedané, potlačený neurón nie je v danom behu súčasťou siete (príznak detegovateľný týmto neurónom je v danom behu ignorovaný). Obrázok 1.8 zobrazuje príklad dočasného vyhodenia skupiny neurónov z jednotlivých vrstiev siete. Keďže eliminácia neurónov prebieha náhodne, je tak v každom behu použitá pre trénovanie iná jemne odlišná architektúra vrstiev/siete (so zachovaním aktuálnych hodnôt parametrov). Vďaka tomu sa neurón učí detegovať len príznak, ktorý všeobecne pomáha pri klasifikácii bez jeho závislosti od súbežného výskytu iných príznakov (detegovaných ostatnými neurónmi). Dropout vrstva pomáha potlačiť vzájomnú závislosť medzi neurónmi a tak zmenšiť výskyt redundancie v sieti. Vrstva používa len jeden hyperparameter - pravdepodobnosť p, s akou je neurón dočasne vyhodený. Pravdepodobnosť je pre všetky neuróny rovnaká (štandardne 50%), pričom vyhodenie jedného neurónu vôbec neovplyvní pravdepodobnosť vyhodenia iných neurónov. Dropout vrstva je aktívna len počas trénovania. Aby bol jej vplyv na presnosť výstupov

započítaný aj počas testovania, sú vo fáze testovania všetky koeficienty neurónov na dropout vrstve vynásobené pravdepodobnosťou $\boldsymbol{p}$.



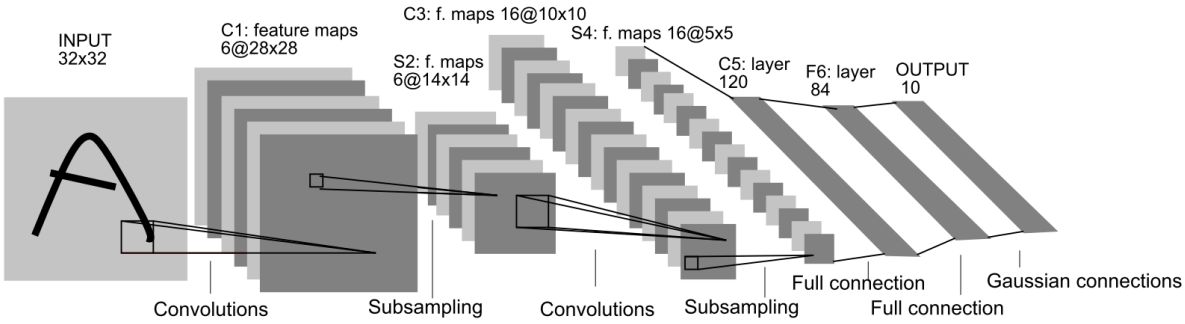
Obr. 1.8: Dropout (Zdroj: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html)

1.3.2 Príklad architektúry konvolučnej siete

Ako praktický príklad konvolučnej siete sme zvolili sieť LaNet-5 [49] pôvodne navrhnutú pre už spomenutú úlohu rozpoznávania rukou písaných číslic. Dôvodom pre voľbu tejto architektúry je jej jednoduchosť a prehľadnosť v porovnaní s novšími ale značne zložitejšími architektúrami. LeNet-5 (graficky zobrazená na obr. 1.9) pozostáva zo vstupnej, výstupnej a šiestich skrytých vrstiev.

Pre extrakciu príznakov sú použité striedajúce dvojice konvolučných a zlučovacích vrstiev ($\boldsymbol{Cx}$ a $\boldsymbol{Sx}$). Kernely konvolučných vrstiev majú rozmery $\mathbf{5 \times 5}$ pixelov s jednotkovým krokom, zlučovacie vrstvy zase $\mathbf{2 \times 2}$ pixelov bez prekrývania. Následnú klasifikáciu vykonávajú dve plne-prepojené vrstvy ($\boldsymbol{Fx}$) nasledované výstupnou vrstvou. Jej úlohou je priradenie finálnej triedy k jednotlivým vstupným vzorkám. V prípade LeNet-5 je pre tento účel použitá ako aktivačná funkcia každého z výstupných neurónov tzv. *Euclidean Radial Basis* (ERB) funkcia. Za každou z uvedených vrstiev (vrátane zlučovacích vrstiev) je nasadená samostatná aktivačná vrstva používajúca aktivačnú funkciu **tanh** (v architektúre nie je zakreslená). Normalizačné a dropout vrstvy v tejto architektúre chýbajú, keďže vznikli až neskôr. Trénovanie a následné testovanie siete boli vykonané nad datasetom MNIST [49], ktorý obsahuje sady obrázkov rukou písaných číslic s rozmermi $\mathbf{32 \times 32}$ pixelov.

Zaujímavosťou siete je spôsob, akým sú redukované rozmery príznakových máp na zlučovacích vrstvách. V porovnaní so súčasnými architektúrami konvolučnej siete je v prípade LeNet-5 použitá rozšírená matematická operácia v (1.20),



Obr. 1.9: Architektúra konvolučnej siete LeNet-5 (Zdroj: [49])

$$\boldsymbol{Y}_{i,j}^{l} = \boldsymbol{b}^{l} + \boldsymbol{a}^{l} \times \left(\sum_{m=0}^{1} \sum_{n=0}^{1} \boldsymbol{X}_{i+m,j+n}^{l} \right) \tag{1.20}$$

kde $\boldsymbol{a}^{l}$ a $\boldsymbol{b}^{l}$ predstavujú ďalšie voľné parametre vrstvy definované pre každú z príznakových máp zvlášť (označených indexom $\boldsymbol{l}$). Obrázok 1.9 ilustruje okrem usporiadania vrstiev aj postupnú transformáciu rozmerov dát, ktoré pretekajú jednotlivými vrstvami siete (šírka a výška máp sa zmenšujú, počet máp naopak narastá). Napriek neaktuálnosti tejto architektúry sa jej princípy nemenia, takže platia rovnako aj pre súčasné konvolučné siete.

1.3.3 Aplikácie konvolučnej siete na reálne úlohy

Prioritnou doménou pre aplikáciu konvolučných sietí bolo a stále je rozpoznávanie vzorov v oblasti spracovania obrazu. Táto oblasť predstavuje počiatky vzniku viacerých modelov (napr. neocognitron [56]), ktoré sa stali predchodcami konvolučných sietí. Modely vychádzajú z prvotných pokusov popísať správanie tej časti mozgu, ktorá je zodpovedná za spracovanie obrazových informácií - **vizuálny kortex**.

Najznámejšie modely, ktoré boli ako prvé oficiálne pomenované konvolučné siete, ponúkli riešenia takých praktických problémov ako je automatizované rozpoznávanie rukou písaných číslic, kľúčovej úlohy pre poštové služby s cieľom automatizovať triedenie pošty podľa PSČ (v USA označované pojmom "ZIP code") [47], [49], [57]. Vďaka úspechu týchto architektúr sa konvolučné siete uchytili aj v iných oblastiach ako je spracovanie zvuku a časových radov [43], [58], [59]. Vďaka rýchlemu rozvoju hardvéru sú konvolučné siete v súčasnosti nasadené do rôznych tak odlišných oblastí ako sú **medicína** (spracovanie medicínskych snímok pre detekciu chorôb [60]–[62]), **bezpečnosť počítačových systémov** (detekcia škodlivých programov [63]), **spracovanie signálov produkovaných nervovými bunkami mozgu** (EEG a detekcia P300 [64]) a iné. Vzhľadom na široké nasadenie konvolučných sietí a ich pozitívne výsledky sme sa rozhodli použiť konvolučnú sieť v oblasti **bezpečnosti počítačových sietí**. V tejto práci sa preto okrem iného venujeme možnostiam nasadenia konvolučných sietí na problém detekcie sieťových

prienikov a anomálneho správania v sieťovej prevádzke.

1.3.4 Trendy v implementácii konvolučnej siete pomocou špecializovaných technických prostriedkov

Vstupnou informáciou do procesu návrhu architektúry konvolučnej siete je abstraktný model siete a zvolená platforma technických prostriedkov. Model popisuje štruktúru konvolučnej siete, danú počtom a typmi vrstiev, ich usporiadaním a prepojením, a rozmery vstupných máp a filtrov. Platforma technických prostriedkov určuje množinu zdrojov použiteľných pri implementácii.

V prípade konvolučnej siete je akcentovaný výber platformy technických prostriedkov s ohľadom na akceleráciu výpočtovo náročných numerických operácií a zníženie celkovej energetickej spotreby. V praxi tak vznikajú rôzne kombinácie platforiem, z ktorých jedna sa sústredí na vykonanie čo najväčšieho počtu výpočtových operácií a druhá slúži ako radič presunu dát systémom, napr. CPU-FPGA, CPU-GPU.

Techniky akcelerácie, použité na model konvolučnej siete, majú vplyv na výkon, zložitosť, a presnosť výsledného systému. Techniky vychádzajú zo základných znalostí o spôsobe, akým všeobecne prebiehajú výpočty v rámci konvolučnej siete. Podľa kap. 1.3.1 je výpočtovo najnáročnejšou operáciou konvolúcia, ktorá sa odohráva pri prechode vstupnej mapy konvolučnou vrstvou. Ide o vektorové operácie, ktoré sú vhodným príkladom vysoko paralelného výpočtu. Preto sa pre implementáciu konvolučnej siete používajú **časová** (angl. *temporal*) a **priestorová** (angl. *spatial*) paralelná architektúra [46], [65].

Časová paralelná architektúra

Platformy CPU a GPU pozostávajú zo skupiny výpočtových jednotiek, ktoré nie sú vzájomne prepojené, ale majú prístup do zdieľanej pamäte (operačná pamäť umiestnená mimo čip). V ich prípade je použitá časová paralelná architektúra so zameraním na akceleráciu operácií nad vektormi dát (angl. *Single Instruction Multiple Data*; SIMD). Vzorovým príkladom vektorových operácií v prípade CNN je **násobenie matíc**. Trendom pre zrýchlenie výpočtov CNN je transformácia vstupných dát do tvaru, ktorý vedie k zníženiu celkového počtu požadovaných operácií pri zachovaní rovnakých výsledkov. Príkladom sú transformácie: rýchla Fourierova transformácia (angl. *Fast Fourier Transform*; FFT), Winograd, a Strassen [46].

Priestorová paralelná architektúra

Platformy FPGA a ASIC tiež pozostávajú zo skupiny výpočtových jednotiek, ktoré majú prístup do operačnej pamäte a zdieľanej vyrovnávacej pamäte, ale sú vzájomne prepojené,

a obsahujú aj vlastnú pamäť. Sieť, prepájajúca výpočtové jednotky, umožňuje prenos dát medzi susednými subsystémami a znižuje požiadavky na frekvenciu prístupov do operačnej pamäte. Registre zase poskytujú rýchly prístup s veľmi malým oneskorením v porovnaní s prístupom do internej spoločnej vyrovnávacej pamäte alebo externej operačnej pamäte. Základom pre implementáciu akcelerátora do FPGA a ASIC je efektívne riadenie prístupu do pamäte a tok signálov architektúrou (angl. *dataflow*).

Presun dát medzi pamäťou a výpočtovými jednotkami predstavuje časovo aj energeticky náročnú úlohu. Preto je hlavným trendom v implementácii konvolučnej siete do zvolenej platformy redukcia vzdialenosti medzi pamäťou a výpočtovými jednotkami. Príkladmi sú vstavané operačné pamäte (eDRAM) [66], umiestnené priamo na čipe, viacvrstvové pamäte (angl. *Hybrid Memory Cube*) [67], [68], a memristory [69]. Podobne je náročný aj presun dát zo senzorov do pamäte a výpočtových jednotiek.

V súčasnosti sa v implementácii konvolučnej siete používajú techniky, ktorých cieľom je zmenšenie objemu vstupných dát a počtu požadovaných operácií pri zachovaní dostatočnej presnosti systému. **Kvantizáciou** dát, ako sú vstupné vzorky a váhy, dochádza ku zmene ich kódovania z formátu s pohyblivou rádovou čiarkou (angl. *floating-point*) na formát s pevnou rádovou čiarkou (angl. *fixed-point*), a ku zmenšeniu ich bitovej šírky. **Kompresiou** modelu je zase zredukovaný počet váh ich vynulovaním a úpravou modelu na riedku sieť. Operácie násobenia takto vynulovaných váh s príznakmi sú vynechané, keďže nemajú žiadny prínos. Zmenšením veľkosti filtrov a zväčšením hĺbky siete pomocou radu sériovo zapojených filtrov je výsledná sieť kompaktná a tak lepšie implementovateľná. Pre dosiahnutie dostatočnej presnosti aj v prípade kompaktnej siete prebieha jej tréning s pomocou už natrénovanej veľkej siete alebo skupiny sietí v úlohe učiteľa. S použitím uvedených techník sa znižujú nároky na výpočtové, pamäťové a prenosové kapacity. Inými slovami, klesá celková spotreba systému a stúpa jeho výkon pri zachovaní hardvérových zdrojov. Detailnejšie sa implementácii konvolučnej siete venujeme v kap. 3.5 a kap. 4.

Kapitola 2

Ciele práce

Hlavným cieľom predkladanej práce je **vytvorenie metodiky návrhu detektora DoS a 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.

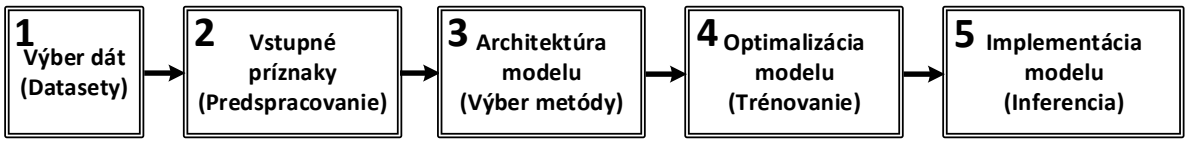
Kroky pre dosiahnutie uvedeného cieľa práce sú:

- návrh jednotlivých etáp metodiky,
- výber vhodných technických prostriedkov pre implementáciu zvolenej etapy navrhnutej metodiky,
- návrh architektúry konvolučnej neurónovej siete (odpovedá zvolenej etape) s cieľom aplikácie FPGA obvodov,
- experimentálne overenie navrhnutej architektúry,
- vyhodnotenie výsledkov vykonaného experimentu.

Kapitola 3

Metodika návrhu detekčnej metódy

V tejto kapitole popisujeme metodiku pre systematický návrh a implementáciu metódy detekcie sieťových anomálií založenej na konvolučnej neurónovej sieti do obvodu FPGA, ktorú sme použili v praktickej časti práce. Sekvencia jednotlivých etáp navrhovanej metodiky je ilustrovaná na obr. 3.1. Navrhovaná metodika nadväzuje na prax v rozpoznávaní vzorov vzhľadom na súvislosť medzi úlohami rozpoznávania vzorov a aplikáciou konvolučnej siete na detekciu sieťových anomálií [70]. Každému z krokov sa detailnejšie venujeme v nasledujúcich podkapitolách.



Obr. 3.1: Sekvencia etáp navrhovanej metodiky

3.1 Dáta - Výber a analýza dostupných datasetov

Na základe teoretického popisu v kap. 1.3 je konvolučná sieť jednou z metód strojového učenia, konkrétne hlbokého učenia. Úspešnosť modelu tak do značnej miery závisí od dostupných trénovacích a testovacích dát. Konkrétne ide o množstvo dostupných validných dát, ktoré sú použiteľné pre trénovanie modelu konvolučnej siete a súčasne dostatočne popisujú zadaný problém. Pre úlohy rozpoznávania vzorov vznikli mnohé datasety z rôznych aplikačných domén, ktoré obsahujú sady vstupných vzoriek spolu s prislúchajúcimi požadovanými výstupmi. Príkladom referenčného datasetu pre úlohy rozpoznávania vzorov v obrazoch je MNIST [49].

Zameraním tejto práce je oblasť detekcie prienikov v sieťovej prevádzke, preto sa ďalej venujeme datasetom z tejto domény. Príkladmi datasetov z oblasti rozpoznávania

sieťových prienikov sú: KDDcup99 [23], NSL-KDD [42], ISCX-2012 [24], TUIDS-DDOS-2015 [71], a CICIDS-2017 [72]. Okrem popisu vlastného datasetu [71] uvádza aj detailnejšie zhrnutie existujúcich a voľne dostupných datasetov sieťovej prevádzky.

Podľa [73] je pre NIDS metódy, založené na detekcii anomálií v sieťovej prevádzke dôležité, aby dataset použitý počas trénovania obsahoval aj samostatný súbor vzoriek reprezentujúcich len normálnu sieťovú prevádzku. Tieto vzorky sú použité pre vytvorenie profilu normálnej prevádzky, s ktorým sú potom porovnávané vstupné vzorky. Všetky zásadné odlišnosti sú následne označené ako anomálie. Všetky ďalej popísané datasety sieťovej prevádzky spĺňajú túto podmienku. Aby bol vytvorený model vhodný pre konkrétnu sieťovú infraštruktúru, teda dokázal čo najpresnejšie odlíšiť normálnu prevádzku od útokov v danej sieti, je zaujímavou myšlienkou použitie kombinácie dostupných datasetov a reálnej prevádzky z tejto siete.

Správna aplikácia datasetov na tvorbu modelu vyžaduje porozumieť štruktúre datasetu, ktorá zahŕňa napríklad pôvod a počet vzoriek použiteľných pre trénovanie a testovanie, dostupnosť, význam a formát ich príznakov ako aj ich rozloženie do rôznych tried. Z tohto dôvodu ďalej uvádzame popis tých datasetov z vyššie uvedených, ktoré boli použité pri realizácii praktickej časti práce.

3.1.1 MNIST

MNIST [49] slúži už dvadsať rokov ako referenčný dataset pre porovnávanie rôznych metód spracovania obrazu. MNIST poskytuje čierno-biele obrázky rukou písaných arabských číslic, ktoré sú rozdelené do desiatich tried (každá trieda reprezentuje jednu číslicu). Dataset pozostáva zo 70 000 obrázkov prevzatých z pôvodných databáz SD-1 a SD-3 inštitúcie NIST. Obrázky sú predspracované a roztriedené do dvoch skupín. Prvá skupina obsahuje 60 000 vzoriek vhodných pre trénovanie, druhá skupina obsahuje 10 000 vzoriek pripravených pre nasledujúce testovanie. Na každom obrázku je zobrazená rukou napísaná číslica s rozmermi $\mathbf{20 \times 20}$ pixelov, ktorá je umiestnená do stredu plochy s rozmermi $\mathbf{28 \times 28}$ pixelov. Aj napriek tomu, že tento dataset nesúvisí priamo s problémom detekcie sieťových prienikov, z dôvodu svojej jednoduchosti je vhodný pre zrozumiteľné vysvetlenie jednotlivých krokov rozpoznávania vzorov a overenie správnosti nami navrhnutého modelu konvolučnej siete.

3.1.2 KDDcup99

KDDcup99 [23] bol niekoľko rokov najpoužívanejší dataset pre ohodnotenie detekčnej metódy a jej porovnanie s inými metódami formou benchmarku. Tento dataset vznikol pôvodne z datasetu DARPA-1998 a bol použitý pre účely súťaže zameranej na detekciu

prienikov (The Third International Knowledge Discovery and Data Mining Tools Competition).

Sieťová prevádzka spracovaná v datasete pochádza z umelo pripraveného prostredia simulujúceho sieť stredne veľkej základne amerického letectva. Trénovacia prevádzka bola zachytená cez TCPdump [74] počas siedmych týždňov (∼ **5** miliónov záznamov o spojeniach), testovacia prevádzka bola získaná rovnakým spôsobom počas dvoch ďalších týždňov (∼ **2** milióny záznamov). Do prevádzky boli umelo vložené aj rôzne typy útočných spojení. Jedna vzorka v datasete predstavuje jedno spojenie – skupinu paketov, ktoré tečú medzi dvojicou IP adries a používajú konkrétny protokol. Spojenie zahŕňa v skutočnosti dva toky, pre každý smer jeden tok.

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. Útočné vzorky sú ďalej rozdelené do podkategórií: počas trénovania 24 typov, počas testovania ďalších 14 nových typov. Útoky sú zhrnuté v Tab. 3.5, pričom ich detailný popis je dostupný v [75]. Trénovacia a testovacia sada sú vygenerované nezávisle.

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 (Tab. 3.1), informácie o **prevádzke** počítané cez posuvné okno podľa dvoch spôsobov zoskupovania spojení: “same host” a “same service”, ktorý sa ďalej delí na “time-based” (okno o 2 sekundách pre útoky typu U2R a R2L) (Tab. 3.3) a “connection-based” (okno o 100 vzorkách pre útoky typu DoS a Probing) (Tab. 3.4), informácie o **obsahu** sledujúce niektoré vlastnosti spojení a vychádzajúce z obsahu paketov skrytých v aplikačných dátach najmä kvôli útokom typu U2R a R2L (Tab. 3.2).

3.1.3 NSL-KDD

NSL-KDD [42] 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. Obmedzený rozsah datasetu vedie k častému používaniu vzoriek z trénovacej sady aj pre testovanie, čím je do značnej miery ovplyvnená presnosť testovania metódy. Nerovnomerné zastúpenie tried zase ovplyvňuje klasifikáciu, počas ktorej sú viac preferované útoky s väčšou mierou zastúpenia v datasete. Inými slovami, klasifikačné metódy trénované na nebalancovanej dátovej množine majú tendenciu k určitej zaujatosti v prospech prevažujúcich tried. Z týchto dôvodov bol pôvodný KDDcup99 nahradený NSL-KDD pri porovnávaní výkonu detekčných metód.

Pri tvorbe datasetu NSL-KDD boli odstránené všetky duplicitné záznamy. Dôsledkom

Tabuľka 3.1: **Základné príznaky v KDDcup99** (Adaptované podľa zdroja: [71])

Značka / Názov príznaku (v angl.)	**Typ**	**Popis**
1. Duration	C	Časový interval spojenia v sekundách
2. Protocol-type	D	Typ transportného protokolu, napr. TCP, UDP a iné
3. Service	D	Sieťová služba cieľovej stanice, napr. HTTP, TELNET a pod.
4. Flag	D	Príznak spojenia
5. Src-bytes	C	Množstvo dát (počet bajtov) prenesených zo zdroja do cieľa
6. Dst-bytes	C	Množstvo dát (počet bajtov) prenesených z cieľa do zdroja
7. Land	D	1 - ak je spojenie z/na rovnakú stanicu/port; 0 - inak
8. Wrong fragment	C	Počet chybných fragmentov
9. Urgen	C	Počet urgentných paketov
C - spojitý (angl. *continuous*) D - diskrétny (angl. *discrete*)		

toho je značné zmenšenie počtu vzoriek a to konkrétne v prípade trénovacej sady z 3 925 650 útočných vzoriek na 262 178 a 972 781 normálnych vzoriek na 812 814, v prípade testovacej sady boli útočné vzorky zredukované z 250 436 na 29 378 a normálne vzorky z 60 591 na 47 911. Výsledkom experimentov vykonaných v [42] nad rôznymi metódami strojového učenia pre detekciu sieťových útokov je zistenie, že všetky testované metódy dosiahli vysokú presnosť a teda nie je možné navzájom ich porovnať s použitím pôvodných metrík, ako sú presnosť, miera detekcia a miera falošných hlásení. Z uvedeného vyplýva nevyhnutnosť návrhu účinnejších metrík.

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.

3.1.4 ISCX-2012

ISCX-2012 [24] je dataset sieťovej prevádzky obsahujúci okrem reálnych normálnych tokov aj rôzne formy útokov vrátane zložitejších DDoS útokov prebiehajúcich vo viacerých fázach. Pre dosiahnutie realistickej, dynamickej a automatizovanej tvorby sieťových datasetov existujú v prípade ISCX-2012 datasetu dva typy profilov s abstraktným popisom správania sa normálnych používateľov a útočníkov cez štatistické charakteristiky: $\boldsymbol{\alpha}$-profily popisujú útoky, $\boldsymbol{\beta}$-profily reprezentujú priebeh služieb normálnej prevádzky. Zvolené charakteristiky $\boldsymbol{\alpha}$-profilov vychádzajú z analýzy scenárov rôznych typov útokov.

Tabuľka 3.2: **Príznaky založené na obsahu správ v KDDcup99** (Adaptované podľa zdroja: [71])

Značka / Názov príznaku (v angl.)	**Typ**	**Popis**
10. Hot	C	Počet "hot" indikátorov ("hot": počet prístupov do adresárov, vytvorenie a vykonanie programu)
11. Num-failed-logins	C	Počet neúspešných pokusov o prihlásenie
12. Logged-in	D	1 - ak úspešné prihlásenie; 0 - inak
13. Num-compromised	C	Počet kompromitujúcich okolností (kompromitujúca okolnosť: počet chýb typu "súbor nenájdený" a príkazov skoku)
14. Root-shell	D	1 - ak bol sprístupnený root-shell; 0 - inak
15. Su-attempted	D	1 - ak pokus o "su root"; 0 - inak
16. Num-root	C	Počet prístupov pod pouŽívateľom "root"
17. Num-file-creations	C	Počet operácií typu "vytvor súbor"
18. Num-shells	C	Počet shell promptov
19. Num-access-files	C	Počet operácií nad súbormi riadenia prístupu
20. Num-outbound-cmds	C	Počet odchádzajúcich príkazov v FTP spojení
21. Is-host-login	D	1 - ak login patrí do "hot" zoznamu; 0 - inak
22. Is-guest-login	D	1 - ak je login "guest"; 0 - inak
C - spojitý (angl. *continuous*) D - diskrétny (angl. *discrete*)		

Na druhej strane, $\boldsymbol{\beta}$-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 práve $\boldsymbol{\alpha}$- a $\boldsymbol{\beta}$-profilmi. Proces generovania a odchytávania správ obsiahnutých v datasete prebiehal v reálnej sieťovej topológii a v reálnom čase po dobu 7 dní. ISCX-2012 dataset poskytuje všetky odchytené správy vo formáte *pcap*, t.j. celý obsah jednotlivých paketov je voľne dostupný. Označenie triedy, do ktorej jednotlivé vzorky patria, nie je uvedené pre každý paket zvlášť, ale je pridelené spojeniam a uložené v samostatnom *XML* dokumente.

Výhodou datasetu ISCX-2012 oproti predchádzajúcim uvedeným datasetom (KDDcup99 a NSL-KDD) je množstvo použiteľných vzoriek, aktuálne typy útokov, a neobmedzenosť výberu príznakov, ktoré sú vďaka formátu *pcap* voľne extrahovateľné. Medzi dostupné príznaky patria všetky hodnoty obsiahnuté v hlavičkách správ ako aj rôzne štatistické charakteristiky vypočítané zo správ obsiahnutých v rámci jedného spojenia. Pod spojením rozumieme v tomto prípade sadu správ, ktoré majú spoločný niektorý príznak, napr. port identifikujúci službu alebo IP adresy zdrojovej a cieľovej stanice.

Tabuľka 3.3: **Časové príznaky v KDDcup99** (Adaptované podľa zdroja: [71])

Značka / Názov príznaku (v angl.)	**Typ**	**Popis**
23. Count	C	Počet spojení na rovnakú stanicu v rámci posledných dvoch sekúnd
24. Srv-count	C	Počet spojení na tú istú službu v rámci posledných dvoch sekúnd v rámci spojení na rovnakú stanicu
25. Serror-rate	C	Percentuálny podiel spojení, ktoré vykazujú "SYN" chyby v rámci spojení na rovnakú stanicu
26. Srv-serror-rate	C	Percentuálny podiel spojení, ktoré vykazujú "SYN" chyby v rámci spojení na rovnakú stanicu a službu
27. Rerror-rate	C	Percentuálny podiel spojení, ktoré vykazujú "REJ" chyby v rámci spojení na rovnakú stanicu
28. Srv-rerror-rate	C	Percentuálny podiel spojení, ktoré vykazujú "REJ" chyby v rámci spojení na rovnakú stanicu a službu
29. Same-srv-rate	C	Percentuálny podiel spojení na rovnakú stanicu a službu
30. Diff-srv-rate	C	Percentuálny podiel spojení na rovnakú stanicu ale rôzne služby
31. Srv-diff-host-rate	C	Percentuálny podiel spojení na rôzne stanice
C - spojitý (angl. *continuous*) D - diskrétny (angl. *discrete*)		

Z týchto dôvodov sme v práci zvolili ISCX-2012 ako referenčný dataset.

3.2 Vstupné príznaky

Pre praktické použitie vzoriek z datasetu je prvým krokom výber tých príznakov, ktoré sú nakoniec použité ako vstupy do modelu. V prípade datasetov s malým počtom dostupných príznakov (kap. 3.1.2 a kap. 3.1.3) môžeme použiť priamo všetky príznaky bez prvotnej analýzy. Ale ak sú vzorky dostupné vo formáte, ktorý priamo neposkytuje konkrétnu množinu príznakov (napr. *pcap* formát uvedený v kap. 3.1.4), je voľba príznakov nejasná a navyše 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 [76]. Predspracovanie zahŕňa

Tabuľka 3.4: **Spojovo-orientované príznaky v KDDcup99** (Adaptované podľa zdroja: [71])

Značka / Názov príznaku (v angl.)	**Typ**	**Popis**
32. Dst-host-count	C	Počet cieľových staníc
33. Dst-host-srv-count	C	*Srv-count* (24) pre cieľovú stanicu
34. Dst-host-same-srv-rate	C	*Same-srv-rate* (29) pre cieľovú stanicu
35. Dst-host-diff-srv-rate	C	*Diff-srv-rate* (30) pre cieľovú stanicu
36. Dst-host-same-src-port-rate	C	*Same-src-port-rate* pre cieľovú stanicu
37. Dst-host-srev-diff-host-rate	C	*Diff-host-rate* pre cieľovú stanicu
38. Dst-host-serror-rate	C	*Serror-rate* (25) pre cieľovú stanicu
39. Dst-host-srv-serror-rate	C	*Srv-serror-rate* (26) pre cieľovú stanicu
40. Dst-host-rerror-rate	C	*Rerror-rate* (27) pre cieľovú stanicu
41. Dst-host-srv-rerror-rate	C	*Srv-rerror-rate* (28) pre cieľovú stanicu
C - spojitý (angl. *continuous*) D - diskrétny (angl. *discrete*)		

Tabuľka 3.5: **Zoznam útokov v KDDcup99 podľa** [71]

DoS	**Probe**	**U2R**	**R2L**
apache2	ipsweep	eject	dictionary
back	mscan	ffbconfig	ftp-write
land	nmap	fdformat	guest
mailbomb	saint	loadmodule	imap
SYN flood	satan	perl	named
ping of death		ps	sendmail
process table		xterm	xlock
smurf			xsnoop
syslogd			
teardrop			
udpstorm			

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. Zložitosť predspracovania súčasne ovplyvňuje jeho aplikáciu v reálnom čase. Ak vyberáme z odchytených paketov len priame príznaky ako sú hodnoty v hlavičkách, môže tento proces bežať v reálnom čase. Na druhej strane, ak sú okrem základných príznakov požadované aj rôzne štatistické charakteristiky, prípadne aplikačné dáta uložené v tele paketov, takéto predspracovanie vyžaduje pre svoj beh v reálnom čase použitie špecializovanej hardvérovej platformy ASIC alebo FPGA.

Príznaky priamo dostupné so sieťovej prevádzky môžeme analyzovať na troch úrovniach: po paketoch, po tokoch, a po časových oknách. Navyše, základné príznaky získané

z hlavičiek majú rôznu oblasť platnosti v závislosti od typu hlavičky, v ktorej sú umiestnené, napr. príznaky z L2 hlavičky obsahujú len informácie o lokálnej sieti. Výber relevantných príznakov závisí do značnej miery od typov útokov, na ktorých detekciu sa konkrétna metóda zameriava.

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 získaných z útokov. Otázkou pri výbere príznakov je hlavne formát pozorovaného objektu, ktorý je základný ďalej už nedeliteľný. V praxi ide hlavne o výber medzi paketom a tokom. Vychádzajúc z [73] je tok vhodnejším objektom analýzy, keďže zachytáva vzájomné vzťahy medzi komunikujúcimi stranami v čase. Ale extrakciou dátových segmentov zo sieťovej prevádzky získame údaje umiestnené v hlavičkách paketov, nie priamo údaje o tokoch. Preto sú najprv extrahované hodnoty základných príznakov priamo z hlavičiek paketov. Až následne sú tieto príznaky metódou posuvných okien použité na výpočet štatistických charakteristík [77].

Unikátnym identifikátorom toku je n-tica základných primárnych príznakov: zdrojové a cieľové IP adresy, zdrojové a cieľové porty (kde cieľový port má značne vyššiu prioritu), a protokol. Kvôli časovej synchronizácii tokov je k uvedeným primárnym príznakom priložená aj časová pečiatka, ktorá uvádza presný čas začiatku toku. Okrem toho sú dostupné aj sekundárne príznaky špecifické len pre konkrétny protokol (TCP, UDP, a ICMP), napr. TCP flagy. Na zachytenie časových závislostí majú pozitívny efekt časové príznaky, ktoré sú vypočítané agregáciou (súčtom) alebo spriemerovaním primárnych príznakov v rámci časového okna, napr. priemerná medzera medzi paketmi a priemerný počet paketov. Šírka okna je určená buď dĺžkou časového intervalu alebo počtom spojení. Voľba vhodnej dĺžky okna vyžaduje znalosti v doméne sieťovej bezpečnosti. Pre zachytenie krátkodobých aj dlhodobých udalostí je vhodné použiť viacero okien s rôznou šírkou. Tento prístup sme zvolili aj my pri tvorbe 3D objektov reprezentujúcich aktuálny stav sieťovej prevádzky, ktorého popis je uvedený v podkapitole 3.2.8. [76] uvádza prehľad rôznych príznakov dostupných na každej úrovni analýzy (priame príznaky, príznaky tokov, a príznaky z časových okien).

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. Jeho matematický predpis je definovaný vzťahom (3.1), kde $\boldsymbol{X}_{\mathbf{min}}$ a $\boldsymbol{X}_{\mathbf{max}}$ sú najmenšia a najväčšia zo všetkých hodnôt daného príznaku, a $\boldsymbol{X}$ je aktuálna hodnota príznaku, ktorú škálujeme.

$$\boldsymbol{X}_{\text{norm}} = \frac{\boldsymbol{X} - \boldsymbol{X}_{\mathbf{min}}}{\boldsymbol{X}_{\mathbf{max}} - \boldsymbol{X}_{\mathbf{min}}} \tag{3.1}$$

Alternatívou je štandardizácia alebo tzv. *Z-score* normalizácia. Aj v tomto prípade ide o transformáciu hodnôt príznakov, ktorej cieľom je dosiahnutie nulovej strednej hodnoty (centrovanie) a jednotkového rozptylu. Normalizácia je dôležitou operáciou najmä v prípade metód, ktoré porovnávajú vzorky na základe vzdialenosti. Vykonaním normalizácie pre všetky príznaky bude vplyv každého z nich na výslednú vzdialenosť rovnaký. Matematický predpis pre normalizáciu je definovaný vzťahom (3.2), kde $\bar{\boldsymbol{X}}$ je stredná hodnota príznaku $\boldsymbol{X}$ a $\boldsymbol{\sigma_X}$ je jeho štandardná odchýlka.

$$\boldsymbol{X}_{\text{z-score}} = \frac{\boldsymbol{X} - \bar{\boldsymbol{X}}}{\boldsymbol{\sigma_X}} \tag{3.2}$$

3.2.4 Redukcia rozmerov

Podľa [78] 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.

Zaujímavým riešením redukcie rozmerov v [79] je náhrada jedného spoločného priestoru za väčší počet malých podpriestorov, ktoré používajú vždy len dvojice vybraných príznakov. Takto je pre každú dvojicu príznakov vytvorený samostatný 2D priestor. Použité príznaky sú uvedené v [79], [80].

Otázkou pri vykonaní redukcie priestoru, napr. cez PCA, je aj voľba počtu príznakov (v PCA ide o voľbu počtu hlavných komponentov), ktoré použijeme v ďalších fázach – kompresný pomer. Keďže toky s rôznymi protokolmi (TCP, UDP a ICMP) majú rôzne vlastnosti, je rozumné pre vzorky každého protokolu vykonať PCA samostatne (s použitím rôzneho počtu hlavných komponentov) s odlišným kompresným pomerom. Takýto prístup sa osvedčil v [81].

3.2.5 Kompresia dát cez zhlukovanie tokov

Mierne odlišný pohľad na tok ako základný objekt pozorovania je použitý v [80], ktorý pod tokom uvádza skupinu paketov s rovnakou IP adresou (zdrojová alebo cieľová adresa) a maskou. Inými slovami, všetky pakety v rámci okna patriace do tej istej siete danej IP adresou a maskou tvoria jeden tok. Adresa a maska určujú úroveň agregácie, pričom v [80] bolo použitých osem úrovní: /8, /16, /24, a /32 pre zdrojovú a cieľovú IP adresu zvlášť. Analýza prebieha v neprekrývajúcich sa oknách s pevnou časovou šírkou. Ako príznaky sú použité rôzne normalizované (max-min) štatistické charakteristiky počítané v rámci okna. Prevažne ide o histogramy, napr. počet rôznych zdrojových a cieľových IP adries, počet paketov, a pod.

Zhlukovanie tokov, t.j. združenie podobných tokov dokopy, prispieva k riešeniu problému s veľkým objemom spracovateľných dát a často vedie k lepším výstupom z dôvodu zvýšenia informačnej hodnoty v jednotlivých zoskupeniach.

3.2.6 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. Pomocou vektora indikátorov sme schopní kódovať jednotlivé symboly. Takéto kódovanie sa nazýva "one-hot", keďže pre každú vzorku je aktívny vždy len jeden z indikátorov. 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.

3.2.7 Príklady vizualizácie sieťovej prevádzky

Kim a Reddy použili v [82] IP adresy, porty a transportné protokoly na rozloženie paketov sieťovej prevádzky do skupiniek (význam každej skupinky je daný konkrétnou n-ticou hodnôt v týchto príznakoch), ktoré reprezentujú jednotlivé toky. Pre každú zo skupiniek (kôpky) sú vypočítané štatistické hodnoty vrátane počtu paketov alebo bajtov, ktoré vyjadrujú intenzitu každej kôpky. Vstupná sieťová prevádzka je spracovaná postupne metódou posuvných okien. Rozloženie do skupiniek a výpočet štatistík je vykonaný pre každé okno samostatne. Výstupy z jedného okna tak reprezentujú jednu vzorku. Na vzorky môžeme ďalej aplikovať metódy pre extrakciu skrytých príznakov, ktoré sa viac hodia pre klasifikáciu v sieťovej doméne.

Pre zachytenie časových závislostí bol v [83] použitý ako jeden z príznakov čas príchodu (odchytenia) paketov z dôvodu sledovania využitia rôznych príznakov v čase. Takto vytvorené snímky v rámci aktuálneho okna popisujú aktuálny stav prevádzky. Pri návrhu je preto úlohou zvoliť taký formát popisu, v ktorom je výskyt rôznych typov sieťových prienikov najlepšie viditeľný zvolenou klasifikačnou metódou a tak vzájomne rozlíšiteľný.

Vizualizáciu sieťovej prevádzky môžeme vnímať ako mapovanie sieťových tokov do dvojrozmerného geometrického priestoru. Príkladom nástroja pre vizualizáciu sieťovej prevádzky je NetSCENE [84], ktorý v pravidelných intervaloch (formou posuvných okien) odchytáva pakety a spracuje obsah ich hlavičiek. Použité sú zdrojová IP adresa rozdelená na dva príznaky: sieťová časť a hostiteľská časť (ich dĺžka je vyvodená zo samotnej infraštruktúry siete, nie z obsahu paketov), a Hop-count (počet skokov vypočítaný z TTL). Pre vytvorenie 2D objektu sú použité ako x-os a y-os sieťová a hostiteľská časť zdrojovej IP adresy, a ako hodnoty pixelov sú použité počty paketov. Spolu s počtom paketov je v každom z pixelov uložená aj hodnota Hop-count.

3.2.8 Náš pohľad na sieťovú prevádzku

Vzhľadom na prístupy publikované vo viacerých vedeckých článkoch a zhrnutých vyššie, zvolili sme ako základný objekt pozorovania tok. Keďže predpokladáme, že v prípade reálnej sieťovej prevádzky sú dostupné len vzorky odchytené vo forme paketov, je nevyhnutné najprv zoskupiť odchytené pakety do tokov. Následne sú z hlavičiek extrahované hodnoty polí, ktoré sa viažu na toky. Nakoniec sú tieto hodnoty použité pre výpočet štatistických charakteristík. Z dôvodu vykonania týchto operácii v reálnom čase 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. Malé okno ale nemusí zachytiť dlhotrvajúce útoky. Na druhej strane, dlhé okno je presným protikladom, t.j. jeho odchytenie, príprava a spracovanie vyžaduje viac času a vedie teda k väčšiemu oneskoreniu, ale takéto okno je schopné zachytiť aj dlhšie priebehy útokov. V prípade detekčnej metódy potrebujeme schopnosti oboch prípadov. Takže nie je možné zvoliť len jednu z týchto možností, keďže potrebujeme zachytiť rôzne časové závislosti. 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 (3.3),

$$\boldsymbol{W_y = W_0 \times (1 + \alpha \times y)} \tag{3.3}$$

kde $\boldsymbol{W_0}$ je základná šírka okna a $\boldsymbol{\alpha}$ je základný prírastok.

Každému prvku v tejto postupnosti je pridelená hodnota, ktorá vyjadruje intenzitu (napr. počet odchytených paketov) v danom časovom okne (index na x-ovej osi) o danej šírke (index na y-ovej osi) a pre daný tok (index na z-ovej osi). Z toho vyplýva, že *y-tý* riadok, braný ako postupnosť časových okien, zachytáva časový interval s dĺžkou danou vzťahom (3.4),

$$\boldsymbol{\Delta T\,(y) = N \times W_y} \tag{3.4}$$

kde $\boldsymbol{N}$ je pevne stanovený počet okien v riadku, a $\boldsymbol{W_y}$ je šírka jedného okna.

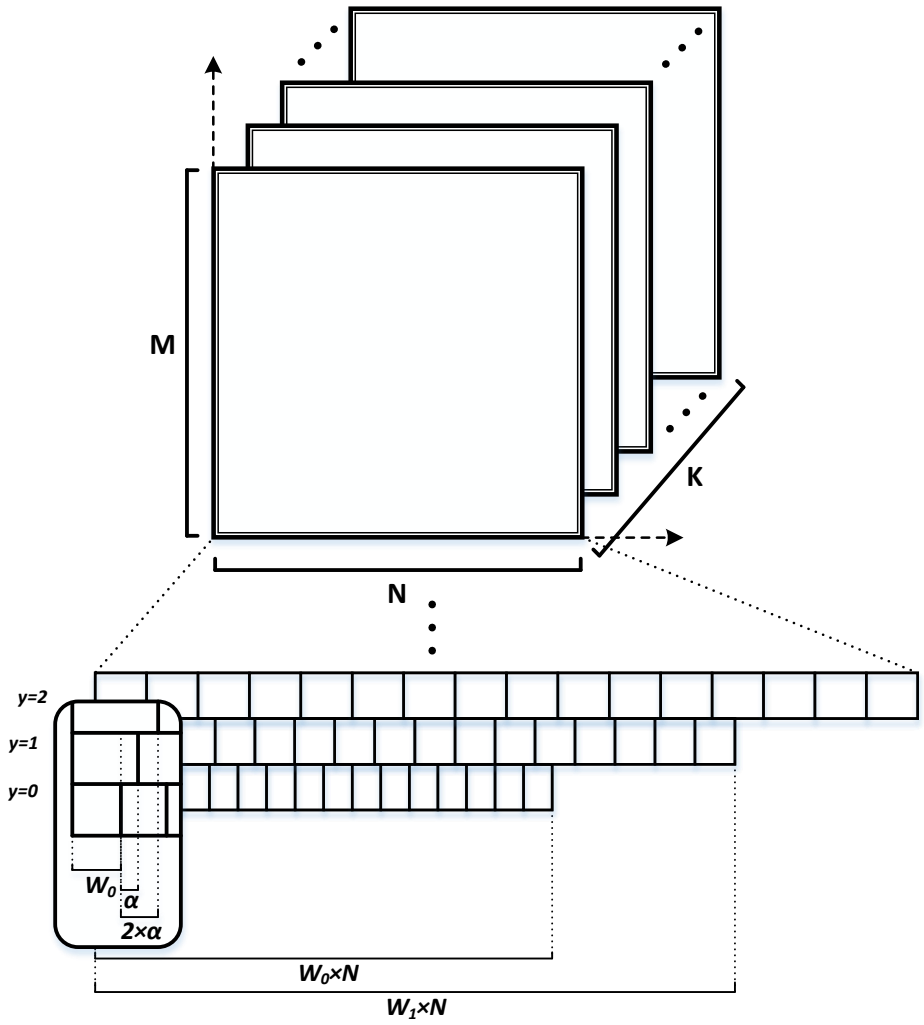
Keďže dĺžka tohto časového intervalu závisí od indexu na y-ovej osi, rôzne riadky zachytávajú rôzne dlhé časové intervaly sieťovej prevádzky. Obr. 3.2 ilustruje vizuálny popis štruktúry takto definovaných snímok.

3.3 Architektúra modelu

Ďalším krokom je návrh architektúry modelu, ktorý na základe vstupných predspracovaných dát vykoná klasifikáciu, v našom prípade detekciu sieťových prienikov. Vzhľadom na súčasný trend nasadenia hlbokého učenia [85] a z dôvodov jeho prínosu nielen do oblasti spracovania obrazu ale aj mnohých ďalších oblastí sme pre implementáciu hlavnej časti detekčnej metódy zvolili model konvolučnej neurónovej siete (popísaná v kap. 1.3). V nasledujúcej podkapitole uvádzame stručný popis hlbokého učenia.

3.3.1 Hlboké učenie

Hlboké učenie patrí do kategórie metód strojového učenia, ktoré vykonávajú nelineárne spracovanie informácie pre účely výberu a transformácie príznakov, následnej analýzy a klasifikácie vzorov. Strojové učenie pristupuje k týmto úlohám ako k hľadaniu vhodných pravidiel na základe dopredu známeho vzťahu medzi vstupnými a požadovanými výstupnými dátami. Tento prístup sa značne odlišuje oproti klasickým riešeniam postaveným



Obr. 3.2: Štruktúra navrhnutých snímok sieťovej prevádzky

na manuálnom (človekom vykonanom) naprogramovaní pravidiel. V podstate ide o transformáciu vstupných dát na zmysluplné výstupy – reprezentácie, ktoré sú najvhodnejšie pre riešenie zadanej úlohy. Pod reprezentáciou rozumieme určitú abstrakciu vstupných dát, kedy sa špecifickým spôsobom pozeráme len na určitú časť dát. Rôzne úlohy vyžadujú nájdenie odlišných reprezentácií v preddefinovanom priestore možných transformácií, pričom závisia od aplikačnej domény. V metódach, ktoré vychádzajú z hľadania reprezentácií, ide o reprezentačné učenie (angl. *representations learning*) [85].

Hlboké učenie rozširuje uvedený prístup hľadaním reprezentácií, ktoré sú určené postupnosťou hlbších skrytých reprezentácií hierarchicky usporiadaných do vrstiev. Zložitejšia reprezentácia je tak tvorená prepojením jednoduchších reprezentácii umiestnených na nižších vrstvách [45]. V tomto kontexte, hlboké učenie zachytáva zložité, často skryté, reprezentácie tým, že vrstvením tvorí oveľa hlbšiu hierarchiu reprezentácii pozorovaných objektov.

Výraznou pridanou hodnotou hlbokého učenia do oblasti strojového učenia je automatizácia extrakcie príznakov. Príznaky závislé od aplikačnej domény boli pôvodne extrahované expertmi ručne pre každý typ úlohy zvlášť. Hlboké učenie umožňuje automaticky zvoliť a extrahovať zložité reprezentácie, pričom vykonáva tieto operácie počas trénovania. Reprezentácie sú tak zakódované priamo v modeli, konkrétne v jeho parametroch. Modely hlbokého učenia sa v porovnaní s modelmi klasických metód strojového učenia učia okrem samotného mapovania vstupu na výstup aj extrahovať efektívne a často skryté reprezentácie vstupných dát. Vďaka hierarchickému usporiadaniu reprezentácií po vrstvách je ich učenie jednoduchšie, keďže rozdiely v zložitosti reprezentácií na susedných vrstvách nie sú veľké. Postupnou aplikáciou základnej transformácie – trénovaním – je možné prejsť od jednoduchých reprezentácií až po zložité. Keďže hľadanie a následný výber vhodnej reprezentácie závisia najmä od vstupných dát a prebiehajú dynamicky, hlboké učenie je možné nasadiť na rôznorodé úlohy spracovania signálov a informácií. Neviazanosť hlbokého učenia na konkrétnu aplikačnú doménu považujeme za dôležitú prednosť tohto prístupu strojového učenia. V prípade hlbokých neurónových sietí, ktoré sú najvýraznejším reprezentantom hlbokého učenia, je uvedený princíp hlbokého učenia vyjadrený hĺbkou modelu, teda počtom jeho vrstiev, kde každá ďalšia vrstva prináša hlbšiu reprezentáciu vstupných dát.

Teória a princípy hlbokého učenia sú známe už dlho, ale výsledky jeho praktického nasadenia sa prejavili až nedávno. Dôvodom pre strmý nárast nasadenia hlbokého učenia na reálne problémy je:

- dostupnosť **rozsiahlych dátových množín**,
- dostupnosť **výpočtových zdrojov** vo forme výkonnejších technických prostriedkov umožňujúcich paralelné výpočty,
- zásadné **pokroky** v oblasti strojového učenia, ktoré sa prejavili až pri trénovaní hlbokých modelov, napr. efektívnejšie aktivačné funkcie, lepšia inicializácia váh, a nové optimalizačné postupy [86].

K rozvoju hlbokého učenia prispel rozvoj softvérových nástrojov, ktoré sú vhodné pre vývoj aplikácii hlbokého učenia. Uvedené novinky umožnili navrhnúť a natrénovať rozsiahlejšie modely, ktoré zodpovedajú aktuálnym problémom. V súčasnosti pretrváva trend prehlbovať modely hlbokého učenia s cieľom dosiahnuť lepšie, presnejšie výsledky. Prehľad histórie hlbokého učenia môžeme nájsť v [45], [86].

Vzhľadom na zameranie práce je aktuálnou otázkou možnosť použiť metódy hlbokého učenia aj 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 tak predstavuje v tomto kontexte špecifický tok alebo skupinu tokov, ktoré sa značne odlišujú od ostatných, normálnych tokov. Okrem toho sú manuálny výber a extrakcia vhodných príznakov zo sieťovej prevádzky náročné a vyžadujú hlboké znalosti v oblasti detekcie sieťových prienikov, viď. kap. 3.2.2. Schopnosť modelov hlbokého učenia automaticky extrahovať skryté príznaky zjednodušuje tento proces. Navyše, hlboké učenie umožňuje objaviť a použiť aj tie 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 [78], [87], [88].

3.4 Optimalizácia modelu - trénovanie

Po získaní vhodnej dátovej množiny, ktorá dostatočne popisuje daný problém, a po návrhu modelu, ktorý v sebe zahŕňa výber metódy a hyperparametrov (statické parametre definujúce architektúru modelu, napr. počet a typ vrstiev, počet neurónov v jednotlivých vrstvách), je kľúčovou fázou nastavenie parametrov modelu. Cieľom tohto kroku je hľadanie takej kombinácie parametrov (váh a biasov) modelu, pri ktorých sa model správa podľa vopred definovaných požiadaviek. Z tohto pohľadu vnímame trénovanie ako proces riešenia optimalizačnej úlohy.

Vzhľadom na zložitosť reálnych problémov nie je ale možné nastaviť model analyticky. Preto existujú optimalizačné metódy, ktoré opakovaním postupnosti krokov posúvajú model do požadovaného stavu. Tento proces sa všeobecne nazýva trénovanie alebo učenie modelu a predstavuje jednu z výpočtovo najnáročnejších kľúčových fáz pri tvorbe modelu strojového učenia. V tejto podkapitole uvádzame len všeobecný informatívny popis trénovania bez detailov a matematických vzťahov. Dôvodom je primárne zameranie práce na akceleráciu inferencie, ktorá je popísaná v ďalšej podkapitole.

Trénovanie, ako každá optimalizačná úloha, požaduje vopred definovanú optimalizačnú funkciu. V prípade trénovania modelov strojového učenia má optimalizačná funkcia niekoľko názvov. V práci používame pomenovanie chybová funkcia (angl. *error function*). Vložením predikovanej hodnoty (výstup z modelu po spracovaní vzorky) a skutočnej vopred známej triedy do chybovej funkcie získame chybu predikcie. Cieľom trénovania je minimalizácia chybovej funkcie 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. Ich základom je použitie spádu (angl. *gradient*), ktorý je matematicky vyjadrený cez deriváciu funkcie a určuje smer zmeny funkčnej hodnoty vzhľadom na zmenu vstupnej premennej – smer zväčšovania funkčnej hodnoty. Zmenšenie funkčnej hodnoty dosiahneme, ak zmeníme vstupnú premennú v opačnom smere, teda so zápornou hodnotou gradientu. Princíp záporného spádu je použitý pre minimalizáciu chybovej funkcie.

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*) namiesto spracovania všetkých vzoriek z dátovej množiny. Slovo "stochastický" v prípade SGD vyjadruje náhodnosť, ktorá je vložená do výpočtu gradientu. V každej iterácii je gradient vypočítaný ako stredná hodnota gradientov vypočítaných samostatne pre každú zo vzoriek v rámci náhodne vybranej skupiny a teda závisí od výberu vzoriek do tejto skupiny. Aplikácia SGD a spracovanie vzoriek po skupinkách je spôsob, ako sa vysporiadať s trénovaním na veľkých datasetoch typických pre hlboké učenie.

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. Ak sa na model viacvrstvovej neurónovej siete pozeráme ako na zloženú funkciu, potom metóda spätného šírenia gradientu počíta gradient na základe reťazového pravidla pre deriváciu zloženej funkcie. Výpočet gradientu na vrstvách prebieha rekurzívne smerom od výstupnej vrstvy až po vstupnú vrstvu. Šírením gradientov v opačnom smere, teda od výstupnej vrstvy späť až k vstupnej vrstve, je možné pomocou optimalizačnej metódy (napr. SGD) upraviť parametre každého neurónu tak, aby sa pre dané vstupy celková chyba modelu zmenšila. To je dosiahnuté zmenou aktuálnych hodnôt parametrov o zápornú hodnotu gradientu. V praxi sa okrem SGD používajú aj ďalšie optimalizačné metódy, napr. AdaGrad, RMS-Prop, a ADAM.

3.4.2 Typy trénovania

Hlavným faktorom pre výber konkrétneho typu učenia je dostupnosť, resp. nedostupnosť značiek pre vzorky v dátovej množine. Pod značkou rozumieme požadovanú triedu (v prípade klasifikácie) alebo hodnotu (v prípade regresie), ktorú má model prideliť danej vzorke.

Podľa dostupnosti trénovacej dátovej množiny a prislúchajúcich značiek sú k dispozícii

štyri typy učenia [85]:

(a) **Riadené učenie** (angl. *supervised learning*) je v súčasnosti najpoužívanejším typom učenia, v oblasti hlbokého učenia dominuje. Toto učenie požaduje, aby pre každú vzorku v dátovej množine bola známa prislúchajúca značka. Použitím riadeného učenia je model upravovaný tak, aby čo najpresnejšie mapoval vstupné vzorky na očakávané výstupné značky. Príkladom úloh používajúcich riadené učenie sú klasifikácia a regresia.

(b) **Neriadené učenie** (angl. *unsupervised learning*) hľadá vhodné transformácie vstupných vzoriek pre odhalenie vlastností dátovej množiny a skrytých závislostí medzi vstupnými vzorkami. Tento prístup učenia nepoužíva pri spracovaní dát výstupné značky ani jednej z dostupných vzoriek. Preto je neriadené učenie vhodné pre neoznačkované trénovacie množiny. Úlohy kompresie, odstránenia šumu, redukcie priestoru a zhlukovania sú príkladmi neriadeného učenia.

(c) **Čiastočne riadené učenie** (angl. *semi-supervised learning*) predpokladá, že sú súčasne dostupné malá množina označkovaných vzoriek a veľká množina neoznačkovaných vzoriek. Ide o kombináciu oboch prístupov – riadeného aj neriadeného učenia. Neoznačkované vzorky sú použité na odhalenie vhodných reprezentácií, ktoré zvýrazňujú odlišnosti medzi skupinami podobných vzoriek s cieľom zoskupiť podobné vzorky do klastrov. Označkované vzorky sú použité na klasifikáciu klastrov, t. j. priradenie triedy každému klastru. Tento prístup učenia sa líši od použitia dvoch samostatných modelov, kde každý model používa jeden konkrétny typ učenia. Parametre modelu v prípade čiastočne riadeného učenia sú zdieľané počas oboch prístupov učenia.

(d) **Učenie odmeňovaním** (angl. *reinforcement learning*) spočíva v opakovaní operácie "pokus-omyl". Pri vzniku udalosti sa model rozhoduje a vyberá jednu z povolených akcií. Spätnou väzbou vo forme odmeny upravuje model svoje správanie tak, aby v budúcnosti dosiahol vyššiu odmenu, napr. zmenou akcie, ktorú vykoná pri opakovanom výskyte danej udalosti.

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**. Zdôvodnenie tejto voľby je uvedené v nasledujúcej časti predkladanej práce.

Trénovanie konvolučnej siete s použitím trénovacej dátovej množiny, ktoré nastaví parametre modelu pre zvolenú úlohu, prebieha len raz. Inými slovami, po získaní parametrov, pre ktoré sa už model správa podľa požiadaviek, proces trénovania končí. Preto sa pre túto operáciu používajú technické prostriedky s maximálnym výkonom bez ohľadu na ich spotrebu. Príkladom takéhoto výpočtového prostriedku je grafická procesorová jednotka (GPGPU), ktorá vykazuje väčší výkon než ostatné dostupné platformy. Ale vzhľadom na trendy hlbokého učenia, GPGPU neumožňuje efektívne využiť svoje zdroje pre implementáciu súčasných modelov hlbokého učenia a to kvôli pevnej architektúre s obmedzenou operačnou pamäťou, ktorá vyžaduje intenzívnu výmenu dát s GPGPU. Okrem toho, z pohľadu spotreby umožňujú iné z uvedených platforiem (FPGA, ASIC) lepšie využiť energiu pre vykonanie výpočtových operácií [89]. Na druhej strane, 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 v práci považujeme za efektívne riešenie s minimálnou spotrebou energie. Všeobecným cieľom optimalizácie inferencie je nájsť vhodný pomer medzi **efektivitou** (počet operácií v pomere ku spotrebe), **výkonom** (oneskorenie a počet operácii za čas), a **presnosťou** výstupov [90].

Použitím klasických výpočtových prostriedkov (CPU, GPGPU) je optimalizácia inferencie konvolučnej siete do značnej miery obmedzená pevnou štruktúrou použitej platformy. Na druhej strane, vzhľadom na svoju štruktúru umožňuje FPGA aj návrh vlastnej používateľom definovanej architektúry, napr. nepravidelná paralelná architektúra, ktorá je vhodnejšia pre používateľom navrhnuté programové vybavenie, zodpovedajúce algoritmu konvolučnej siete [91].

CPU a GPGPU sú založené na princípoch Von Neumannovej architektúry, ktorá oddeľuje vstupno-výstupné, výpočtové a pamäťové zdroje. Pravidelný prenos dát medzi pamäťou a výpočtovými jednotkami v tejto architektúre spotrebuje najviac energie a času. Narozdiel od Von Neumannovej architektúry, 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. S použitím uvedených prostriedkov je možné spracovať veľký objem dát v reálnom čase aplikáciou vhodnej paralelnej architektúry výpočtového systému a následnej optimálnej organizácie prechodu dát týmto systémom. Oneskorenie implementovaných algoritmov, založených na prúdovom spracovaní dát (angl. *pipelined streaming*), je znížené vďaka priamemu prepojeniu funkčných blokov v prúde, t. j. výstupy z jedného bloku nie sú uložené do pamäte a opäť načítané ďalším blokom. 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 (paralelný výpočet 3D konvolúcie pozostávajúcej z viacerých 2D konvolúcií, a paralelný výpočet 2D konvolúcie pozostávajúcej z viacerých 1D konvolúcií) [92].

Návrh efektívnej varianty konvolučnej siete vyžaduje okrem trénovania parametrov modelu aj experimentovanie s jeho všeobecnou štruktúrou, konkrétne s hyper-parametrami, ktoré určujú jeho štruktúru. Obvody FPGA poskytujú takúto formu flexibility tým, že umožňujú nielen definovať vlastnú architektúru, ale ju aj opakovane testovať a upravovať na základe spätnej väzby [89]. Príkladom úprav sú vlastné dátové typy dané voľbou spôsobu numerického kódovania (aritmetické operácie s pevnou alebo pohyblivou rádovou čiarkou) a parametrizovateľnej bitovej šírky. Vhodnou kvantizáciou (znižovanie bitovej šírky príznakov a váh) a kompresiou modelu sa zásadne znižujú výpočtové a pamäťové nároky modelu pri zachovaní požadovanej presnosti výstupov. Návrh konvolučnej siete popísaný v [93] je príkladom modelu, ktorý používa výhradne binárne hodnoty.

Vo všeobecnosti je naším cieľom použiť platformu s vhodným pomerom medzi flexibilitou výpočtových a pamäťových prostriedkov a ich výkonom z pohľadu nami zvolenej úlohy – inferencie konvolučnej siete. Ide teda o efektívne využitie zdrojov, ktoré sú dostupné v rámci danej platformy technických prostriedkov, a jednoduchosť vývoja výpočtového systému s použitím zvolenej implementačnej platformy. Vzhľadom na výhody FPGA, uvedené v [89]–[92], zhrnuté v predchádzajúcich odstavcoch a uplatniteľné pre našu úlohu, považujeme túto platformu za vhodnú pre riešenie zadanej úlohy. FPGA nám umožňuje aplikovať rôzne návrhové metódy, ktorými môžeme dosiahnuť zvýšenie výkonu navrhovaného systému. V ďalších podkapitolách uvádzame stručný popis nami zvolených návrhových techník.

3.5.2 Prúdové spracovanie dát

Prúdové spracovanie dát (angl. *pipelining*) vyžaduje špecifickú koncepciu architektúry číslicového systému, ktorá umožňuje podstatným spôsobom zvýšiť jeho výpočtový výkon [94]. Základom techniky je **rozloženie zložitej operácie na viacero jednoduchších operácií**, ktoré sú vykonávané súčasne rôznymi výpočtovými prvkami, a umožňuje tak vynechať predspracovanie dát súvisiace s ich pravidelným preusporiadaním v pamäti.

V prípade kombinačného obvodu sa nesmie vstupný signál meniť, pokiaľ ešte nie je ustálený príslušný výstupný signál (viď. obr. 3.3a). V takom prípade, ak obvod pozostáva z mnohých za sebou pripojených výpočtových jednotiek, sú tieto jednotky aktivované postupne. Inými slovami, v danom okamihu pracuje z pohľadu efektívneho spracovania práve jedna výpočtová jednotka (viď. obr. 3.3b). V prípade obvodu FPGA, ktorého primárnou silnou stránkou je možnosť paralelného spracovania a používania mnohých výpočtových zdrojov súčasne, je takýto návrh systému neefektívny.

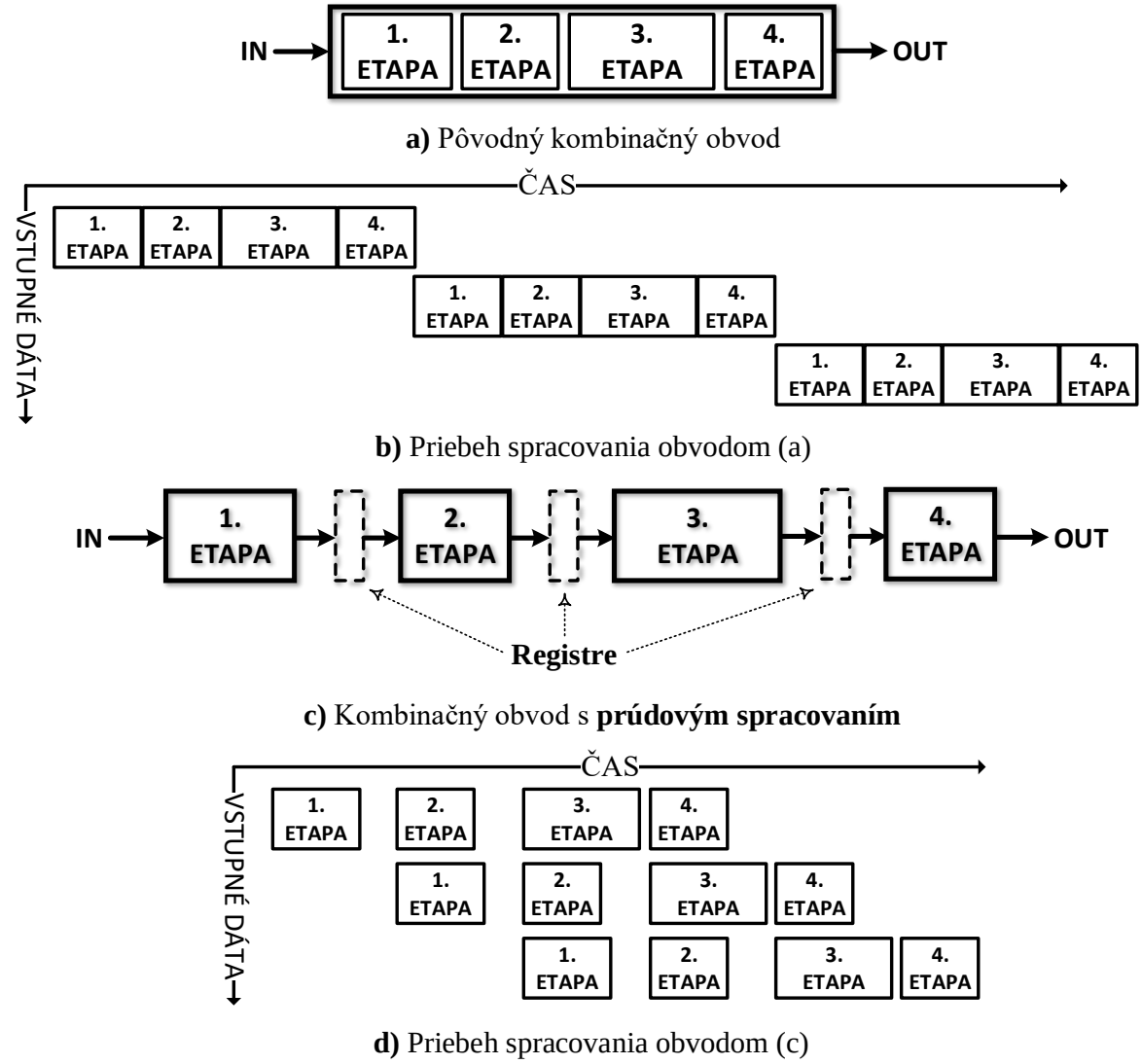
Aplikácia techniky prúdového spracovania predstavuje vloženie **registrov** do vhodných pozícií číslicového systému. Pridaním registrov sa spracovanie rozloží na viacero etáp s možnosťou paralelného vykonávania každej etapy (viď. obr. 3.3c). Úlohou registrov je odpamätávanie čiastočných výsledkov tak, aby sa dáta, použité ako vstupy do jednotlivých úrovní, menili vždy v tom istom okamihu (synchronizácia procesu je dosiahnutá použitím hodinového signálu). Vďaka registrom sa stávajú výpočtové bloky z rôznych etáp vzájomne nezávislými a pracujú tak súčasne nad rôznymi vstupnými dátami (viď. obr. 3.3d). Latentné oneskorenie platných výsledkov výpočtu je určené najdlhšou cestou prechodu dát výpočtovým systémom. Preto je cieľom prúdového spracovania **rozložiť systém na približne rovnako dlhé etapy** a prípadne preusporiadať spracovanie výpočtovými jednotkami pre dosiahnutie optimálneho výkonu.

Oneskorenie (angl. *delay* - celkový čas medzi prijatím konkrétnej vstupnej vzorky a odovzdaním prislúchajúceho výsledku) sa nasadením prúdového spracovania zvýši o latentné oneskorenie spôsobené vloženými registrami. Na druhej strane, **priechodnosť** systému (počet spracovaných vstupných vzoriek za jednotku času) sa zvýši vďaka vyššej taktovacej frekvencii hodinového signálu.

Podľa [94] sú vhodnými kandidátmi na efektívne použitie prúdového spracovania systémy, pre ktoré platí:

- Systém má na vstupe stále dostupné dáta pripravené na spracovanie.
- Hlavným kritériom výkonu systému je jeho priechodnosť (angl. *throughput*).
- Systém je možné rozložiť na etapy s približne rovnakými oneskoreniami.
- Oneskorenie etapy je oveľa väčšie než oneskorenie spôsobené registrom.

V praxi ide prevažne o systémy, ktorých spracovanie je možné vyjadriť ako reťaz za sebou umiestnených funkčných blokov. Príkladom sú oblasti číslicového spracovania signálov, spracovanie obrazu alebo sieťovej prevádzky a pod.



a) Pôvodný kombinačný obvod

b) Priebeh spracovania obvodom (a)

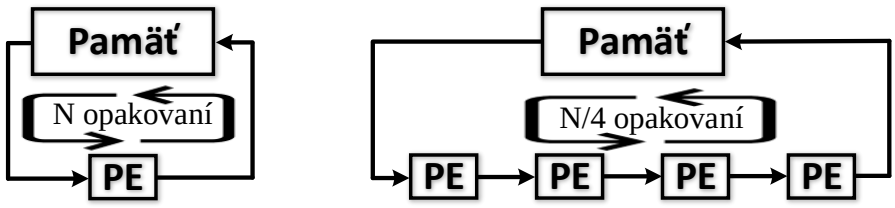
c) Kombinačný obvod s **prúdovým spracovaním**

d) Priebeh spracovania obvodom (c)

Obr. 3.3: Porovnanie pôvodného obvodu (a) a obvodu s prúdovým spracovaním (c) (Adaptované podľa zdroja: [94])

3.5.3 Systolické polia

Ďalšou z techník návrhu výpočtových systémov so zameraním na dosiahnutie maximálneho výkonu sú **systolické polia** [95], [96]. Architektúry, navrhnuté podľa princípov tohto prístupu, umožňujú prirodzene vykonávať **masívne paralelné výpočty** a **prúdové spracovanie dát** (kap. 3.5.2), ktoré zvyšujú celkový výkon systému. Princípy systolických polí sú použiteľné na problémy, ktorých riešenia vykazujú *jednoduchosť* a *pravidelnosť* operácií pravidelne opakujúcich sa výpočtových vzorov. Ilustračné schémy sekvenčného



a) Sekvenčné vykonávanie **b)** Systolické pole

Obr. 3.4: Schéma sekvenčného prístupu (a) a systolických polí (b) (Adaptované podľa zdroja: [95])

prístupu a systolických polí sú uvedené na obr. 3.4.

Z pohľadu systolických polí pozostáva návrh modulárneho systému z množiny základných buniek – **výpočtových prvkov** (angl. *processing elements*; PE) – usporiadaných do pravidelnej konfigurácie, napr. reťaz, mriežka, alebo strom. Jednotky PE sú vzájomne prepojené sieťou spojov. Táto sieť riadi a usmerňuje dátový tok medzi prvkami PE korektným spôsobom a tak zabezpečuje ich vzájomnú synchronizáciu. Každý vstupný údaj prúdi postupne sieťou, pričom každým prechodom cez PE je spracovaný vykonaním určitej operácie v závislosti od aktuálnej definície správania PE (závisí od úlohy a návrhu štruktúry PE). Takto sa medzivýsledky operácií šíria sieťou od vstupu až na výstup a postupne sa spájajú dokopy, takže sa na výstupe objaví už požadovaný finálny výsledok.

Vďaka škálovateľnosti a flexibilite, ktorú prináša modularita návrhu, je možné nasadiť systém, vytvorený pre riešenie konkrétneho problému, aj na väčšie problémy podobného charakteru. To často vyžaduje len malé zmeny bez nutnosti začínať s návrhom od začiatku. Aplikácia systolických polí je efektívna najmä v prípade výpočtovo náročných úloh, ktoré vyžadujú vykonať nad každou vstupnou vzorkou viacero numerických operácií. Systolické polia maximalizujú využitie každej vstupnej vzorky v systéme práve vďaka svojej štruktúre.

Princípy systolických polí poskytujú možnosť realizovať rôzne modely, ktoré sa líšia v spôsobe usporiadania a vzájomného prepojenia prvkov PE. Článok [95] vysvetľuje pôvod a základy systolických polí spolu s ich výhodami. Súčasťou článku sú príklady rôznych modelov konvolútora. Konvolúcia je jednou zo základných úloh číslicového spracovania signálov, ktorá pomocou základných numerických operácií (súčet a súčin) spája dva samostatné vstupné dátové toky (vstup a koeficienty filtra).

Na druhej strane, nie všetky systémy je možné efektívne navrhnúť s použitím systolických polí. Systém, vhodný pre systolické polia, musí spĺňať viaceré požiadavky [95]:

- **viacnásobné spracovanie vstupných dát** s použitím množiny rôznych operácií,
- **súbežné výpočty**, ktoré povoľujú paralelné a prúdové spracovanie,

- **jednoduchá špecifikácia** a tvar základného výpočtového prvku PE,
- **pravidelné usporiadanie** a tvar siete.

Konvolúcia je vhodným vzorovým príkladom aplikácie systolických polí, keďže spĺňa všetky uvedené vlastnosti. Vzhľadom na zameranie predkladanej práce je z nášho pohľadu vhodný návrh, ktorý je v článku označený ako *W2*. Tento model umožňuje s použitím prúdového spracovania permanentný beh všetkých prvkov PE a súčasne poskytuje súvislý tok výstupných dát. Detailný popis nášho prístupu k návrhu a implementácii inferencie konvolučnej siete sa nachádza v kap. 4.

Kapitola 4

Architektúra subsystémov konvolučnej siete

V tejto kapitole je detailnejšie popísaný návrh architektúry výpočtových subsystémov konvolučnej neurónovej siete (odpovedá zvolenej etape na obr. 3.1) s cieľom aplikácie FPGA obvodov. Vzhľadom na modulárnu štruktúru konvolučnej siete, popísanej v kap. 1.3, navrhnutá architektúra pozostáva z viacerých funkčne odlišných blokov, odpovedajúcich rôznym typom vrstiev. Z vrstiev boli vybrané: **konvolučná**, **zlučovacia**, **plne-prepojená** a **aktivačná** vrstva. Návrhy jednotlivých blokov sú uvedené v nasledujúcich podkapitolách.

4.1 Subsystém konvolučnej vrstvy

Vzhľadom na výpočtovú náročnosť konvolučnej vrstvy v konvolučnej sieti má jej akcelerácia vplyv na celkový výkon systému, teda je opodstatnená. Z tohto dôvodu sa kapitola zaoberá najmä návrhom subsystému konvolučnej vrstvy. V kontexte konvolučnej vrstvy, aplikáciou rôznych váhových matíc na vstupné mapy sú generované separátne výstupy, dané operáciou 2D konvolúcie. Teda vychádzajúc z popisu konvolučnej vrstvy v kap. 1.3.1 je táto vrstva vyjadrená ako súčet hodnôt, získaných z viacerých operácií 2D konvolúcie. Pre získanie požadovanej výstupnej mapy je potrebné následne spojiť tieto čiastočne výstupy dokopy ich sčítaním. Z dôvodov jednoduchosti a efektivity sme pre túto operáciu použili súčtový strom. Popis a schéma súčtového stromu sú uvedené v podkapitole subsystému plne-prepojenej vrstvy.

V návrhu modelu 2D konvolútora aplikujeme rôzne optimalizačné návrhové techniky, popísané v kap. 3.5, s cieľom dosiahnuť efektívnosť výpočtov. Aby sme mohli použiť tieto techniky (prúdové spracovanie a systolické polia), predpokladáme špecifický formát vstupných signálov, ktorých vzorky postupne prúdia systémom a tvoria jeden

súvislý tok. S postupnosťou za sebou idúcich vzoriek následne pracujeme ako s vektorom. Dôvodom je spôsob, akým pracujú techniky prúdového spracovania dát a systolických polí. Avšak, v prípade 2D konvolúcie prebiehajú numerické operácie nad maticami. Z tohto dôvodu, model vyžaduje rozloženie maticových operácií na vektorové, preusporiadanie medzivýsledkov, a ich následné zoskupenie do finálnych výstupov. Preto je požadovaná zmena pohľadu na 2D konvolúciu a úprava spôsobu, akým vo všeobecnosti prebieha výpočet 2D konvolúcie.

V článkoch [97]–[101] sú uvedené rôzne prístupy a návrhy systémov so zameraním na optimálny výpočet 2D konvolúcie. Všeobecná myšlienka prúdenia vstupných signálov výpočtovými prvkami je pre všetky z týchto modelov rovnaká; modely sa odlišujú len v štruktúre a prepojení základných výpočtových prvkov.

Vzhľadom na popis 2D konvolúcie (kap. 1.3.1) a prislúchajúci matematický vzťah (1.10), je možné rozdeliť jej výpočet do viacerých krokov. V každom kroku prebieha výpočet jedného z vnútorných súčtov v uvedenom vzťahu. Inými slovami, 2D konvolúcia je rozložiteľná na skupinu 1D konvolúcii, ktoré používajú len vektorové operácie a predstavujú všeobecný v súčasnosti už vyriešený problém číslicového spracovania signálov. Vďaka tomu môžeme aplikovať prístupy všeobecne dostupných riešení. Pred popisom štruktúry a správania nami navrhovanej architektúry uvedieme v nasledujúcom odseku predpoklady o formáte dátových vstupov a výstupov systému. Tieto predpoklady určujú celkový návrh 2D konvolútora.

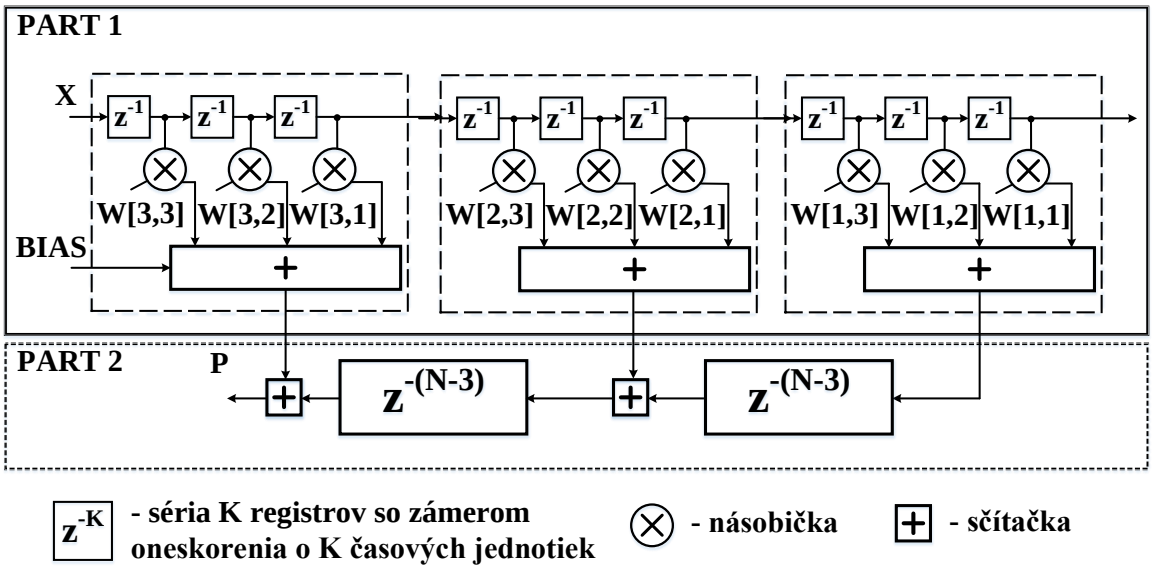
Predpokladáme, že vstupná príznaková mapa má pravouhlý tvar s rovnakou konštantnou výškou a šírkou. V tejto práci je veľkosť vstupnej mapy vyjadrená parametrom N. Rovnaký predpoklad o konštantnej vopred definovanej šírke platí aj pre maticu koeficientov (angl. *kernel*). Šírka matice koeficientov je 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. V prípade zmien parametrov N a K je potrebná aj zmena samotného modelu a jeho komponentov. Na druhej strane, požadované zmeny sú priamočiare vzhľadom na zvolený prístup systolických polí, a vyžadujú len pridanie ďalších výpočtových prvkov.

4.1.1 Architektúra modelu

Navrhovaný model predstavuje štandardný synchrónny obvod, ktorý pozostáva z dátovej a riadiacej štruktúry. Popis modelu a jeho štruktúr je rozdelený na niekoľko častí. Najprv uvádzame popis architektúry modelu vrátane detailného popisu jeho komponentov. V ďalších podkapitolách sa následne venujeme funkcii jednotlivých komponentov a uvádzame prínos nášho návrhu.

Popis dátovej štruktúry

Dátová štruktúra pozostáva z dvoch častí, ktoré v práci označujeme **PART1** a **PART2**. Ich model a jeho adaptovaná verzia sú zobrazené na obr. 4.1 a obr. 4.2.



Obr. 4.1: Model dátovej štruktúry 2D konvolútora (Zdroj: [96]). V prvej časti (PART 1), vstupná vzorka $\boldsymbol{X}$ prechádza postupne radom registrov. Výstupy z registrov predstavujú vstupné dáta v po sebe idúcich časových okamihoch – $\boldsymbol{X_i}, \boldsymbol{X_{i-1}}, \boldsymbol{X_{i-2}}, \ldots, \boldsymbol{X_{i-9}}$. Medzivýsledky sú vypočítané vynásobením vstupných vzoriek s váhovými koeficientami $\boldsymbol{W}\,[\boldsymbol{3,3}]\,, \boldsymbol{W}\,[\boldsymbol{3,2}]\,, \ldots, \boldsymbol{W}\,[\boldsymbol{1,1}]$ a následným sčítaním súčinov po blokoch. K prvému medzivýsledku je pripočítaný bias vstupnej vzorky. V druhej časti (PART 2) sú medzivýsledky rôzne oneskorené a nakoniec sčítané do finálneho výsledku $\boldsymbol{P}$, ktorý je výstupom z 2D konvolútora.

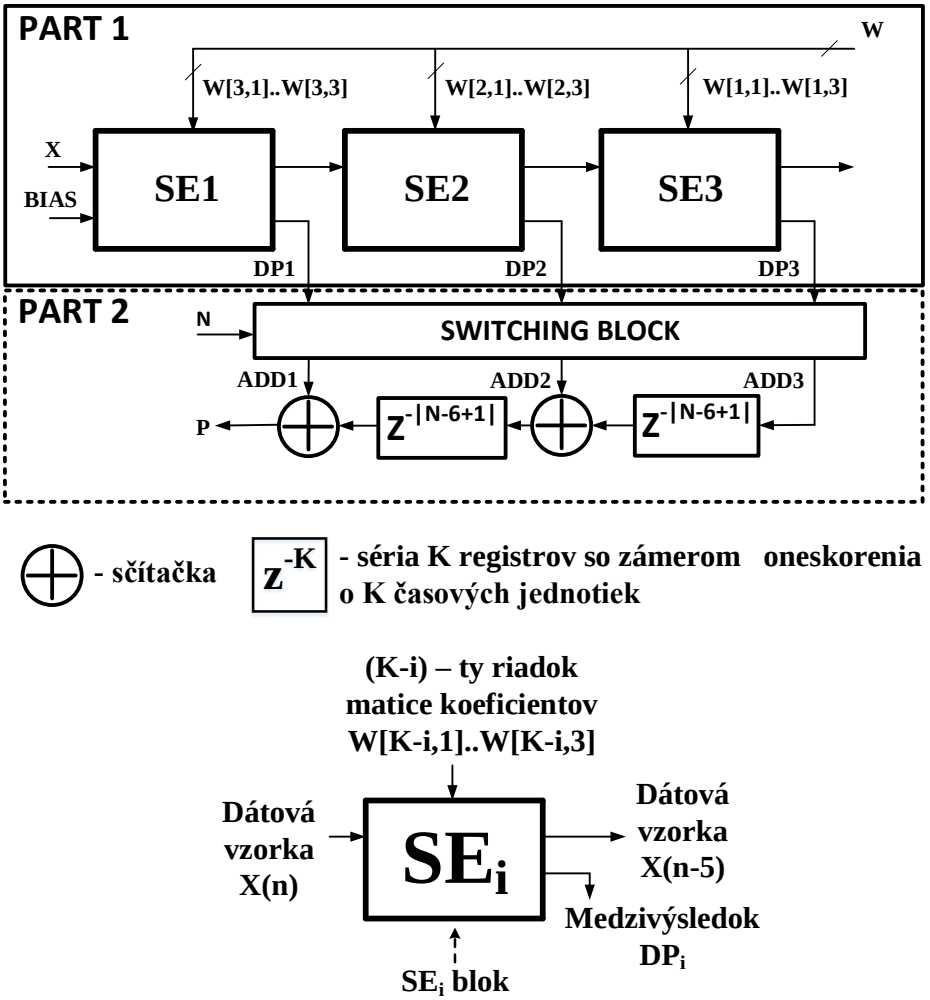
Úlohou prvej časti (*PART1*) je výpočet jednotlivých vnútorných súčtov. Ide o štandardnú úlohu z oblasti DSP, ktorá pozostáva z opakovaného násobenia dvojíc – vstupná vzorka, koeficient – a následného sčítania súčinov (angl. *Multiply-And-Accumulate*; MAC). Výstupmi z tejto časti modelu sú čiastočné skalárne súčiny DP_i (4.1) a (4.2),

$$\mathrm{DP}_1 = \mathrm{BIAS} + \sum_{j=1}^{K} \boldsymbol{X}\,(\boldsymbol{n-j+1}) \times \boldsymbol{W}\,[\boldsymbol{K,j}] \tag{4.1}$$

$$\mathrm{DP}_{i=2\ldots K} = \sum_{j=1}^{K} \boldsymbol{X}\,(\boldsymbol{n-j+1}) \times \boldsymbol{W}\,[\boldsymbol{K-i+1,j}] \tag{4.2}$$

ktoré musia byť vhodne spojené do finálneho výsledku ($\boldsymbol{P}$) (4.3).

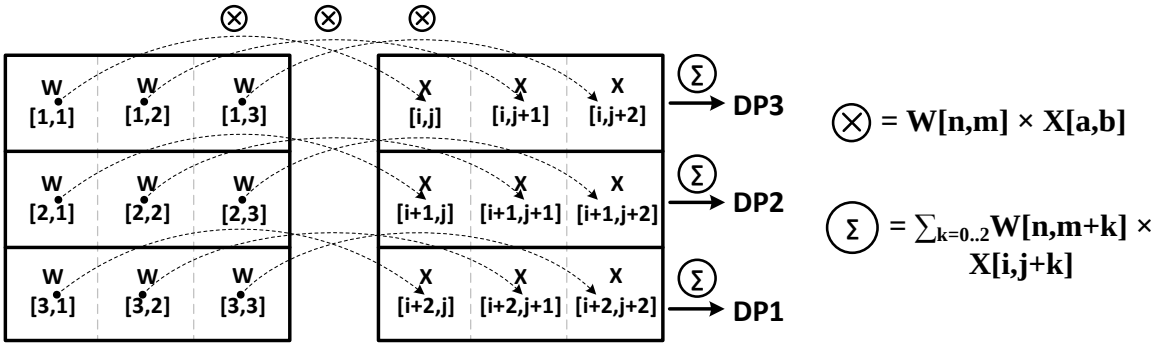
$$\boldsymbol{P} = \sum_{i=1}^{K} \mathrm{DP}_i \tag{4.3}$$



Obr. 4.2: Adaptovaný model dátovej štruktúry 2D konvolútora so zámerom následnej efektívnej implementácie do FPGA obvodu (Zdroj: [96]). 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**.

Z pohľadu 2D konvolúcie predstavuje DP_i skalárny súčin i-tého riadku vstupného okna a i-tého riadku matice koeficientov (obr. 4.3). Ak je použitý *BIAS* s nenulovou hodnotou, je táto hodnota pripočítaná k DP_1 (4.1). Úlohou druhej časti (*PART2*) je spájanie medzivýsledkov a udržiavanie synchronizovaného stavu. Inými slovami, druhá časť určuje časovanie súvisiace so spájaním jednotlivých medzivýsledkov, operáciou sčítania, keďže

rôzne medzivýsledky pre výpočet jedného konkrétneho výstupu sú platné v rôznych časových intervaloch.



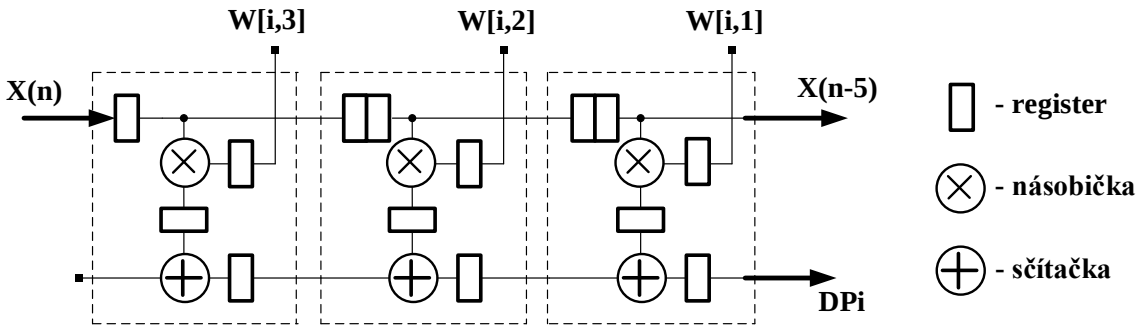
Obr. 4.3: Ilustračný pohľad na Hadamardov súčin vstupného okna **X** a matice koeficientov **W** (Zdroj: [96]). Každý koeficient $\boldsymbol{W}\,[\boldsymbol{i},\boldsymbol{j}]$ je vynásobený s prislúchajúcim prvkom vstupného okna $\boldsymbol{X}\,[\boldsymbol{i},\boldsymbol{j}]$. Súčet všetkých prvkov v **i**-tom riadku vo výslednej matici udáva hodnotu prislúchajúceho medzivýsledku **DPi**.

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 zo známych časovo-overených a odporúčaných metód [102], [103]. Uvedené zdroje popisujú rôzne modely výpočtového systému konvolúcie vhodné pre FPGA. Pôvodný model je priamočiary a zodpovedá systolickému poľu. Na druhej strane, adaptovaný model so zámerom následnej efektívnej implementácie 2D konvolútora do FPGA obvodu vykazuje odlišnosti spôsobené použitím DSP blokov a nasadením prúdového spracovania (model na obr. 4.2 používa pre prúdenie vstupných signálov systolickými prvkami až $\boldsymbol{2 \times K - 1}$ registrov namiesto K registrov, uvedených v pôvodnom modeli na obr. 4.1). Každý DSP blok v adaptovanom modeli vykonáva súčin konkrétnej vstupnej vzorky a koeficientu. Výsledný súčin je pripočítaný ku kumulatívnemu výstupu. Vychádzajúc z prúdového spracovania sú medzi operátory vložené registre, ktoré umožňujú zvýšenie maximálnej pracovnej frekvencie.

DSP bloky obsahujú subsystémy, ktoré umožňujú optimálnu implementáciu konvolútora. Vstavaná násobička a sčítačka podporujú symetrické zaokrúhľovanie a kvantizáciu výsledkov numerických operácií. Reťazovým vykonávaním numerických operácií násobenia a sčítania narastá efektívna bitová šírka signálov a hrozí pretečenie (angl. *overflow*). Použitím zaokrúhľovania a orezávania výstupov z numerických operácií je možné predísť pretečeniu a udržať tak validnosť výstupov. Kaskádové prepojenie interných dátových signálov susedných DSP blokov umožňuje rýchlo šíriť signál medzi DSP blokmi. Navyše, použitím vstavaných zdrojov dostupných v DSP blokoch – sčítačka, násobička, a registre (dočasná vyrovnávacia pamäť pre uchovanie vstupných signálov počas ich používania

v násobičkách), nespotrebujeme žiadne dodatočné zdroje v obvode FPGA. 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) [102]. Systolický konvolútor je všeobecne považovaný za optimálny model s paralelným spracovaním pre obvody FPGA. Latentné oneskorenie nemá žiadny zreteľný vplyv na výkon v porovnaní s modelom používajúcim súčtový strom. Kaskádový model zásadne znižuje spotrebu energie a zvyšuje rýchlosť spracovania tým, že odzrkadľuje pravidelné usporiadanie a priame prepojenia medzi DSP, BRAM, a CLB. Navyše, tento model je obmedzený len počtom dostupných DSP blokov v FPGA [103]. Ilustračný model systolického konvolútora s tromi koeficientami je zobrazený na obr. 4.4. Zdroje [102]–[104] poskytujú implementačné detaily vhodného nasadenia DSP blokov s cieľom zlepšiť celkový výkon a súčasne znížiť spotrebu výsledného návrhu.



Obr. 4.4: Model SE$_i$ bloku (Zdroj: [96]). $\boldsymbol{SE_i}$ 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 $\boldsymbol{W}$ $\boldsymbol{[i, 3]}$, $\boldsymbol{W}$ $\boldsymbol{[i, 2]}$ a $\boldsymbol{W}$ $\boldsymbol{[i, 1]}$. 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**.

Druhá časť dátovej štruktúry preusporiadava medzivýsledky s cieľom ich následného spojenia do požadovaného výstupu. Dôvodom pre takúto manipuláciu s medzivýsledkami je maticový tvar operandov numerických operácií v prípade 2D konvolúcie. Keďže matica nezodpovedá tvaru vstupných dátových vzoriek, prvá časť dátovej štruktúry nami navrhovaného modelu nahrádza maticové operácie postupnosťou vektorových operácií, ktorých

výstupy (v texte označované ako medzivýsledky) sú dostupné v rôznych časových okamihoch. Preusporiadanie je tak 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*), na obr. 4.1 a 4.2 zakreslené ako bloky medzi sčítačkami. Dĺžka oneskorenia je zvolená so zámerom zachovať platnosť výstupov a synchronizovaný stav modelu.

Ak predpokladáme, že vstupný tok dát vchádza do systému ako vektor usporiadaných dátových vzoriek, a v každom hodinovom takte sa po spracovaní tento vektor posunie o vzorku, tak časová medzera medzi medzivýsledkami toho istého výstupu (spracovanie toho istého okna) je rovná šírke vstupnej mapy N. Inými slovami, vzdialenosť medzi susednými medzivýsledkami toho istého výstupu je rovná N časových jednotiek. Navyše, prvá časť dátovej štruktúry už prináša oneskorenie $\boldsymbol{2 \times K - 1}$ časových jednotiek (viď. adaptovaný model na obr. 4.2). Odčítaním oneskorenia z prvej časti od celkového požadovaného oneskorenia sú tak oneskorovacie pamäte zodpovedné za zostávajúce oneskorenie dlhé $\boldsymbol{N - 2 \times K + 1}$ časových jednotiek.

Sériové zapojenie oneskorovacích pamätí v kombinácii so sčítačkami v druhej časti nahrádza súčtový strom – štandardný prístup používaný v súčasných metódach. Sériové usporiadanie prináša výhodu jednoduchej implementácie a efektívneho využitia času. Spájanie medzivýsledkov sčítavaním prebieha zatiaľ, čo čakáme na výpočet chýbajúcich medzivýsledkov práve spracovávaného okna. Vďaka tomu nie je spôsobené žiadne ďalšie oneskorenie v porovnaní s alternatívami, používajúcimi súčtový strom.

Popis riadiacej štruktúry

Dátová štruktúra, popísaná v predchádzajúcej podkapitole, je riadená signálmi, ktoré určujú validitu výstupného signálu (výstupné príznaky) v závislosti od validity vstupného signálu (vstupné dátové vzorky). Riadiaca štruktúra je realizovaná konečným stavovým automatom (angl. *Finite State Machine*; FSM), navrhnutým konkrétne pre náš model. 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, ako aj o konštantnej dĺžke neplatných intervalov pre pravidelné nastavovanie a sledovanie použitých 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.

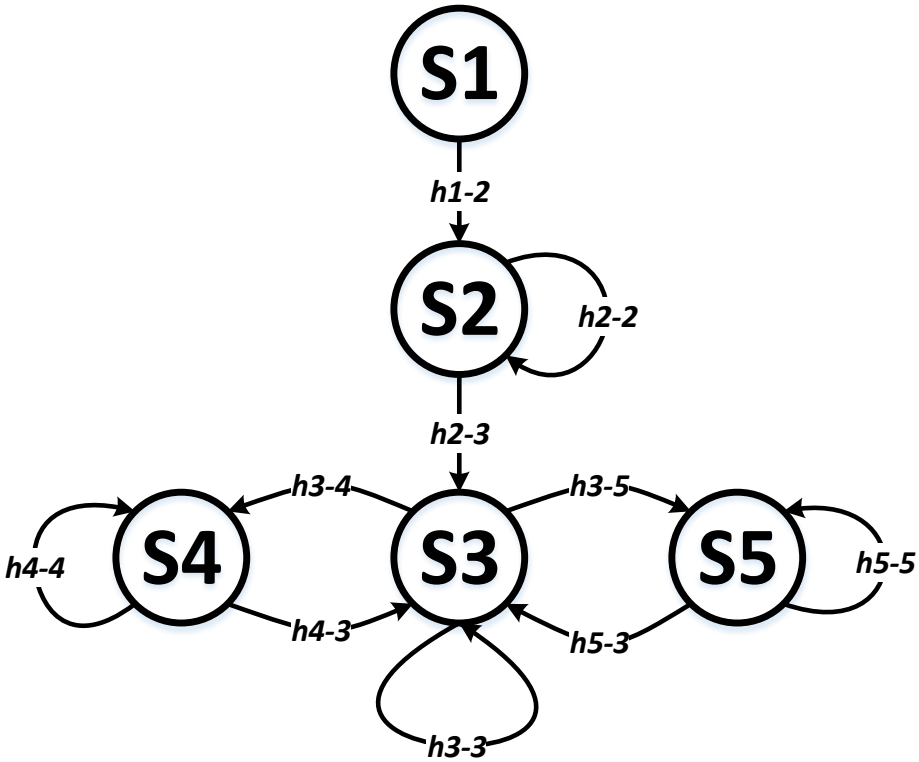
FSM rozlišuje tri prípady prechodu okna vstupnou mapou:

(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 druhého (b) a tretieho (c) z vymenovaných prípadov sú považované za neplatné a musia byť ignorované v ďalšom spracovaní. Takže, len výsledky získané počas prechodu okna vo vnútri vstupnej mapy (a) sú platné. Dôvod neplatnosti je popísaný neskôr v tejto kapitole. Stavový diagram FSM je uvedený na obr. 4.5. Diagram pozostáva z piatich stavov, ktoré mapujú okrem troch vyššie uvedených prípadov ešte fázy inicializácie (init) a rozbehnutia (startup). Riadiaca časť implementuje dve počítadlá s dynamickou prahovou hodnotou – *PIXEL_COUNTER* a *LINE_COUNTER*. Každé z počítadiel je zodpovedné za konkrétny prechod medzi stavmi. Aktuálny model FSM pracuje len, ak vstupný signál indikuje platnosť vstupných vzoriek. V opačnom prípade, FSM čaká v poslednom nadobudnutom stave bez zmeny atribútov (aktuálny stav interných počítadiel). Voľba tohto správania zjednodušuje návrh FSM a súčasne ovplyvňuje funkciu dátovej štruktúry, ktorá je počas neplatných vstupov neaktívna.



Obr. 4.5: Stavový diagram FSM 2D konvolútora. Riadiaca časť (FSM) pozostáva z piatich stavov *S1*, . . . , *S5* (tab. 4.2). Hrany medzi stavmi reprezentujú podmienené prechody *h1-2*, . . . , *h5-5*, kde prvá cifra označuje číslo zdrojového stavu a druhá cifra označuje číslo cieľového stavu (tab. 4.3).

Tabuľka 4.1: **Prednastavené hodnoty riadiacich signálov a výstupov pre všetky stavy FSM 2D konvolútora** (obr. 4.5). Signály sú nastavené na hodnoty v tabuľke vždy na začiatku procesu v zdrojovom kóde jazyka VHDL. V niektorých stavoch a prechodoch dochádza k ich zmene (tab. 4.2 a tab. 4.3).

Názov	Hodnota
ready	0
valid_out	0
pixel_counter_clear	0
pixel_counter_set	0
pixel_counter_threshold	0
line_counter_clear	0
line_counter_set	0
line_counter_ce	0
line_counter_threshold	0

Tabuľka 4.2: **Prehľad stavov v automate riadiacej časti FSM 2D konvolútora** (obr. 4.5). Nastavenia Moorovych výstupov sú v tabuľke zapísané vo formáte priraďovacieho príkazu jazyka VHDL. Parameter *STARTUP_DELAY* udáva požadovaný počet prijatých platných bodov vstupnej mapy pred odovzdaním prvého platného bodu výstupnej mapy – ide o naplnenie reťaze registrov vo všetkých SE blokoch platnými vstupnými dátami.

Stav	Názov	Moorove výstupy (odlišné od tab. 4.1)
S1	Inicializačný stav (INIT)	pixel_counter_threshold ⇐ STARTUP_DELAY; pixel_counter_set ⇐ '1';
S2	Začiatok výpočtov (STARTUP)	
S3	Vo vnútri mapy (INSIDE_IMAGE)	valid_out ⇐ '1';
S4	Vertikálna hrana (VERTICAL_BORDER)	
S5	Horizontálna hrana (HORIZONTAL_BORDER)	

4.1.2 Funkcia modelu

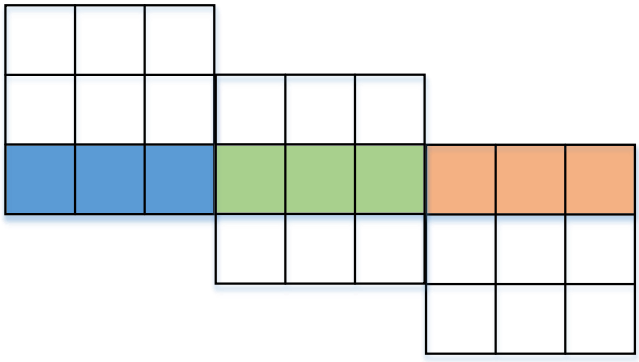
Primárna funkcia modelu je daná správaním jeho komponentov a ich vzájomným prepojením. V prípade nášho modelu sú podstatné komponenty: séria blokov SE v *PART1*, prepínací blok (angl. *switching block*) a oneskorovacie pamäte v *PART2* (obr. 4.2). Ich správanie je popísané v tejto kapitole. Pre interpretáciu sme použili vizuálny pohľad na spôsob, akým prebieha spracovanie vstupnej mapy použitím 2D konvolúcie (kap. 1.3.1). Výpočet výstupných príznakov 2D konvolúciou spočíva v skenovaní vstupnej príznakovej mapy technikou posuvných okien. Vzhľadom na tento proces je jednou z možností vizualizácie série SE blokov ich zobrazenie vo forme vzoru, ktorý pozostáva z K okien a je ilustrovaný na obr. 4.6.

Každé okno vzoru je rozdelené do K riadkov, pričom v rámci okna je označený

Tabuľka 4.3: **Prehľad prechodov v automate riadiacej časti FSM 2D konvolútora** (hrany na obr. 4.5). Prechod medzi stavmi sa vykoná len, ak sú splnené všetky požadované podmienky v stĺpci **Podmienky prechodu** a aktuálny vstupný bod je platný (*din_valid = '1'*). Nastavenia Mealyho výstupov ([94], kap. 10) sú v tabuľke zapísané vo formáte priraďovacieho príkazu jazyka VHDL. Parametre *NO_VALID_PIXELS_PER_LINE*, *NO_VALID_LINES_PER_IMAGE*, *NO_INVALID_PIXELS_PER_LINE* a *NO_INVALID_PIXELS_PER_TRANSITION* závisia od šírky vstupnej mapy N a šírky filtra K, a udávajú rozmery platných a neplatných oblastí (obr. 4.9).

Hrana	Podmienky prechodu	Mealyho výstupy
h1-2		
h2-2	pixel_counter_alert = '0'	
h2-3	pixel_counter_alert = '1'	pixel_counter_threshold ⇐ NO_VALID_PIXELS_PER_LINE; pixel_counter_set ⇐ '1'; line_counter_threshold ⇐ NO_VALID_LINES_PER_IMAGE; line_counter_set ⇐ '1';
h3-3	pixel_counter_alert = '0' line_counter_alert = '0'	
h3-4	pixel_counter_alert = '1' line_counter_alert = '0'	pixel_counter_threshold ⇐ NO_INVALID_PIXELS_PER_LINE; pixel_counter_set ⇐ '1';
h3-5	line_counter_alert = '1'	pixel_counter_threshold ⇐ NO_INVALID_PIXELS_PER_TRANSITION; pixel_counter_set ⇐ '1'; line_counter_clear ⇐ '1';
h4-3	pixel_counter_alert = '1'	pixel_counter_threshold ⇐ NO_VALID_PIXELS_PER_LINE; pixel_counter_set ⇐ '1'; line_counter_ce ⇐ '1';
h4-4	pixel_counter_alert = '0'	
h5-3	pixel_counter_alert = '1'	pixel_counter_threshold ⇐ NO_VALID_PIXELS_PER_LINE; pixel_counter_set ⇐ '1';
h5-5	pixel_counter_alert = '0'	

práve jeden riadok. Tvar vzoru závisí od rozloženia koeficientov v matici medzi jednotlivé SE bloky. Okná vzoru prislúchajú jednotlivým SE blokom. Každý SE blok generuje medzivýsledok (DP_i), ktorý zodpovedá farebnému riadku okna. Inými slovami, konkrétne okno vzoru zafarbuje vždy rovnaký riadok všetkých okien vo vstupnej mape. Okno vstupnej mapy považujeme za úplne spracované, ak bolo pokryté každým z okien vzoru pre spracovanie všetkých jeho riadkov ($DP_{i=1...K}$). Pod funkciou vzoru, reprezentujúceho sériu SE blokov, si môžeme predstaviť farbenie jednotlivých riadkov okien vstupnej mapy. Spracovanie okna je úplné až, keď sú všetky jeho riadky zafarbené, t. j.



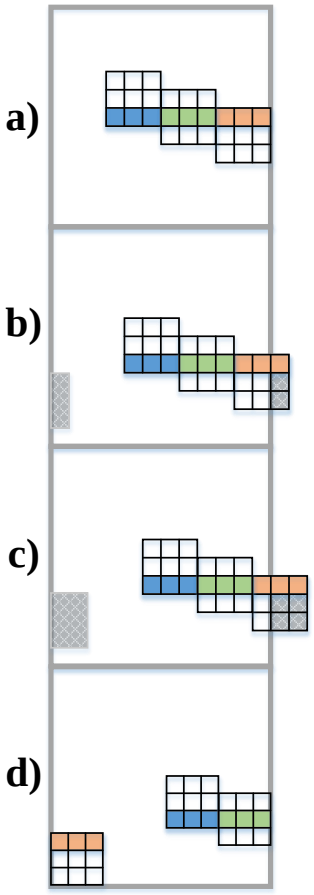
Obr. 4.6: Vzor reprezentujúci sériové zapojenie SE blokov. V každom okne je zvýraznený práve jeden riadok. Okno vykonáva spracovanie len tohto riadku.

keď cez okno prešli všetky okná vzoru. Keďže vizuálny tvar vzoru určuje rôzne pozície okien (okná vzoru sú vzájomne posunuté vždy o riadok), medzivýsledky získané z SE blokov v danom časovom okamihu zodpovedajú rôznym oknám vstupnej mapy. Z tohto dôvodu, nie je možné okamžite skombinovať tieto medzivýsledky. Požadované preusporiadanie medzivýsledkov prináša požiadavku na ich synchronizáciu vykonanú vhodne umiestnenými oneskorovacími pamäťami. Sčítačky, umiestnené medzi pamäte, sumarizujú všetky súvisiace medzivýsledky do finálneho výstupu.

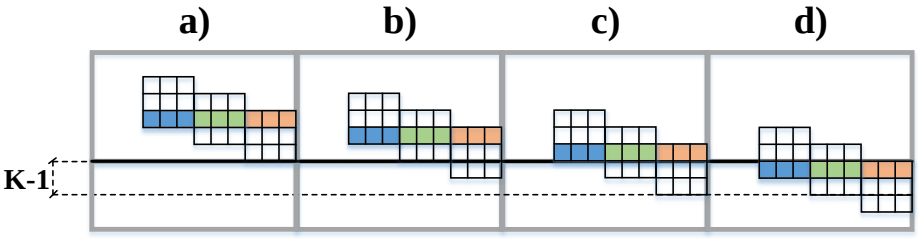
Vzhľadom na tvar vzoru, jeho okná prechádzajú vstupnou mapou sekvenčne v poradí zľava doprava a zhora nadol. Medzera medzi dvoma susednými oknami mapy je $\boldsymbol{N - K}$ časových jednotiek vychádzajúc zo spôsobu prechodu vzoru mapou riadok-po-riadku. Medzivýsledky dvoch susedných okien vzoru, zobrané v časových okamihoch $\boldsymbol{t_i}$ a $\boldsymbol{t_i + N - K}$ prislúchajú k tomu istému výsledku. Použitím tohto princípu na všetkých K okien vzoru, medzivýsledky získané v časových okamihoch $\boldsymbol{t_i}$, $\boldsymbol{t_i + N - K}$, ..., $\boldsymbol{t_i + (N - K) \times (K - 1)}$ tvoria jeden výstupný príznak. Tento časový posun je dôvodom použitia oneskorovacích pamätí. Vychádzajúc so sériového zapojenia pamätí v pôvodnom modeli (obr. 4.1) je dĺžka každej pamäte rovná $\boldsymbol{N - K}$.

Vzor, uvedený na obr. 4.6, sekvenčne prechádza vstupnou mapou riadok-po-riadku z dôvodu prúdového príchodu vstupných príznakov. Pravidelný pohyb vzoru v preddefinovanom smere (zľava doprava a zhora nadol) ovplyvňuje platnosť výstupov v závislosti od rôznych prípadov, uvedených v predchádzajúcej kap. 4.1.1. Počas prechodu vzoru vertikálnym okrajom vstupnej mapy sú sériou SE blokov spracované vstupné príznaky, ktoré ale tvoria nesúvislé a tak neplatné okno (obr. 4.7 - (b) a (c)).

Podobná situácia nastáva aj v prípade prechodu vzoru cez horizontálny okraj na nasledujúcu mapu. V tomto prípade, spracované okná obsahujú príznaky z oboch susedných máp a teda sú tiež neplatné (obr. 4.8 - (b) a (c)).



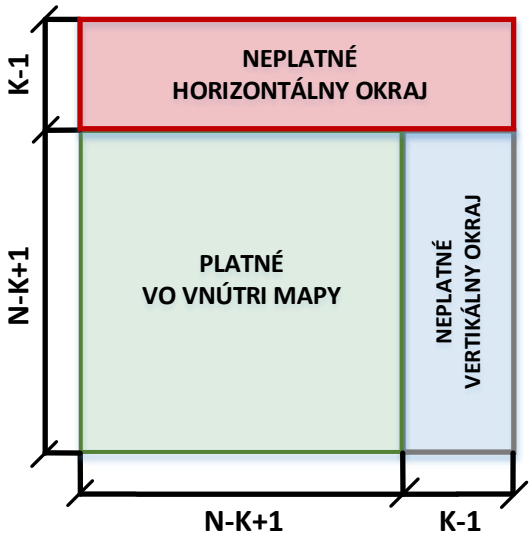
Obr. 4.7: Prechod vertikálnym okrajom vstupnej mapy. Posúvaním vzoru vstupnou mapou sa pri prechode na ďalší riadok objavujú neplatné okná. Výsledky z týchto okien sú taktiež neplatné a musia byť vynechané – prípady (b) a (c).



Obr. 4.8: Prechod horizontálnym okrajom vstupnej mapy. Posúvaním vzoru sa pri prechode na ďalšiu vstupnú mapu objavujú neplatné okná, ktoré obsahujú body z oboch vstupných máp. Výsledky z týchto okien sú taktiež neplatné a musia byť vynechané – prípady (b) a (c).

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. Ilustrácia na obr. 4.9 zobrazuje oblasti platnosti výstupov vrátane ich rozmerov, vyjadrených cez parametre N a K.



Obr. 4.9: Platné a neplatné oblasti. Šírka vstupnej mapy je daná parametrom N, šírka matice koeficientov (filtra) je daná parametrom K.

Funkcia SE bloku

Všetky numerické operácie modelu, súvisiace s 2D konvolúciou, vykonáva séria SE blokov so štruktúrou systolického poľa, zobrazenou na obr. 4.4. Štruktúra pozostáva z dvoch dátových tokov umiestnených nad sebou. Vrchný tok reprezentuje prúdenie vstupných dátových vzoriek SE blokom tak, aby boli dostupné pre jednotlivé násobičky. Spodný tok reprezentuje čiastkový súčet súčinov vstupných vzoriek s príslušnými koeficientami. Princíp požadovaného fungovania SE bloku spočíva v rôznej rýchlosti vrchného a spodného toku, ktorá je spôsobená pridaním ďalších registrov do vrchnej časti. Z tohto dôvodu, dátový tok vo vrchnej časti je spomalený na polovičnú rýchlosť v porovnaní s tokom v spodnej časti.

Keďže súčin aktuálnej vstupnej vzorky s koeficientom (úplne na ľavej strane štruktúry) predstavuje **posledný prvok** prislúchajúceho výstupu (ostatné prvky už tečú SE blokom), musí tento súčin "dobehnúť" súčiny viacerých predchádzajúcich vstupných vzoriek (konkrétny počet vzoriek závisí od počtu koeficientov). Táto požiadavka je zabezpečená dvojnásobnou rýchlosťou čiastkového súčtu. Na výstupe sa tak objaví vždy platný výsledok. Kvôli prúdovému spracovaniu je výpočet konvolúcie SE blokom oneskorený o K časových jednotiek (čas medzi odovzdaním požadovaného súčtu na výstupe a prijatím jeho poslednej vzorky na vstupe).

4.1.3 Popis adaptovaného modelu

Cieľom adaptácie pôvodného modelu (obr. 4.1) je zámer efektívnej implementácie 2D konvolútora do FPGA obvodu, ktorá vhodným spôsobom používa dostupné prvky štruktúry FPGA obvodu. Efektívna implementácia SE bloku do obvodu FPGA s použitím systolického poľa vyžaduje dodatočné registre vo vstupnom dátovom toku, ktoré nie sú zahrnuté v pôvodnom modeli. Celkové oneskorenie SE bloku je tak rovné $\boldsymbol{2 \times K - 1}$ namiesto $\mathbf{K}$ časových jednotiek, ako už bolo uvedené v kap. 4.1.1. Vychádzajúc z vyššie uvedeného popisu správania série SE blokov, je tak požadované oneskorenie medzi každou dvojicou susedných sčítačiek medzivýsledkov a dĺžka každej z vyrovnávacích pamätí upravená na hodnotu $\boldsymbol{N - 2 \times K + 1}$.

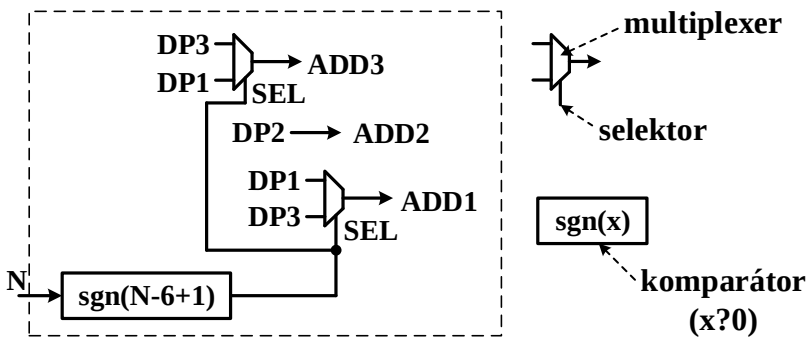
V prípade veľkej vstupnej mapy v porovnaní s rozmermi matice koeficientov, t. j. keď platí $\boldsymbol{N \geq 2 \times K - 1}$ je správanie pôvodného modelu v poriadku. Ale v prípade malej vstupnej mapy v porovnaní s rozmermi matice koeficientov, t. j. keď platí $\boldsymbol{N < 2 \times K - 1}$, je požadované oneskorenie **záporné**. Vyrovnávacia pamäť s negatívnou dĺžkou nie je realizovateľná. Riešením problému je pridanie pozitívneho oneskorenie pre všetky signály v modeli tak, aby sme eliminovali negatívne oneskorenie vyrovnávacích pamätí. Synchronizovaný stav modelu je zachovaný. Pre tento účel slúži prepínací blok, ktorý je vložený medzi sériu SE blokov a oneskorovacie pamäte.

Prepínací blok

Štruktúra druhej časti dátovej štruktúry (*PART2*) nám umožňuje využiť jej symetriu – všetky oneskorovacie pamäte majú rovnakú dĺžku a sú zapojené do série – pri návrhu prepínacieho bloku. 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 ($\mathrm{ADD}_{\mathbf{1}}$). Efektívna implementácia popísaného systému vychádza z použitia multiplexerov, riadených vhodným selektorom. Ide o signál určený znamienkom výrazu $\boldsymbol{N - 2 \times K + 1}$. Príklad prepínacieho bloku pre $\boldsymbol{K = 3}$ je uvedený na obr. 4.10.

4.1.4 Originálny prínos nášho návrhu

Nami navrhovaný model 2D konvolútora predstavuje originálnu štruktúru, pričom originalita návrhu spočíva v unikátnom umiestnení oneskorovacích pamätí. Štandardným umiestnením oneskorovacích pamätí je ich vloženie pred výpočtovú časť alebo častejšie používané vloženie pamätí do vnútra výpočtovej časti medzi jednotlivé výpočtové bloky ([105], kap. 8).



Obr. 4.10: 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 $\boldsymbol{N - 6 + 1}$ – $\boldsymbol{sgn\,(N - 6 + 1)}$. 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.

V prvom prípade je zámerom preusporiadanie vstupných vzoriek do formátu matice ešte predtým, ako sú tieto vzorky vložené do výpočtových blokov. Takýto prístup ale vyžaduje dodatočné použitie súčtových stromov, aby boli čiastkové súčiny následne sumarizované do výsledného súčtu, predstavujúceho finálny výsledok.

Druhý spôsob umiestnenia pamätí rieši nedostatok súčtového stromu tým, že medzivýsledky sú synchronizované a sčítané už vo výpočtových blokoch. Tento prístup ale neumožňuje využiť niektoré výhody FPGA obvodu, súvisiace s jeho internou štruktúrou. 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. Ale vzhľadom na fakt, že sú prepojenia umiestnené priamo v DSP blokoch, nie je možné upravovať ich zapojenie zmenou konfigurácie FPGA obvodu. Z tohto dôvodu, modely s umiestnením oneskorovacích pamätí medzi výpočtovými blokmi nie sú schopné využiť interné prepojenia medzi DSP blokmi. Príklady modelov sú uvedené v článkoch [46], [106], [107].

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. Správnosť navrhnutého modelu 2D konvolútora, jeho dátovej a riadiacej štruktúry, bola overená formou simulácie. Spôsob overenia a priebeh simulácie sú uvedené v kap. 5.

Kompromisom popísaného efektívneho návrhu modelu 2D konvolútora je aktuálne

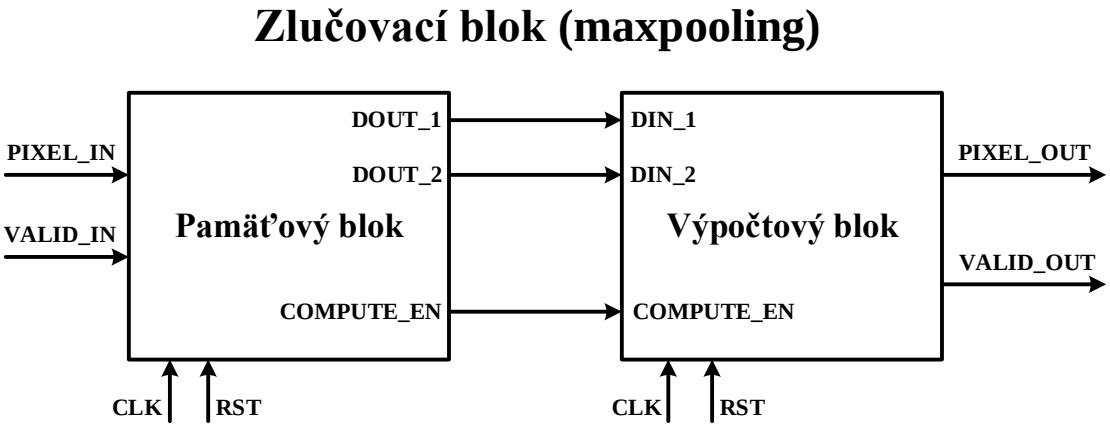
obmedzenie podporovaného formátu vstupných vzoriek z dôvodu konštantných parametrov N a K v celej architektúre. Možné zlepšenia návrhu poskytuje flexibilita FPGA obvodu s možnosťou dynamického nastavovania spomenutých parametrov priamo v priebehu výpočtu. Ak predpokladáme variabilnosť parametrov N a K so zámerom umožniť znovupoužitie blokov 2D konvolútora pre vstupné vzorky a matice koeficientov rôznych rozmerov, je nevyhnutná úprava modelu. Všetky komponenty, ktoré sú závislé od parametrov N a K, vyžadujú úpravy pri zachovaní ich požadovaného správania. Ide o komponenty: **séria SE blokov** a **oneskorovacie pamäte**. V prípade SE blokov je možné použiť neutrálny prvok operácie sčítania – *nulu* – a nahradiť všetky nadbytočné súčiny vzhľadom na aktuálnu hodnotu K. Táto funkcia je realizovateľná ako súčasť prepínacieho bloku. Dynamická veľkosť oneskorovacej pamäte je realizovateľná pomocou blokovej pamäte FPGA obvodu (BRAM) [105]. Implementácia údajovej štruktúry FIFO (angl. *First-In First-Out*) v úlohe oneskorovacej pamäte používa pre manipuláciu s pamäťou o variabilnej dĺžke offset medzi adresou aktuálnej bunky pre zápis a aktuálnej bunky pre čítanie. Zmenou týchto adries je možné nastaviť offset tak, aby dĺžka pamäte spĺňala nastavené požiadavky.

4.2 Subsystém zlučovacej vrstvy

Podobne ako konvolučná vrstva, aj zlučovacia vrstva (kap. 1.3.1) konvolučnej siete patrí medzi oknové operácie, teda vstupné operandy vykonaných operácií majú tvar matice. Z tohto dôvodu je jednou z možností návrhu architektúry subsystému zlučovacej vrstvy s následnou implementáciou do FPGA obvodu použitie rovnakých princípov ako v prípade subsystému konvolučnej vrstvy (kap. 4.1). Náhradou operácií sčítania a násobenia v navrhnutej architektúre subsystému konvolučnej vrstvy za inú operáciu, napr. maximum alebo priemer podľa spôsobu zlučovania, vznikne architektúra subsystému zlučovacej vrstvy. Vzhľadom na obmedzenú množinu hodnôt parametrov zlučovacej vrstvy (operácia, rozmery okna, dĺžka kroku) v súčasných, reálne používaných konvolučných sieťach je v práci použitý vlastný návrh s najčastejšie používanými hodnotami parametrov: operácia **maximum**, rozmery okna $\mathbf{2 \times 2}$ pixelov a dĺžka kroku **2** pixely. Nami navrhnutý subsystém zlučovacej vrstvy pozostáva z dvoch blokov: **pamäťový** blok a **výpočtový** blok, zobrazené na obr. 4.11. Rozkladom subsystému na dva samostatné bloky je zreteľne oddelená pamäť od výpočtov. V nasledujúcich podkapitolách je popísaná štruktúra každého z týchto blokov.

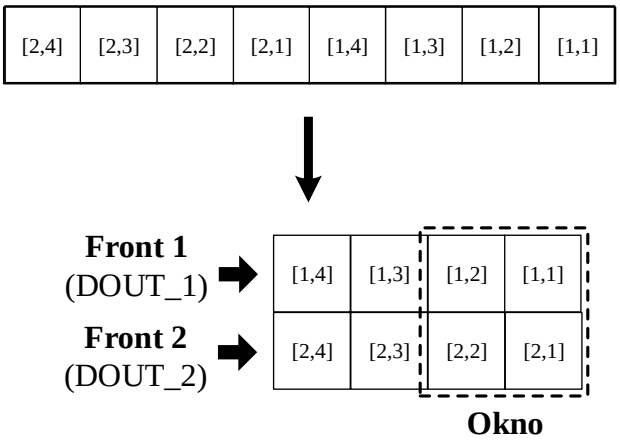
4.2.1 Pamäťový blok

Úlohou pamäťového bloku je preusporiadanie vstupných pixelov v čase. Tieto pixely prichádzajú sekvenčne cez vstupné rozhranie a ukladajú sa do pamäťových štruktúr



Obr. 4.11: 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*.

s cieľom poskytnúť susedné páry pixelov na ich ďalšie spracovanie výpočtovým blokom. Pamäťový blok má rovnakú úlohu ako oneskorovacie registre v druhej časti modelu 2D konvolútora (kap. 4.1.1). Obr. 4.12 ilustruje účinok použitia frontov na poradie pixelov.



Obr. 4.12: 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.

Štruktúra pamäťového bloku, zobrazená na obr. 4.13, pozostáva z dvoch samostatných dvojíc frontov v režime FIFO. Dôvodom pre použitie dvoch párov frontov je spôsob, ako

sú fronty realizované (formou posuvných registrov) a z toho vyplývajúci spôsob vkladania a výberu dát z frontov. Výber položky z frontu automaticky vyvolá vloženie novej položky do frontu zo vstupného rozhrania. Pre korektné fungovanie navrhnutej štruktúry pamäťového bloku požadujeme, aby pasívne fronty obsahovali len platné vstupné pixely. Počas napĺňania pasívneho páru frontov sú vložené len platné vstupné pixely. Počas vyprázdňovania aktívneho páru frontov sú ale vyberané pixely bez ohľadu na platnosť aktuálnych vstupných pixelov, keďže výpočtový blok vyžaduje neprerušovaný prúd vstupných pixelov. Tak by sa mohli do frontu dostať aj neplatné pixely, ktoré porušujú požiadavku subsystému nevkladať neplatné položky do pasívneho frontu.

Každý front má pevne zvolenú dĺžku, danú šírkou vstupnej mapy **N**. Zvolená šírka je stanovená tak, aby po naplnení každý z dvojice pasívnych frontov obsahoval pixely jedného riadku vstupnej mapy. Navyše, fronty v tomto stave obsahujú pixely susedných riadkov, umiestnených vo vstupnej mape nad sebou. Fronty sú riadené spoločným hodinovým signálom (*CLK*). Okrem hodín používa každý pár frontov vlastný povoľovací signál (angl. *Clock Enable*; CE), ktorým je riadené vkladanie a vyberanie pixelov.

V danom časovom okamihu má každý z párov frontov pridelený stav: *aktívny* alebo *pasívny*. Pasívna dvojica frontov prijíma platné vstupné pixely. Aktívna dvojica frontov vyberá pixely a posúva ich ďalej do výpočtového bloku alebo je neaktívna (čaká). Fronty pixelov sú spracovávané v striedavom režime, teda si vzájomne vymieňajú úlohu aktívneho a pasívneho stavu. Ide o mechanizmus frontov, známy ako *Ping-Pong*, použitý napríklad v [100]. Situácia preplnenia pasívneho páru frontov ešte pred zmenou ich stavu na aktívne nenastane, keďže sa aktívny pár frontov vyprázdňuje dvakrát rýchlejšie ako prebieha napĺňanie pasívneho páru frontov. Z aktívnych frontov sú totiž vyberané pixely použitím dvoch výstupných portov, zatiaľ čo pre napĺňanie pasívnych frontov je použitý len jeden vstupný port (viď. obr. 4.13).

Vkladanie len platných vstupných pixelov do pasívnych frontov, výber pixelov z aktívnych frontov vo vhodnom okamihu a striedanie roly frontov vyžaduje riadenie. Pre tento účel je navrhnutý riadiaci automat pamäťového bloku (obr. 4.14). Riadiaci automat zabezpečuje generovanie korektných riadiacich signálov pre obidva páry frontov. Automat využíva vopred známe informácie o rozmeroch vstupnej mapy **N** a veľkosti okna **K**, podobne ako v prípade riadiaceho automatu 2D konvolútora (kap. 4.1.1). Pre tento účel slúžia dve počítadlá – *COUNTER_FULL* a *COUNTER_EMPTY*. *COUNTER_FULL* počíta aktuálne vložené platné pixely do pasívnej dvojice frontov a oznamuje, kedy je pár pasívnych frontov už plný. Aby sa zabránilo preplneniu frontov, je nevyhnutné zmeniť stav plných pasívnych frontov na aktívne. Udalosť zaplnenia pasívnych frontov aktivuje *COUNTER_EMPTY*. Toto počítadlo počíta pixely, vybrané z aktívnych frontov, a oznamuje, kedy sú aktívne fronty už prázdne (neobsahujú už žiadne platné pixely) a môžu

byť opäť použité v úlohe pasívnych frontov. Stavový automat mení stavy frontov podľa hodnôt indikátorov uvedených počítadiel.

Tabuľka 4.4: **Prednastavené hodnoty riadiacich signálov a výstupov pre všetky stavy FSM pamäťového bloku subsystému zlučovacej vrstvy** (obr. 4.14). Signály sú nastavené na hodnoty v tabuľke vždy na začiatku procesu v zdrojovom kóde jazyka VHDL. V niektorých stavoch dochádza k ich zmene (tab. 4.5).

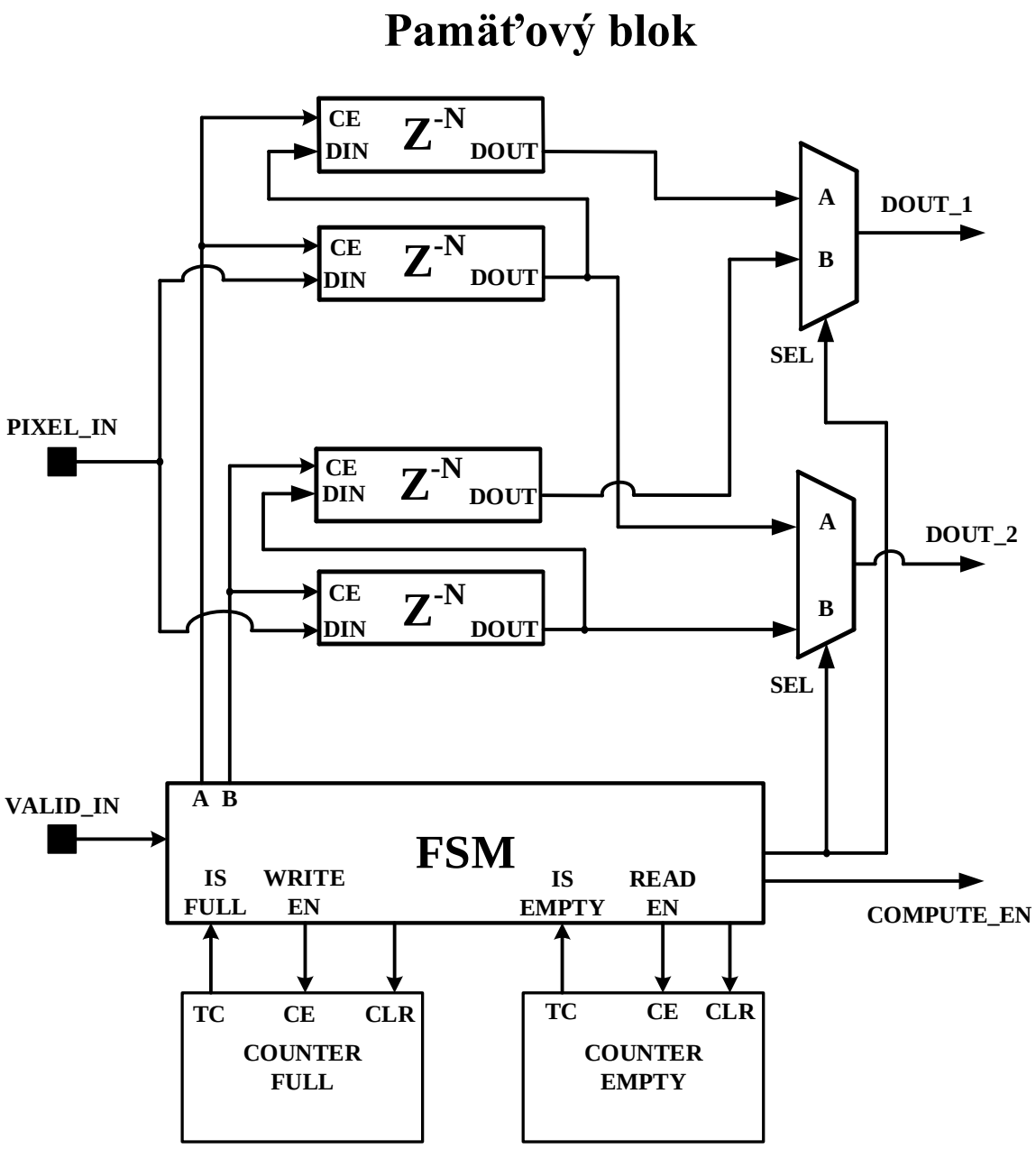
Názov	Hodnota
SEL	0
READ_EN	0
WRITE_EN	VALID_IN
CLR_FULL	0
CLR_EMPTY	0

Tabuľka 4.5: **Prehľad stavov v FSM pamäťového bloku** (obr. 4.14). Nastavenia Moorovych výstupov ([94], kap. 10) sú v tabuľke zapísané vo formáte priraďovacieho príkazu jazyka VHDL.

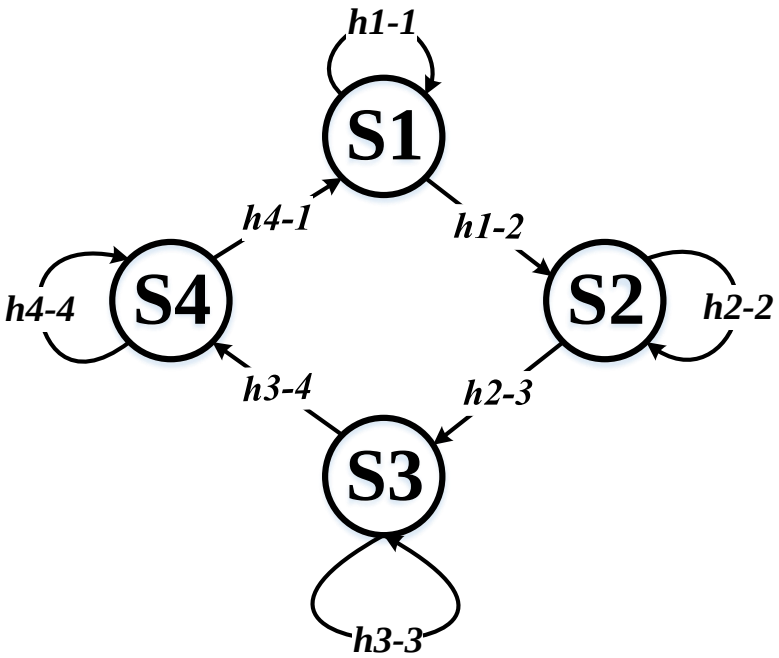
Stav	Situácia	Moorove výstupy (odlišné od tab. 4.4)
S1	Fronty A - vkladaj Fronty B - čakaj	A ⇐ VALID_IN; B ⇐ '0';
S2	Fronty A - vyberaj Fronty B - vkladaj	A ⇐ '1'; B ⇐ VALID_IN; READ_EN ⇐ '1';
S3	Fronty A - čakaj Fronty B - vkladaj	A ⇐ '0'; B ⇐ VALID_IN;
S4	Fronty A - vkladaj Fronty B - vyberaj	A ⇐ VALID_IN; B ⇐ '1'; READ_EN ⇐ '1'; SEL ⇐ '1';

Tabuľka 4.6: **Prehľad prechodov v automate riadiacej časti FSM pamäťového bloku** (hrany na obr. 4.14). Prechod medzi stavmi sa vykoná len, ak sú splnené všetky požadované podmienky v stĺpci **Podmienky prechodu**.

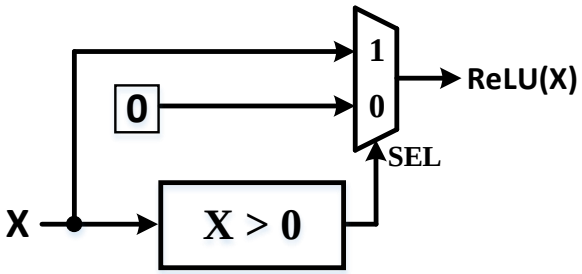
Hrana	Podmienky prechodu
h1-1	IS_FULL='0'
h1-2	IS_FULL='1'
h2-2	IS_EMPTY='0'
h2-3	IS_EMPTY='1'
h3-3	IS_FULL='0'
h3-4	IS_FULL='1'
h4-4	IS_EMPTY='0'
h4-1	IS_EMPTY='1'



Obr. 4.13: 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.



Obr. 4.14: Stavový diagram (FSM) pamäťového bloku. Riadiaca časť pozostáva zo štyroch stavov *S1*, . . . , *S4* (tab. 4.5). Hrany medzi stavmi reprezentujú podmienené prechody *h1-1*, . . . , *h4-4*, kde prvá cifra označuje číslo zdrojového stavu a druhá cifra označuje číslo cieľového stavu (tab. 4.6).

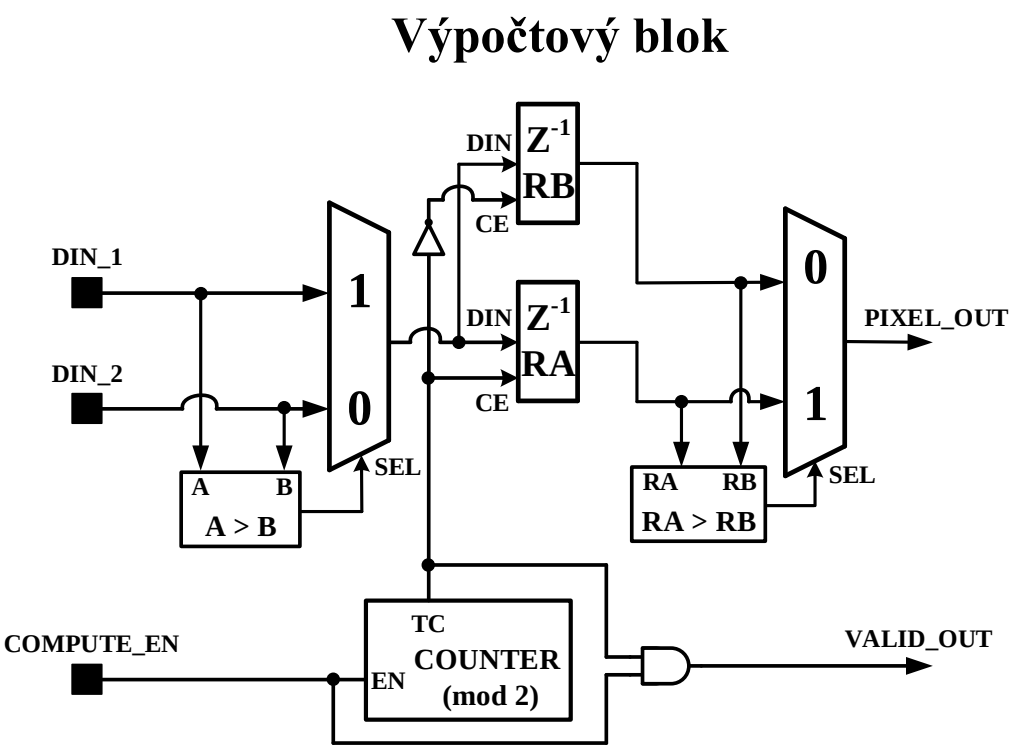


Obr. 4.15: Model rektifikovanej lineárnej jednotky (ReLU). Model správaním zodpovedá aktivačnej funkcii ReLU (1.16), pozostáva z komparátora a multiplexera. Komparátor porovnáva vstupný pixel X s nulou. Multiplexer následne posiela na výstup hodnotu vstupného pixelu alebo nulu podľa riadiaceho signálu *SEL* z komparátora.

4.2.2 Výpočtový blok

Výpočtový blok (obr. 4.16) obsahuje dva kaskádovo zapojené operátory *maximum*, každý realizovaný prostredníctvom komparátora a multiplexera. Medzi operátory sú vložené dva registre (*RA* a *RB*), ktoré slúžia na dočasné uloženie dvojice po sebe idúcich operandov. Z registrov je vždy aktívny práve jeden, registre sa pravidelne striedajú. Platnosť výstupných pixelov výpočtového bloku je daná platnosťou vstupnej dvojice pixelov z pamäťového bloku a povoľovacím signálom *CE* pre registre. Výstupná hodnota je tak platná až po prijatí a spracovaní štyroch platných pixelov, ktoré tvoria okno $\mathbf{2 \times 2}$ pixelov

(zvýraznené okno na obr. 4.12).



Obr. 4.16: 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.

4.3 Subsystém plne-prepojenej vrstvy

V prípade plne-prepojenej vrstvy (kap. 1.3.1) sme zvolili priamočiaru architektúru založenú na súčtovom strome (angl. *adder tree*). V nasledujúcej časti práce je popísaná jeho štruktúra.

4.3.1 Súčtový strom

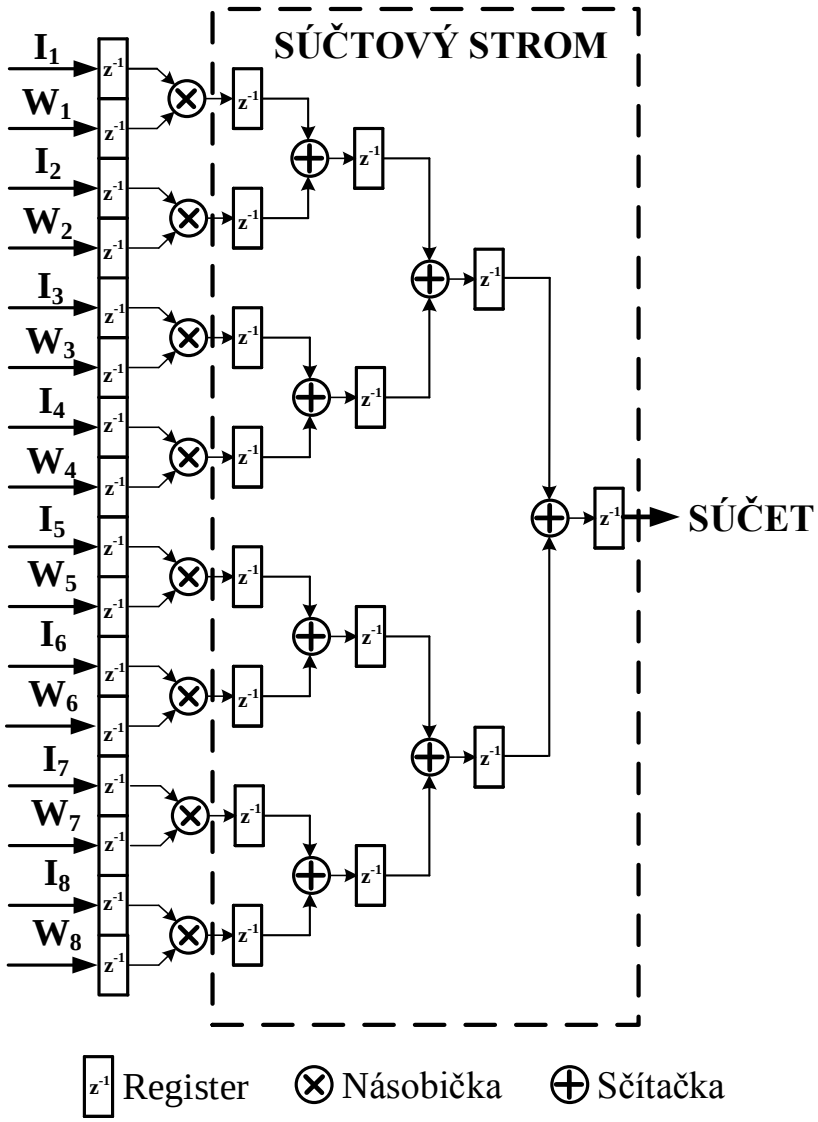
Súčtový strom vykonáva súčet N vstupných dát (pixelov), pričom využíva štruktúru binárneho stromu a prúdové spracovanie dát. Uzly v strome sú operátory dvoj-vstupového sčítania. Vstupné hrany do konkrétneho uzla predstavujú hodnoty operandov. Vychádzajúca hrana z uzla vyjadruje hodnotu výsledku sčítania vstupných operandov. 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 finálneho výsledku – súčtu všetkých vstupných dát, prijatých v rovnakom časovom okamihu. Voľba súčtového stromu vychádza z jeho efektívneho, jednoduchého návrhu, ktorý má menšie oneskorenie ako sekvenčná metóda sčítania. Navyše, susedné úrovne stromu sú prepojené prostredníctvom registrov, ktoré umožňujú ešte zvýšiť taktovaciu frekvenciu systému.

Architektúra subsystému plne-prepojenej vrstvy rozširuje pôvodne navrhnutý súčtový strom o ďalšiu úroveň uzlov. Táto úroveň je vložená pred pôvodný súčtový strom a slúži na výpočet súčinov vstupných pixelov s prislúchajúcimi váhami. Schéma rozšíreného súčtového stromu je zobrazená na obr. 4.17.

4.4 Subsystém aktivačnej vrstvy

V subsystéme aktivačnej vrstvy (kap. 1.3.1) sa výpočet vzťahuje na každý pixel vstupnej mapy samostatne, čím sa odlišuje od predchádzajúcich vrstiev. Z tohto dôvodu, návrh subsystému aktivačnej vrstvy spočíva len v návrhu subsystému konkrétnej aktivačnej funkcie (kap. 1.3.1). Subsystém aktivačnej funkcie tak predstavuje jediný výpočtový prvok subsystému aktivačnej vrstvy a vykonáva transformáciu pixelov vstupnej mapy. Spôsob, akým pracuje aktivačná vrstva, sa zhoduje s prúdovým spracovaním a nevyžaduje tak zmenu formátu alebo rozloženia vstupných pixelov. Návrh modelov známych aktivačných funkcií pre následnú implementáciu do FPGA je riešený vo viacerých vedeckých článkoch [108]–[112]. V práci je uvedený model aktivačnej funkcie *ReLU* (1.16), zobrazený na obr. 4.15. Model je daný operátorom *maximum*, ktorý pozostáva z komparátora a multiplexera, rovnako ako v prípade zlučovacej vrstvy.

Vzhľadom na usporiadanie pixelov výstupných máp, ktoré v prípade navrhovaného subsystému vychádzajú z konvolučnej a zlučovacej vrstvy vo forme vektora, je následné použitie aktivačnej vrstvy zrejmé. Ide o vloženie funkčného bloku, vykonávajúceho aktivačnú funkciu, priamo za subsystémy týchto vrstiev. Aktivačná funkcia je tak aplikovaná na každý z pixelov zvlášť. V reálnych architektúrach konvolučnej siete je aktivačná vrstva umiestnená za každou konvolučnou, zlučovacou a plne-prepojenou vrstvou s výnimkou výstupnej vrstvy. Z pohľadu implementácie systému konvolučnej siete do



Obr. 4.17: 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: $\sum_{i=1}^{8} \boldsymbol{I_i \times W_i}$. Susedné úrovne stromu sú prepojené cez registre z dôvodu obmedzenia dĺžky kritickej cesty. Vložené registre tak umožňujú zvýšiť taktovaciu frekvenciu.

FPGA je funkčný blok, reprezentujúci subsystém aktivačnej vrstvy, vložený za každú z implementovaných vrstiev.

Kapitola 5

Experimentálne overenie architektúry

Z dôvodu overenia požadovaného správania subsystémov, popísaných v predchádzajúcej kap. 4, bola zvolená vstupná referenčná konvolučná sieť a jej referenčné hodnoty: vstupné mapy a požadované výstupné hodnoty. Architektúra siete (typy vrstiev, ich počet a usporiadanie, váhové matice a pod.) slúžila ako podklad pre vytvorenie systému, zloženého z v práci popísaných subsystémov jednotlivých vrstiev. Architektúra referenčnej siete spolu s formátom referenčných hodnôt je popísaná v podkapitole 5.1.

Keďže je cieľovou implementačnou platformou FPGA obvod, boli následne zvolené dva konkrétne prístupy popisu nami navrhnutého systému: **modelovo-riadený prístup** [113] 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 a jeho subsystémov formou simulácie, teda overenie miery zhody referenčných hodnôt s hodnotami, získanými z nami vytvoreného modelu. Zámerom druhého prístupu je **overenie možnej implementácie navrhnutého systému do FPGA obvodu**, **analýza syntézy systému** (prevod HDL kódu do zdrojov FPGA) a **výsledné požiadavky systému na typ a množstvo zdrojov FPGA obvodu**. Tieto prístupy sú popísané v podkapitolách 5.2.1 a 5.2.2. Zdrojové súbory, diagramy vzájomnej závislosti entít a iné podporné dokumenty ku obom uvedeným modelom sú dostupné v elektronickej forme v *prílohe A* predkladanej práce.

5.1 Referenčná konvolučná sieť

Pre overenie korektnosti správania nami navrhnutých subsystémov v kap. 4 bola použitá konkrétna, jednoduchá konvolučná sieť. Dané vstupné a výstupné hodnoty tejto konvolučnej siete, v práci označované ako *referenčné hodnoty* použité počas simulácie, pochádzajú z jej implementácie do platformy GPU. Popis implementácie správania siete do GPU ani overenie jej správnosti nie sú súčasťou práce. V predkladanej práci označujeme danú konvolučnú sieť ako **referenčnú sieť** a jej implementáciu do GPU ako **referenčný model**. Popis architektúry zvolenej referenčnej siete, jej váhových matíc, vstupných a výstupných máp je uvedený v tejto podkapitole.

5.1.1 Architektúra referenčnej siete

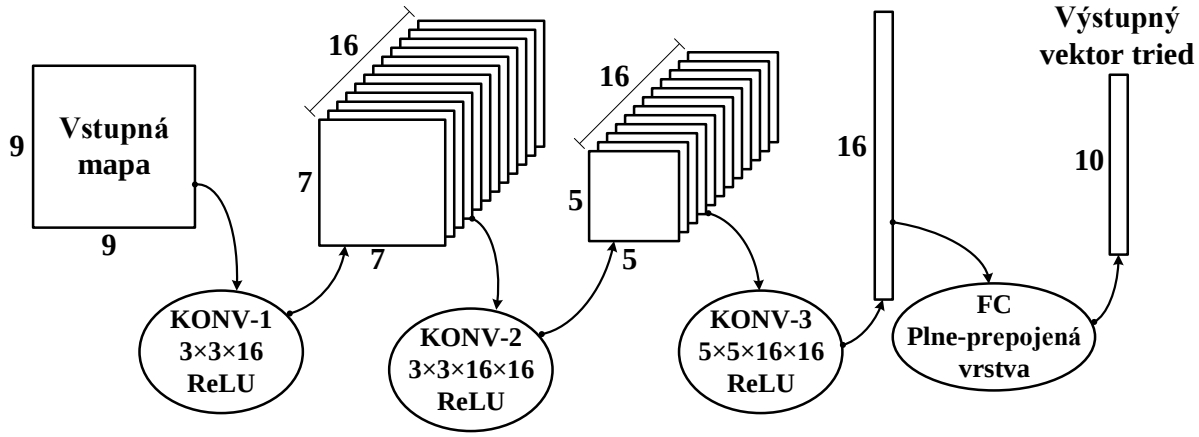
Referenčná konvolučná sieť, podľa obrázku 5.1, pozostáva zo štyroch vrstiev (okrem vstupnej vrstvy, ktorá nie je uvedená): *KONV-1*, *KONV-2*, *KONV-3* a *FC*. *KONV-1* je konvolučná vrstva s jednou vstupnou mapou (vstupný obrázok s rozmermi $\mathbf{9 \times 9}$ pixelov) a šestnástimi výstupnými mapami. Pre výpočet používa šestnásť váhových matíc s rozmermi $\mathbf{3 \times 3}$. Výstupné príznakové mapy vrstvy *KONV-1* majú rozmery $\mathbf{7 \times 7}$ pixelov. Ďalšou vrstvou v poradí je druhá konvolučná vrstva (*KONV-2*), ktorá používa $\mathbf{16 \times 16}$ váhových matíc s rozmermi $\mathbf{3 \times 3}$. Výstupom z *KONV-2* je opäť šestnásť príznakových máp s rozmermi $\mathbf{5 \times 5}$ pixelov. *KONV-3* je poslednou z konvolučných vrstiev, používa $\mathbf{16 \times 16}$ váhových matíc s rozmermi $\mathbf{5 \times 5}$ a generuje vektor šestnástich príznakov. Za každou z uvedených konvolučných vrstiev je vložená aktivačná vrstva. Teda, na príznaky, vychádzajúce z konvolučnej vrstvy, je aplikovaná aktivačná funkcia ReLU (1.16). Výstupnou vrstvou siete je plne-prepojená vrstva *FC*, ktorá generuje na výstupe vektor desiatich hodnôt. Hodnoty reprezentujú mieru príslušnosti vstupného obrázku do jednotlivých tried.

5.1.2 Bloková schéma referenčnej siete

Bloková schéma referenčnej siete je zobrazená na obr. 5.2. Jej štruktúra je vystavaná zo skupiny funkčných blokov, ktoré zodpovedajú subsystémom jednotlivých vrstiev z kap. 4. Funkčné bloky sú vzájomne prepojené tak, aby správanie výsledného modelu zodpovedalo správaniu referenčnej siete.

5.1.3 Testovací dataset

Upravená testovacia sada obrázkov z datasetu MNIST (3.1.1) bola použitá ako testovacia sada pre overenie správania navrhnutého modelu (obr. 5.2). Zvolená testovacia sada (obr.



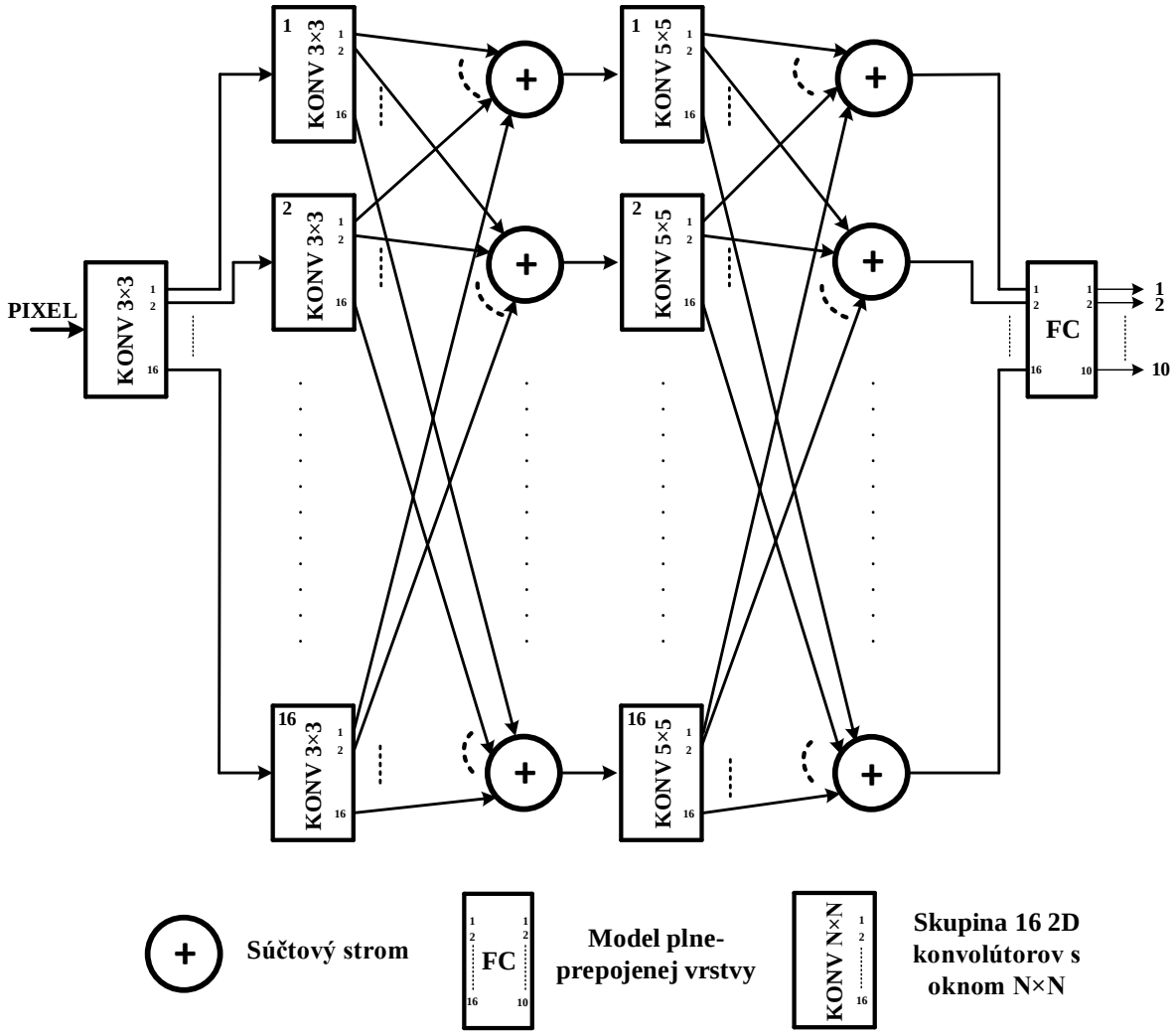
Obr. 5.1: Architektúra referenčnej siete. Zvolená konvolučná sieť pozostáva zo štyroch vrstiev (okrem vstupnej vrstvy): *KONV-1*, *KONV-2*, *KONV-3* a *FC*. Čísla pri vstupných a výstupných mapách uvádzajú ich rozmery. Počet a rozmery váhových matíc sú uvedené pod označeniami jednotlivých vrstiev.

5.3) pozostáva zo sto obrázkov (desať obrázkov pre každú cifru). 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 vo formáte pevnej rádovej čiarky (angl. *fixed-point*). Keďže pixely vstupných obrázkov sú pôvodne normované a nezáporné, pre vyjadrenie hodnôt pixelov (čísla z intervalu $\mathbf{(0, 1)}$) bolo všetkých osem bitov (s výnimkou najvyššieho znamienkového bitu) použitých pre desatinnú časť. Váhy vo váhových maticiach všetkých vrstiev sú vyjadrené ako 5-bitové čísla z intervalu $\mathbf{(-1, 1)}$. Pre vyjadrenie váh bolo rovnako tak použité kódovanie s pevnou rádovou čiarkou ako v prípade vstupných pixelov. Ale z dôvodu zápornej časti intervalu bol tiež rezervovaný jeden bit pre znamienko. Zvyšné štyri bity sú použité pre desatinnú časť. Hodnoty *BIAS* sú kvôli prehľadnosti vynechané, teda sú v celej sieti nulové.

Kódovanie číselných hodnôt

Použitý formát čísel s pevnou rádovou čiarkou a s obmedzeným počtom bitov prináša potenciálne nebezpečenstvo pretečenia (angl. *overflow*). Pretečenie spôsobuje nepresnosť numerických operácií sčítania a násobenia. Pre zamedzenie vzniku pretečenia je potrebné zväčšovať efektívnu bitovú šírku medzivýsledkov, ktoré sú spracované použitím numerických operácií opakovaného násobenia a sčítania. Detaily aritmetiky čísel s pevnou rádovou čiarkou sú uvedené v [114], [115].

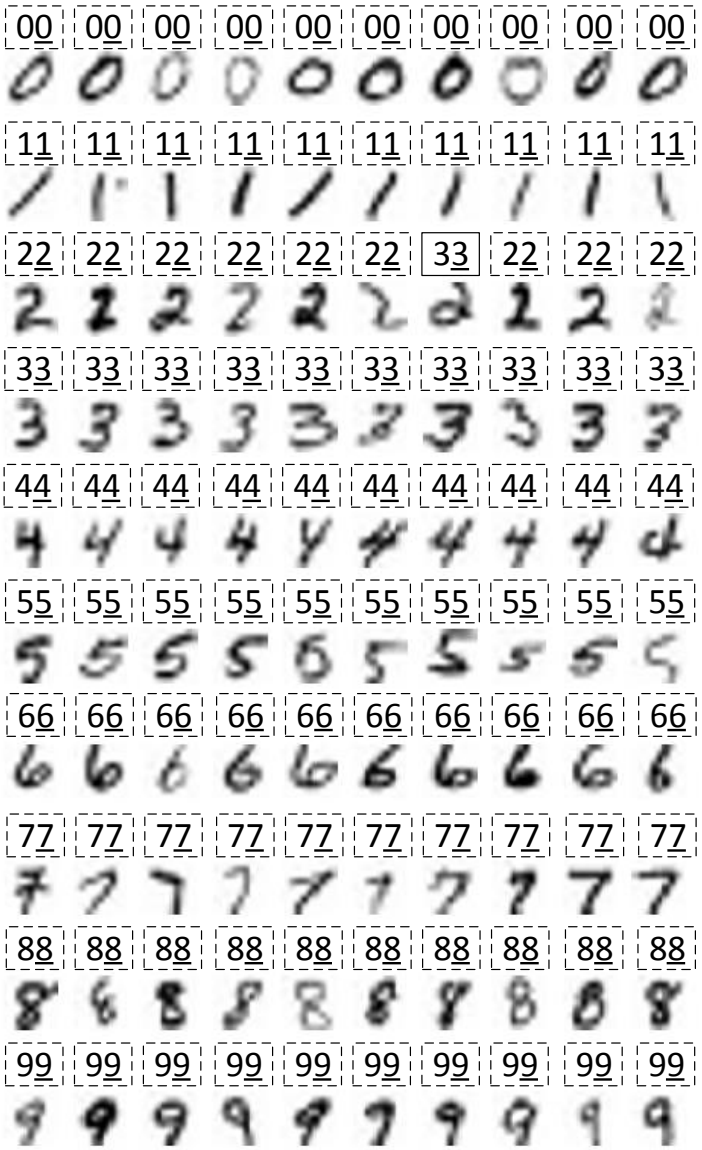
Neustále zvyšovanie bitovej šírky príznakov ale nie je možné, keďže sú výstupné príznaky jednej vrstvy použité následne ako vstupné príznaky ďalšej vrstvy. Zachovanie rovnakej bitovej šírky vstupných príznakov pre všetky vrstvy dosiahneme orezaním



Obr. 5.2: Bloková schéma referenčnej siete. Schéma zobrazuje prepojenie funkčných blokov tak, aby sa model správal ako referenčná sieť. Funkčné bloky zodpovedajú modelom vrstiev z kap. 4. Každý z blokov *KONV* $\mathbf{3 \times 3}$ a *KONV* $\mathbf{5 \times 5}$ pozostáva zo skupiny šestnástich 2D konvolútorov a spolu zabezpečujú funkciu konvolučnej vrstvy. Výstupy z týchto blokov sú sčítané dokopy s použitím súčtových stromov pre získanie výstupných príznakových máp. Blok *FC* poskytuje funkciu plne-prepojenej vrstvy.

výstupných príznakov na každej vrstve, teda odstránením najnižších bitov. Pre minimalizovanie nepresnosti, spôsobenej touto operáciou, sú pred orezaním hodnoty príznakov zaokrúhlené. Kombinácia rozširovania bitovej šírky medzivýsledkov v rámci vrstvy, zaokrúhlenie výstupných príznakov pred odchodom z vrstvy a orezanie ich bitovej šírky zamedzujú vzniku pretečenia. Na druhej strane, obmedzenie bitovej šírky spôsobuje taktiež nepresnosť a zaokrúhľovanie vyžaduje doplniť implementáciu modelu o ďalšie numerické operácie ([116], kap. 4).

V práci vytvorených modeloch boli pre hodnoty vstupných máp do jednotlivých vrstiev siete použité rôzne parametre formátu s pevnou rádovou čiarkou, uvedené v tab. 5.1. Voľba šírky celej a desatinnej časti hodnôt bola zvolená tak, aby nedošlo k pretečeniu.



Obr. 5.3: Upravená testovacia sada obrázkov z datasetu MNIST. Obrázky majú rozmery $\mathbf{9 \times 9}$ 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. Z pohľadu presnosti klasifikácie je chybne pridelená trieda len v prípade jednej vzorky (pridelené triedy danej vzorky sú ohraničené celou čiarou).

Tabuľka 5.1: Parametre formátu s pevnou rádovou čiarkou pre hodnoty vstupných máp jednotlivých vrstiev siete. Hodnoty vyjadrujú počet vyhradených bitov pre danú časť.

	CONV1	**CONV2**	**CONV3**	**FC**
Celá časť	1	3	4	6
Desatinná časť	8	7	6	4

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*) [117] (kap. 8). Dôvodom je možnosť použiť nástroje Simulink a System Generator pre automatický prevod grafického modelu na HDL kód (automatický prevod modelu do HDL kódu s použitím uvedených nástrojov nebol počas praktickej realizácie práce vykonaný).

Simulácia modelu v nástroji Matlab/Simulink prebiehala po vrstvách z dôvodov spotreby operačnej pamäte počas jej behu. Pôvodný model celého systému bol rozložený do viacerých menších modelov, prislúchajúcich jednotlivým vrstvám. Simulácia konkrétnej vrstvy bola spustená až po vykonaní a overení správnosti simulácie všetkých predchádzajúcich vrstiev modelu. Údaje modelu (parametre referenčnej siete, referenčné hodnoty a hodnoty príznakových máp vypočítané počas simulácie) sú zdieľané všetkými modelmi prostredníctvom pracovného priestoru nástroja Matlab (angl. *workspace*). Tento prístup postupného vykonávania simulácie čiastočných modelov umožňuje dynamicky sledovať a overovať výstupné hodnoty z každej vrstvy siete. To prináša výhody rýchleho objavenia chýb modelu už v skorej fáze testovania.

Pre funkčné overenie modelu sme zvolili viacero parametrov, ktorých priebeh je v práci aj graficky zobrazený:

- *finálna klasifikácia vstupných obrázkov*, t.j. porovnanie tried, pridelených referenčným modelom, s triedami, ktoré vzorkám priradil nami navrhnutý model počas simulácie (obr. 5.3),
- *stredná kvadratická chyba* (angl. *Root Mean Square Error*; RMSE) modelu (obr. 5.4) pre každú z vrstiev siete,
- *absolútna hodnota rozdielov* medzi simuláciou získanými hodnotami a referenčnými hodnotami (obr. 5.5) pre každú z vrstiev siete.

Pre porovnanie klasifikácie v práci popísaného modelu s referenčnými hodnotami slúži ilustrácia vstupných testovacích obrázkov a k nim pridelených tried na obr. 5.3. Trieda, pridelená nami navrhnutým modelom, a trieda, predikovaná referenčným modelom, sa vo všetkých testovacích prípadoch zhodujú.

Pre vyčíslenie priemernej chyby na jednotlivých vrstvách siete je na obr. 5.4 pre každú z vrstiev zachytená stredná kvadratická chyba (RMSE) s použitím vzťahu (5.1),

$$\mathrm{RMSE}\,(\boldsymbol{i}) = \sqrt{\frac{\sum_{\boldsymbol{k}=\boldsymbol{1}}^{\boldsymbol{N}} \left(\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{sim}} - \boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{ref}}\right)^{\boldsymbol{2}}}{\boldsymbol{N}}} \tag{5.1}$$

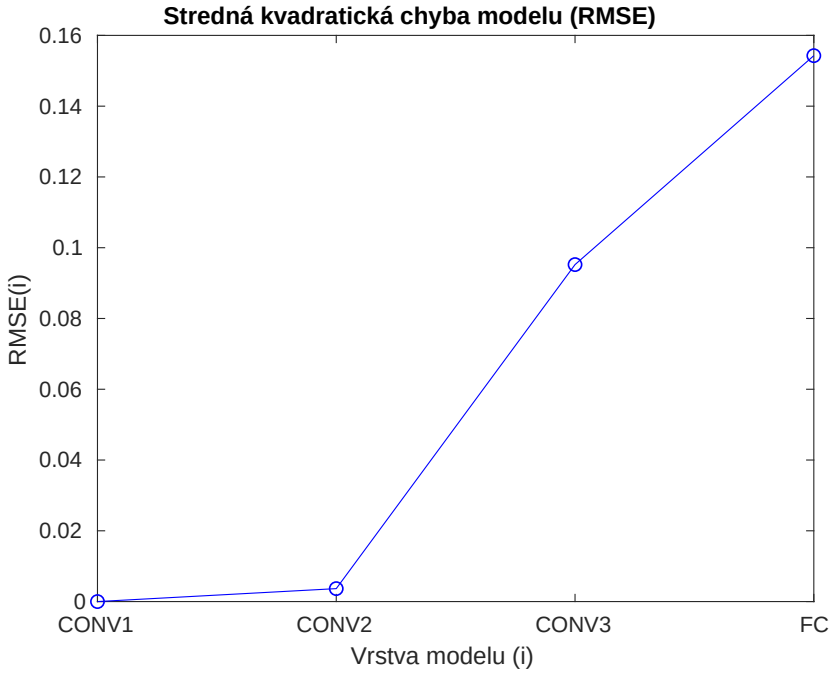
kde $\boldsymbol{i}$ je index vrstvy, $\boldsymbol{N}$ je počet všetkých výstupných príznakov na $\boldsymbol{i}$-tej vrstve, prislúchajúcich ku všetkým testovacím obrázkom, $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{sim}}$ je hodnota $\boldsymbol{k}$-teho výstupného príznaku $\boldsymbol{i}$-tej vrstvy získaná simuláciou nami navrhnutého modelu a $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{ref}}$ je hodnota prislúchajúceho $\boldsymbol{k}$-teho výstupného príznaku z referenčného modelu. Pre detailnejšie porovnanie presnosti výpočtov modelu je na obr. 5.5 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 (5.2),

$$\boldsymbol{\Delta}_{\boldsymbol{i},\boldsymbol{k}} = \left|\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{sim}} - \boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{ref}}\right| \tag{5.2}$$

kde podobne ako v (5.1), $\boldsymbol{i}$ je index vrstvy, $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{sim}}$ je hodnota $\boldsymbol{k}$-teho výstupného príznaku $\boldsymbol{i}$-tej vrstvy získaná simuláciou a $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k},\boldsymbol{ref}}$ je hodnota prislúchajúceho $\boldsymbol{k}$-teho výstupného príznaku z referenčného modelu. 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. Napriek tomu vykazuje prevažná väčšina hodnôt len minimálnu chybu.

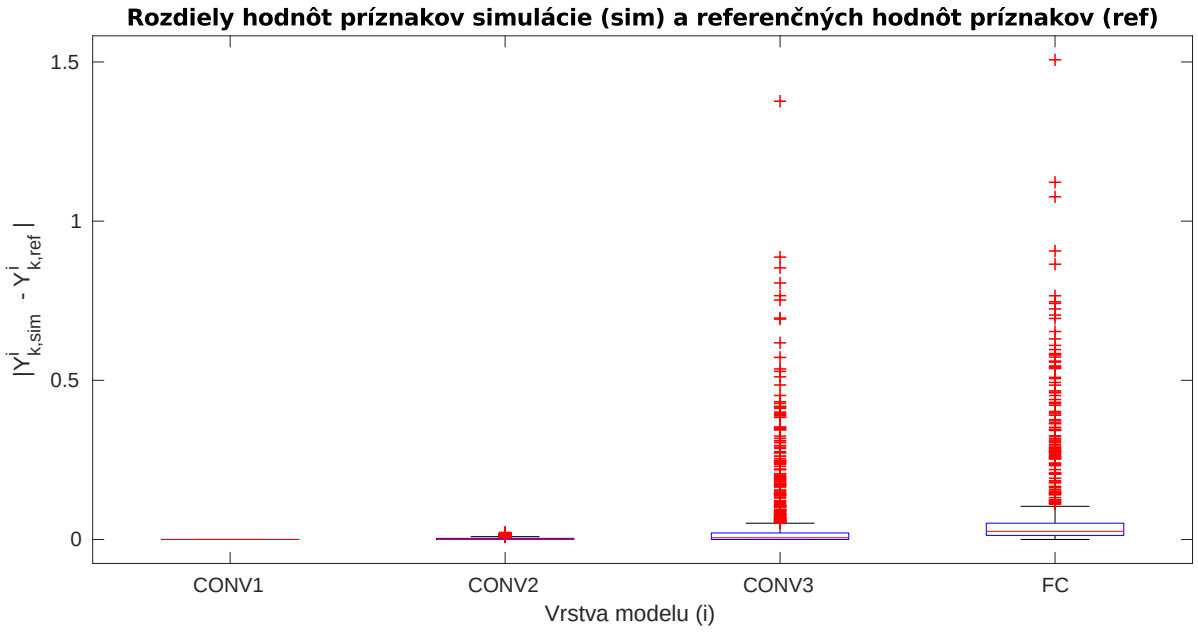
Nárast chyby je spôsobený zmenou formátu kódovania vstupných a výstupných hodnôt (formát čísel s pevnou rádovou čiarkou, kap. 5.1.3), konkrétne zmenšovaním počtu bitov určených pre vyjadrenie desatinnej časti hodnôt. Kým v prípade prvej vrstvy (v schéme 5.2 označená ako *CONV1*) bolo použitých pre desatinnú časť vstupných hodnôt až osem bitov (presnosť je $\boldsymbol{1/2^9 \approx 0.00195}$), pre desatinnú časť vstupných hodnôt štvrtej vrstvy (označená ako *FC*) boli použité už len štyri bity (presnosť je $\boldsymbol{1/2^5 = 0.03125}$). Najnižší bit desatinnej časti výstupných hodnôt z jednej vrstvy je pri vstupe do ďalšej vrstvy odstránený. Ďalším dôvodom sú nepresnosti vo výpočtoch jednej vrstvy, ktoré vedú k narastajúcej nepresnosti aj vo výpočtoch ďalších vrstiev. Navyše, počet numerických operácií v ďalších vrstvách referenčnej siete sa zväčšuje vzhľadom na počet vstupných a výstupných máp ako aj rozmery váhových matíc vrstiev *CONV2* a *CONV3*.

Ďalším parametrom modelu je **oneskorenie** spracovania (angl. *delay*), ktoré je v práci vyjadrené ako dĺžka časového intervalu medzi prijatím prvého platného pixelu vstupného obrázku a odovzdaním modelom pridelenej výstupnej triedy. Vzhľadom na diskrétnu povahu nástroja Matlab/Simulink, dĺžka časového intervalu je vyjadrená ako počet periód



Obr. 5.4: Stredná kvadratická chyba modelu. X-os vyjadruje jednotlivé vrstvy referenčnej siete. Y-os vyjadruje strednú kvadratickú chybu jednotlivých vrstiev danú vzťahom: $\mathrm{RMSE}\,(\boldsymbol{i}) = \sqrt{\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} \left(\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k,sim}} - \boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k,ref}}\right)^{\mathbf{2}} / \boldsymbol{N}}$, kde $\mathbf{N}$ je celkový počet výstupných príznakov na i-tej vrstve, prislúchajúcich ku všetkým testovacím obrázkom, $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k,sim}}$ je hodnota k-teho výstupného príznaku i-tej vrstvy získaná simuláciou nami navrhnutého modelu a $\boldsymbol{Y}^{\boldsymbol{i}}_{\boldsymbol{k,ref}}$ je hodnota prislúchajúceho k-teho výstupného príznaku z referenčného modelu.

synchronizačného *hodinového* signálu. Súčasne, platnosť vstupných a výstupných príznakov pre každý zo subsystémov nie je určená hodnotou vstupných a výstupných **dátových** signálov ale hodnotou **stavových** signálov *valid_in* a *valid_out*. Signál platnosti je nastavený len v riadiacej časti subsystému, a tak je jeho oneskorenie dané len oneskorením riadiacej časti. Vychádzajúc z kap. 4, riadiaca časť je tvorená z reťaze oneskorovacích registrov alebo zo stavového automatu. V prvom prípade ide o subsystémy, ktoré nevyžadujú špecifické riadenie; oneskorovacie registre slúžia len pre synchronizáciu so spracovaním signálov v dátovej časti, ich dĺžka určuje oneskorenie subsystému. V druhom prípade je oneskorenie dané štruktúrou stavového automatu. Celkové oneskorenie modelu je dané ako súčet oneskorení, spôsobených prechodom vstupného signálu validity postupne cez jednotlivé subsystémy v poradí podľa obr. 5.2. Okrem toho platí, že časový rozdiel medzi príchodom prvého platného pixelu susedných obrázkov, daný celkovým počtom pixelov na obrázku (pre referenčnú sieť v predkladanej práci je rovný 81 períód), je zachovaný na každej vrstve modelu. Preto je dostačujúce, vyjadriť oneskorenie subsystémov len pre prvý pixel jedného vstupného obrázku. Oneskorenia subsystémov sú zhrnuté v tab. 5.2, kde čas príchodu prvého platného údaju do subsystému $\boldsymbol{i}$-tej vrstvy $\boldsymbol{T}^{\boldsymbol{i}}_{\mathbf{1}}$ je vyjadrený ako počet



Obr. 5.5: Rozdiely hodnôt príznakov simulácie modelu a referenčných hodnôt príznakov. X-os vyjadruje jednotlivé vrstvy referenčnej siete. Y-os vyjadruje absolútnu hodnotu rozdielu hodnôt výstupných príznakov zo simulácie ($\boldsymbol{Y^i_{k,sim}}$) a referenčných hodnôt ($\boldsymbol{Y^i_{k,ref}}$), kde i je index vrstvy a k je poradie príznaku na i-tej vrstve.

periód hodinového signálu od začiatku simulácie. Oneskorenie subsystému $\boldsymbol{i}$-tej vrstvy $\boldsymbol{\Delta T^i}$ je v tab. 5.2 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 po spustení simulácie (aktiváciou resetovacieho signálu *RESET*) vchádza prvý platný vstupný pixel do prvej vrstvy modelu (CONV1) až v čase $\boldsymbol{T_1^1 = 4}$.

Tabuľka 5.2: Oneskorenia subsystémov navrhnutého modelu

	CONV1	**CONV2**	**CONV3**	**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*.

Hierarchická štruktúra a rozhrania VHDL modelu

Štruktúra výsledného VHDL modelu je organizovaná **hierarchicky** (obr. B.1 v prílohe B). Jednotlivé uzly grafu reprezentujú entity. Prepojenia medzi uzlami vyjadrujú vzťah *rodič-potomok*, kde entita umiestnená vyššie predstavuje rodiča a entita umiestnená nižšie jeho priameho potomka. Top entita *CNN* pozostáva z inštancií entít: *CONV-LAYER*, *MAXPOOLING-LAYER*, *FC-LAYER* a *ACTIVATION-LAYER*, reprezentujúcich subsystémy jednotlivých typov vrstiev. Každá z týchto entít sa ďalej vetví podľa svojej vlastnej štruktúry.

Rozhrania entít sú zoskupené do logických blokov a ilustrované samostatne na obrázkoch, umiestnených v *prílohe B* predkladanej práce. Rozhranie každej z entít pozostáva z dvoch častí: **zoznam parametrov** (uvedený nad deliacou čiarou) a **zoznam portov** (uvedený pod deliacou čiarou). Dátový typ všetkých použitých parametrov je celé číslo (*Integer*). Porty sú popísané cez: **názov**, **smer** (vstup - *in* / výstup - *out*), **dátový typ** a **rozmer**. Pre zápis portov na obrázkoch v prílohe B bola použitá syntax jazyka VHDL.

Rozhranie konvolučnej vrstvy (*CONV-LAYER*) a jej priamych potomkov je uvedené na obr. B.2. Rozhranie 2D konvolútora (*CONV-2D*) spolu s jeho potomkami je zobrazené na obr. B.3 a B.4. Rozhranie riadiacej časti 2D konvolútora (*FSM*) je uvedené na obr. B.5. Obr. B.6 popisuje rozhrania entít tvoriacich zlučovaciu vrstvu (*MAXPOOLING-LAYER*). Rozhrania plne-prepojenej (*FC-LAYER*) a aktivačnej vrstvy (*ACTIVATION-LAYER*) sú zobrazené na obr. B.7 a B.8.

RTL schémy entít VHDL modelu

Pre znázornenie vnútornej štruktúry entít (uvedených na obr. B.1) boli vytvorené zodpovedajúce RTL (angl. *Register-Transfer Level*) schémy, umiestnené v *prílohe C* predkladanej práce. Zloženie entity je v schéme vyjadrené cez prepojenie modulov a základných RTL prvkov. RTL schéma pre entitu subsystému konvolučnej vrstvy je uvedená na obr. C.1. Táto schéma pozostáva z opakujúcich sa modulov 2D konvolútora (kap. 4.1.1) a súčtového stromu (kap. 4.3.1). RTL schémy 2D konvolútora a súčtového stromu sú uvedené na obr. C.2 a C.8. Prvá časť 2D konvolútora, pozostávajúca z reťaze 1D konvolútorov (obr. 4.4), má RTL schému uvedenú na obr. C.3 a C.4. Obr. C.5 ilustruje RTL schému druhej časti 2D konvolútora, ktorá pozostáva z oneskorovacích registrov a sčítačiek. RTL schémy riadiacej časti 2D konvolútora a interne-použitého počítadla (kap. 4.1.1) sú zobrazené na obr. C.6 a C.7. Zlučovacia vrstva (kap. 4.2), jej pamäťová a výpočtová časť majú RTL schémy vykreslené na obr. C.9, C.10 a C.11. RTL schémy plne-prepojenej a aktivačnej vrstvy nie sú v práci uvedené. Schéma plne-prepojenej vrstvy je takmer identická so schémou súčtového stromu a schéma aktivačnej vrstvy, konkrétne ReLU jednotky, je zhodná s obr.

4.15.

Spotreba FPGA zdrojov

Pre vyjadrenie spotreby FPGA zdrojov boli subsystémy z kap. 4 analyzované nezávisle, pre každý subsystém bola vykonaná samostatná syntéza. Dôvodom je získanie spotreby zdrojov pre každý subsystém zvlášť (Vivado poskytuje len tabuľku so sumárnou spotrebou zdrojov pre syntetizovaný projekt). Spotreba zdrojov FPGA pre jednotlivé subsystémy je zhrnutá v tab. 5.3, 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* (spotreba zdrojov na jednú funkčnú jednotku ReLU je uvedená v poslednom riadku tabuľky).

Tabuľka 5.3: 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**
CONV1	1	16	3	750	320	818	144	173
CONV2	16	16	3	7627	0	17784	2304	324
CONV3	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, konkrétne sčítačky v druhej časti 2D konvolútora, a násobičky a sčítačky v plne-prepojenej vrstve, sú implementované prostredníctvom LUT. Z tohto dôvodu závisí spotreba DSP blokov len od počtu 2D konvolútorov (daných počtom vstupných a výstupných príznakových máp) a rozmerov váhových matíc, použitých v konvolučnej vrstve. 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. Spotreba vstupno-výstupných (*IO*) zdrojov je daná počtom vstupných a výstupných portov subsystému, a ich bitovou šírkou. V prípade ostatných typov zdrojov, uvedených v tab. 5.3, nie sme schopní vzhľadom na zložitosť procesu syntézy a mapovania zdrojov do FPGA obvodu priamo vyjadriť ich spotrebu v závislosti od parametrov subsystému.

Vychádzajúc z hodnôt tab. 5.3 je ideálny úplne paralelný návrh implementovateľný len do najvýkonnejších FPGA obvodov z dôvodu vysokých požiadaviek na DSP bloky (viď.

spotreba DSP blokov pre *CONV3*). Preto je jedným z riešení úprava úplne paralelného návrhu na **čiastočne paralelný návrh**. Ide o náhradu viacerých funkčne rovnakých výpočtových blokov za jeden funkčný blok rovnakého typu, 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. Čiastočne paralelnému návrhu sa predkladaná práca detailnejšie nevenuje.

Výkon systému ovplyvňujú najmä časovo a výpočtovo kritické operácie. V prípade konvolučnej siete ide o operácie násobenia a následného sčítania, ktoré dominujú v prvej časti 2D konvolútora. Z tohto dôvodu je výkon systému konvolučnej siete štandardne meraný počtom operácií, vykonaných nad číslami vo formáte s pohyblivou rádovou čiarkou za jednu sekundu (angl. *Floating-point operations per second*; Flops). V prípade navrhnutého systému v predkladanej práci sú ale použité operácie nad číslami vo formáte s pevnou rádovou čiarkou. Preto je pre systém v práci použitá jednotka **FMA** (angl. *Fused Multiply Add*). Ide o počet kombinovaných operácií, pozostávajúcich z po sebe idúcich operácií násobenia a sčítania. V FPGA obvode sú za ich výpočet zodpovedné DSP bloky, ktoré sú kvôli svojej špecializácii schopné realizovať tieto operácie rýchlejšie než konfigurovateľné logické bloky (CLB) [118]. V prípade FPGA obvodov od výrobcu *Xilinx* je jeden DSP blok schopný vykonať jednu FMA operáciu za jednu periódu hodinového signálu. Z toho vyplýva, že výkon systému môžeme vyjadriť 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 maximálnej operačnej frekvencie týchto DSP blokov v zvolenom type FPGA obvodu. Numerické operácie sčítania, použité v sčítačkách druhej časti 2D konvolútorov a v súčtových stromoch, sú realizované prostredníctvom CLB. V prípade týchto operácií nejde o kombináciu násobenia a sčítania, preto sú z uvedeného vzťahu vyňaté. Maximálna operačná frekvencia DSP blokov v FPGA obvode závisí od typu obvodu a jeho rýchlostnej triedy (angl. *speed grade*). Hodnota maximálnej operačnej frekvencie DSP blokov je uvedená v dokumente, popisujúcom technické parametre konkrétneho typu FPGA obvodu. Podľa tab. 5.3 používa navrhnutý systém spolu 8848 DSP blokov, teda jeho teoretický výkon je $\mathbf{8848} \times \boldsymbol{F}_{\mathrm{max}}$. Ako príklad predpokladajme FPGA obvody od výrobcu Xilinx rodiny **Artix-7** a **Kintex-7** s rýchlostnou triedou **-1**, konkrétne *XC7A100T-1CSG324C* ($\boldsymbol{F}_{\mathrm{max}} = \mathbf{464,25}$Mhz) a *XC7K325T-1FFG676* ($\boldsymbol{F}_{\mathrm{max}} = \mathbf{547,95}$Mhz) v prípade použitia všetkých registrov v DSP bloku. Teoretický výkon systému a maximálny výkon FPGA obvodu pre zvolené príklady FPGA obvodov sú uvedené v tab. 5.4.

Tabuľka 5.4: Prehľad výkonu podľa FPGA obvodu

	XC7A100T-1	**XC7K325T-1**
Počet DSP blokov	240	840
$\boldsymbol{F_{max}}$	464,25 Mhz [119]	547,95 Mhz [120]
Teoretický výkon	4107,6 GFMA/s	4848,3 GFMA/s
Maximálny výkon	111,4 GFMA/s	460,3 GFMA/s

5.3 Zhrnutie

Subsystémy, navrhnuté v kap. 4, boli použité pre vytvorenie modelu konvolučnej siete s konkrétnou štruktúrou (obr. 5.1). Pre overenie požadovaného správania jednotlivých subsystémov bol vytvorený ich grafický model v softvérovom nástroji Matlab/Simulink (kap. 5.2.1). Vykonaním simulácie modelu s použitím referenčných hodnôt boli analyzované výstupy z modelu. Hodnoty modelu boli porovnané s referenčnými hodnotami. Pre vyjadrenie rozdielov medzi referenčnými hodnotami a hodnotami navrhnutého modelu boli sledované tri premenné: **presnosť klasifikácie** (obr. 5.3), **absolútna hodnota rozdielov** (obr. 5.5) a **stredná kvadratická chyba** (obr. 5.4). Porovnaním klasifikácie referenčného modelu s v práci navrhnutým modelom nastala zhoda v pridelených triedach všetkých vstupných obrázkov testovacieho datasetu. Ďalšie dve uvedené premenné zachytili rozdiely medzi hodnotami referenčného a navrhnutého modelu. Tieto rozdiely ale neovplyvňujú správnosť výsledkov modelu, sú spôsobené prevažne prirodzenou nepresnosťou spojenou s použitím kódovania s pevnou rádovou čiarkou s obmedzenou bitovou šírkou. Pre overenie možnosti implementácie modelu do FPGA obvodu bol následne vytvorený popis navrhnutého modelu v jazyku VHDL (kap. 5.2.2), spracovaný vývojovým nástrojom Vivado. Výstupom boli **RTL schémy** (*príloha C*) a **tabuľka spotreby FPGA zdrojov** (tab. 5.3) jednotlivých subsystémov, ktoré podporujú možnosť implementácie v práci navrhnutých modelov subsystémov konvolučnej siete do FPGA obvodu.

Záver

Rozsiahly nárast DoS/DDoS útokov na komunikačné siete a nimi spôsobené škody v poslednom období upozornili na problémy, týkajúce sa nedostatočnej kvality zabezpečenia súčasných počítačových sietí. Vzhľadom na deštruktívny dopad útokov sa detekcia a prevencia pred sieťovými útokmi stali jedným z kľúčových problémov, ktorým sa zaoberá súčasný výskum v oblasti zabezpečenia komunikačných sietí. Hľadaniu riešenia tohto problému, konkrétne problematike detekcie DoS/DDoS útokov, je venovaná aj predkladaná práca.

Rešeršou aktuálneho stavu v oblasti detekcie sieťových útokov typu DoS/DDoS a nových prístupov založených na strojovom učení sme dospeli k zhrnutiu hlavných nedostatkov súčasných detekčných metód a uviedli trendy v ich zdokonaľovaní (kap. 1.2.5). 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, jedného z najsľubnejších prístupov strojového učenia súčasnosti.

Na základe výsledkov vyššie uvedenej analýzy sme sformulovali metodiku návrhu detektora sieťových útokov (kap. 3), ktorú popisujeme v predkladanej práci. Metodika bola vybudovaná na piatich primárnych krokoch. Predlohou špecifikácie jednotlivých krokov metodiky sa stala oblasť rozpoznávania vzorov a postupy, všeobecne používané pri návrhu metód v tejto oblasti. Každý krok metodiky návrhu detektora bol navrhnutý so zámerom dosiahnuť čo najúčinnejšiu detekciu DoS/DDoS útokov v počítačovej sieti.

V prvom kroku (kap. 3.1) sú uvedené spôsoby získavania vzoriek s použitím dostupných datasetov a zberom dát priamo z reálnej sieťovej prevádzky. V tejto časti bol vykonaný rozbor sieťových datasetov a výber príznakov, ktoré slúžia ako dátový vstup do detektora. Predspracovaním vzoriek a ich transformáciou v druhom kroku (kap. 3.2) boli tieto naformátované tak, aby vyhovovali zvolenej metóde. Zdôvodnenie výberu hlbokého učenia, konkrétne konvolučnej neurónovej siete, a možnosti optimalizácie modelu trénovaním jeho parametrov boli riešené v treťom (kap. 3.3) a v štvrtom kroku (kap. 3.4). Aplikácia detektora v reálnej počítačovej sieti vyžaduje implementáciu jeho funkcií do niektorej

z existujúcich platforiem. Úloha výberu implementačnej platformy pre detektor sieťových útokov vo vysoko-rýchlostnej sieti bola riešená v piatom kroku (kap. 3.5). Ako dôvody pre voľbu obvodu FPGA boli použité jeho vlastnosti spolu so základnými technikami efektívneho návrhu. 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). Vzorová realizácia ostatných krokov metodiky bola z náplne práce vyňatá. V návrhu systému konvolučnej siete bol použitý systémový prístup, ktorý viedol ku špecifikácii subsystémov. Subsystémy zodpovedali jednotlivým typom vrstiev, použiteľným v modeli konvolučnej siete (kap. 1.3.1). Najväčšia pozornosť bola venovaná subsystému konvolučnej vrstvy a to z dôvodu jeho zložitosti a dôležitosti v porovnaní s ostatnými subsystémami. Originálny prístup k návrhu 2D konvolútora (kap. 4.1), ako kľúčového výpočtového prvku tohto subsystému, považujeme za jeden z kľúčových prínosov predkladanej práce. Pre popis návrhu subsystémov boli použité dva prístupy: modelovo-riadený prístup v nástroji Matlab/Simulink (kap. 5.2.1) a RTL prístup v jazyku VHDL (kap. 5.2.2). Priložené ilustrácie a diagramy jednotlivých návrhov viedli k lepšiemu pochopeniu štruktúry subsystémov. Funkčnosť jednotlivých subsystémov bola overená formou simulácie. Vykonané simulačné procedúry boli vyhodnotené porovnaním ich výstupných hodnôt s hodnotami referenčného modelu. Zhoda v správaní referenčného modelu a simulačných modelov potvrdila korektnú funkciu navrhnutých subsystémov.

V práci popísané subsystémy sú navrhnuté generický. Ich správanie je možné meniť zmenou hodnoty generických parametrov. Okrem rozmerov dátových vstupov sú nastaviteľné aj počty a rozmery vstupných a výstupných príznakových máp prislúchajúcej vrstvy. Použitím parametrov sú tak navrhnuté subsystémy uplatniteľné pre širokú množinu rôznych architektúr konvolučnej siete.

Metodika návrhu detektora tvorí základ riešenia, ale nepokrýva úplne problém so sieťovými útokmi typu DoS/DDoS v počítačovej sieti. Sieťové útoky sa vyvíjajú, a tak sa každý deň objavujú nové prípady doposiaľ neznámych sieťových útokov. Každý typ útoku je odlišný, a preto vyžaduje špecifickú architektúru konvolučnej siete, schopnú detegovať a klasifikovať útok. Návrh špecifickej architektúry konvolučnej siete, určenie jej parametrov a jej implementácia 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. Nové prístupy návrhu a popisu v práci uvedených subsystémov pre FPGA obvody s použitím vyšších programovacích jazykov, ako je napríklad *C/C++*

a *High-Level Synthesis*, by mohli viesť k zrozumiteľnejšiemu kódu, rýchlejšiemu testovaniu a teda aj k väčšej agilnosti implementácie subsystémov. Nové výkonnejšie FPGA obvody poskytujú viac interných zdrojov, najmä aritmetických jednotiek, čím sa otvára priestor pre úplne paralelnú implementáciu rozsiahlych modelov konvolučnej siete. Tento typ obvodu reprezentujú FPGA obvody s architektúrou *Adaptive Compute Acceleration Platform* (ACAP) od výrobcu Xilinx. Všetky vyššie uvedené body rozšírenia práce otvárajú možnosti ďalšieho výskumu.

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, L. 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.

Zoznam použitej literatúry

[1] C. Douligeris a A. Mitrokotsa, "Ddos attacks and defense mechanisms: Classification and state-of-the-art", *Computer Networks*, roč. 44, č. 5, s. 643–666, 2004, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2003.10.003. url: http://www.sciencedirect.com/science/article/pii/S1389128603004250.

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

[3] 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.

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

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

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

[7] S. T. Zargar, J. Joshi a D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks", *IEEE Communications Surveys and Tutorials*, roč. 15, č. 4, s. 2046–2069, 2013. DOI: 10.1109/SURV.2013.031413.00127. url: http://dblp.uni-trier.de/db/journals/comsur/comsur15.html#ZargarJT13.

[8] 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.

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

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

[11] M. D. Singh, "Analysis of Host-Based and Network-Based Intrusion Detection System", *I.J. Computer Network and Information Security*, roč. 8, č. 8, s. 41–47, 2014. DOI: 10.5815/ijcnis.2014.08.06.

[12] K. Letou, D. Devi a Y. J. Singh, "Host-based Intrusion Detection and Prevention System (HIDPS)", *International Journal of Computer Applications*, roč. 69, č. 26, 2013, ISSN: 0975 – 8887. DOI: 10.5120/12136-8419.

[13] G. Creech a J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns", *IEEE Transactions on Computers*, roč. 63, č. 4, s. 807–819, apr. 2014, ISSN: 0018-9340. DOI: 10.1109/TC.2013.13.

[14] M. Thottan a C. Ji, "Anomaly detection in ip networks", *IEEE Transactions on Signal Processing*, roč. 51, č. 8, s. 2191–2204, aug. 2003, ISSN: 1053-587X. DOI: 10.1109/TSP.2003.814797.

[15] 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.

[16] 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.

[17] J. Pearl, "Fusion, propagation, and structuring in belief networks", *Artif. Intell.*, roč. 29, č. 3, s. 241–288, sept. 1986, ISSN: 0004-3702. DOI: 10.1016/0004-3702(86)90072-X. url: http://dx.doi.org/10.1016/0004-3702(86)90072-X.

[18] Vikramkumar, Vijaykumar a Trilochan, "Bayes and naive bayes classifier", *CoRR*, roč. abs/1404.0933, 2014. arXiv: 1404.0933. url: http://arxiv.org/abs/1404.0933.

[19] 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.

[20] M. Alkasassbeh, G. Al-Naymat, A. B. Hassanat a M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques", *International Journal of Advanced Computer Science and Applications*, roč. 7, č. 1, 2016. DOI: 10.14569/IJACSA.2016.070159. url: http://dx.doi.org/10.14569/IJACSA.2016.070159.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] S. M. Erfani, S. Rajasegarar, S. Karunasekera a C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning", *Pattern Recogn.*, roč. 58, č. C, s. 121–134, okt. 2016, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2016.03.028. url: https://doi.org/10.1016/j.patcog.2016.03.028.

[29] 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.

[30] 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.

[31] M. M. Javidi a M. Hassan, "A new and quick method to detect dos attacks by neural networks", roč. 6, č. 2, s. 85–96, 2013, ISSN: ISSN 2008-949X. DOI: 10.22436/jmcs.06.02.01. url: http://dx.doi.org/10.22436/jmcs.06.02.01.

[32] 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.

[33] M. A. Garcia a T. Trinh, "Detecting simulated attacks in computer networks using resilient propagation artificial neural networks", en, *Polibits*, s. 5–10, jún 2015, ISSN: 1870-9044. url: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1870-90442015000100002&nrm=iso.

[34] 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.

[35] 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.

[36] 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.

[37] W. Pan a W. Li, "A hybrid neural network approach to the classification of novel attacks for intrusion detection", in *Parallel and Distributed Processing and Applications*, Y. Pan, D. Chen, M. Guo, J. Cao a J. Dongarra, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 564–575, ISBN: 978-3-540-32100-2.

[38] 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.

[39] D. Jiang, Y. Yang a M. Xia, "Research on intrusion detection based on an improved som neural network", in *2009 Fifth International Conference on Information Assurance and Security*, zv. 1, aug. 2009, s. 400–403. DOI: 10.1109/IAS.2009.247.

[40] K. Choksi, B. Shah a A. Ompriya Kale, "Intrusion Detection System using Self Organizing Map: A Survey", *Journal of Engineering Research and Applications www.ijera.com ISSN*, roč. 4, č. 4, s. 2248–962 211, 2014. url: www.ijera.com.

[41] 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.

[42] 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.

[43] 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.

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

[45] I. Goodfellow, Y. Bengio a A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[46] 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.

[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard a L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural Comput.*, roč. 1, č. 4, s. 541–551, dec. 1989, ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. url: http://dx.doi.org/10.1162/neco.1989.1.4.541.

[48] Y. LeCun, "Generalization and network design strategies", English (US), in *Connectionism in perspective*, R. Pfeifer, Z. Schreter, F. Fogelman a L. Steels, ed. Elsevier, 1989.

[49] 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.

[50] V. Dumoulin a F. Visin, "A guide to convolution arithmetic for deep learning", mar. 2016. arXiv: 1603.07285. url: http://arxiv.org/abs/1603.07285.

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke a A. Rabinovich, "Going deeper with convolutions", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jún 2015, s. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[52] A. Krizhevsky, I. Sutskever a G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou a K. Q. Weinberger, ed., Curran Associates, Inc., 2012, s. 1097–1105. url: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[53] S. Ioffe a C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *CoRR*, roč. abs/1502.03167, 2015. arXiv: 1502.03167. url: http://arxiv.org/abs/1502.03167.

[54] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever a R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *CoRR*, roč. abs/1207.0580, 2012. arXiv: 1207.0580. url: http://arxiv.org/abs/1207.0580.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever a R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.*, roč. 15, č. 1, s. 1929–1958, jan. 2014, ISSN: 1532-4435. url: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[56] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, roč. 36, č. 4, s. 193–202, apr. 1980, ISSN: 1432-0770. DOI: 10.1007/BF00344251. url: https://doi.org/10.1007/BF00344251.

[57] Y. L. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel a D. Henderson, "Advances in neural information processing systems 2", in, D. S. Touretzky, ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, kap. Handwritten Digit Recognition with a Back-propagation Network, s. 396–404, ISBN: 1-55860-100-7. url: http://dl.acm.org/citation.cfm?id=109230.109279.

[58] L. Tóth, "Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition", in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, máj 2014, s. 190–194. DOI: 10.1109/ICASSP.2014.6853584.

[59] 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.

[60] 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.

[61] 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.

[62] P. Mobadersany, S. Yousefi, M. Amgad, D. A. Gutman, J. S. Barnholtz-Sloan, J. E. Velázquez Vega, D. J. Brat a L. A. D. Cooper, "Predicting cancer outcomes from histology and genomics using convolutional networks", *Proceedings of the National Academy of Sciences*, 2018, ISSN: 0027-8424. DOI: 10.1073/pnas.1717139115. eprint: http://www.pnas.org/content/early/2018/03/09/1717139115.full.pdf. url: http://www.pnas.org/content/early/2018/03/09/1717139115.

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

[64] 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/.

[65] D. J. Hwang, S. H. Cho, Y. D. Kim a S. Y. Han, "Exploiting spatial and temporal parallelism in the multithreaded node architecture implemented on superscalar risc processors", in *1993 International Conference on Parallel Processing - ICPP'93*, zv. 1, aug. 1993, s. 51–54. DOI: 10.1109/ICPP.1993.85.

[66] D. Keitel-Schulz a N. Wehn, "Embedded dram development: Technology, physical design, and application issues", *IEEE Design Test of Computers*, roč. 18, č. 3, s. 7–15, máj 2001, ISSN: 0740-7475. DOI: 10.1109/54.922799.

[67] J. Jeddeloh a B. Keeth, "Hybrid memory cube new dram architecture increases density and performance", *2012 Symposium on VLSI Technology (VLSIT)*, s. 87–88, 2012.

[68] J. Zhang, S. Khoram a J. Li, "Boosting the performance of fpga-based graph processor using hybrid memory cube: A case for breadth first search", in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ed. FPGA '17, Monterey, California, USA: ACM, 2017, s. 207–216, ISBN:

978-1-4503-4354-1. DOI: 10.1145/3020078.3021737. url: http://doi.acm.org/10.1145/3020078.3021737.

[69] D. B. Strukov, G. S. Snider, D. R. Stewart a R. S. Williams, "The missing memristor found", *Nature*, roč. 453, č. 7191, s. 80–83, 2008.

[70] 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.

[71] M. H. Bhuyan, D. K. Bhattacharyya a J. K. Kalita, "Towards generating real-life datasets for network intrusion detection", *I. J. Network Security*, roč. 17, s. 683–701, 2015.

[72] I. Sharafaldin, A. H. Lashkari a A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC, SciTePress, 2018, s. 108–116, ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116.

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

[74] V. Jacobson, C. Leres a S. McCanne. (2010). Tcpdump official webpage, url: http://www.tcpdump.org/ (cit. 19. 04. 2018).

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

[76] 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.

[77] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis a P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project", in *In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, IEEE Computer Press, s. 130–144.

[78] 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.

[79] J. Dromard, G. Roudière a P. Owezarski, "Online and scalable unsupervised network anomaly detection method", *IEEE Transactions on Network and Service Management*, roč. 14, č. 1, s. 34–47, mar. 2017, ISSN: 1932-4537. DOI: 10.1109/TNSM.2016.2627340.

[80] J. Dromard, G. Roudière a P. Owezarski, "Unsupervised network anomaly detection in real-time on big data", in *New Trends in Databases and Information Systems*, T. Morzy, P. Valduriez a L. Bellatreche, ed., Cham: Springer International Publishing, 2015, s. 197–206, ISBN: 978-3-319-23201-0.

[81] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu a J. Hu, "Detection of denial-of-service attacks based on computer vision techniques", *IEEE Transactions on Computers*, roč. 64, č. 9, s. 2519–2533, sept. 2015, ISSN: 0018-9340. DOI: 10.1109/TC.2014.2375218.

[82] S. S. Kim a A. L. N. Reddy, "Netviewer: A network traffic visualization and analysis tool", in *Proceedings of the 19th Conference on Large Installation System Administration Conference - Volume 19*, ed. LISA '05, San Diego, CA: USENIX Association, 2005, s. 18–18. url: http://dl.acm.org/citation.cfm?id=1251150.1251168.

[83] R. Fontugne, T. Hirotsu a K. Fukuda, "An image processing approach to traffic anomaly detection", in *Proceedings of the 4th Asian Conference on Internet Engineering*, ed. AINTEC '08, Pratunam, Bangkok, Thailand: ACM, 2008, s. 17–26, ISBN: 978-1-60558-127-9. DOI: 10.1145/1503370.1503377. url: http://doi.acm.org/10.1145/1503370.1503377.

[84] T. H. Kim, D. S. Kim a H. Y. Jung, "Defending against ddos attacks under ip spoofing using image processing approach", *IEICE Transactions on Communications*, roč. E99.B, č. 7, s. 1511–1522, 2016. DOI: 10.1587/transcom.2015EBP3457.

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

[86] L. Deng a D. Yu, *Deep Learning: Methods and Applications*, ed. Foundations and trends in signal processing. Now Publishers, 2014, ISBN: 9781601988140. url: https://books.google.sk/books?id=46qNoAEACAAJ.

[87] 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.

[88] 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.

[89] 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.

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

[91] 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.

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

[93] Y. Zhou, S. Redkar a X. Huang, "Deep learning binary neural network on an fpga", in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, aug. 2017, s. 281–284. DOI: 10.1109/MWSCAS.2017.8052915.

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

[95] 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.

[96] 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.

[97] J. J. Lee a G. Y. Song, "Super-systolic array for 2d convolution", in *TENCON 2006 - 2006 IEEE Region 10 Conference*, nov. 2006, s. 1–4. DOI: 10.1109/TENCON.2006.343739.

[98] C. Farabet, C. Poulet, J. Y. Han a Y. LeCun, "Cnp: An fpga-based processor for convolutional networks", in *2009 International Conference on Field Programmable Logic and Applications*, aug. 2009, s. 32–37. DOI: 10.1109/FPL.2009.5272559.

[99] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis a M. A. Horowitz, "Convolution engine: Balancing efficiency & flexibility in specialized computing", *SIGARCH Comput. Archit. News*, roč. 41, č. 3, s. 24–35, jún 2013, ISSN: 0163-5964. DOI: 10.1145/2508148.2485925. url: http://doi.acm.org/10.1145/2508148.2485925.

[100] 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.

[101] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang a H. Yang, "Going deeper with embedded fpga platform for convolutional neural network", in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ed. FPGA '16, Monterey, California, USA: ACM, 2016, s. 26–35, ISBN: 978-1-4503-3856-1. DOI: 10.1145/2847263.2847265. url: http://doi.acm.org/10.1145/2847263.2847265.

[102] 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.

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

[104] ——, *Ug073 xtremedsp for virtex-4 fpgas user guide*, 2008.

[105] 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.

[106] 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.

[107] W. Lu, G. Yan, J. Li, S. Gong, Y. Han a X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks", in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, feb. 2017, s. 553–564. DOI: 10.1109/HPCA.2017.29.

[108] A. Tisan, S. Oniga, D. Mic a B. Attila, "Digital implementation of the sigmoid function for fpga circuits", *ACTA TECHNICA NAPOCENSIS Electronics and Telecommunications*, roč. 50, jan. 2009.

[109] T. Jamel a B. Mohammed, "Implementation of a sigmoid activation function for neural network using fpga", apr. 2012.

[110] F. Ortega-Zamorano, J. M. Jerez, G. Juárez, J. O. Pérez a L. Franco, "High precision fpga implementation of neural network activation functions", in *2014 IEEE Symposium on Intelligent Embedded Systems (IES)*, dec. 2014, s. 55–60. DOI: 10.1109/INTELES.2014.7008986.

[111] B. Yuan, "Efficient hardware architecture of softmax layer in deep neural network", in *2016 29th IEEE International System-on-Chip Conference (SOCC)*, sept. 2016, s. 323–326. DOI: 10.1109/SOCC.2016.7905501.

[112] S. Pan, Z. Li, Y. Huang a W. Lin, "Fpga realization of activation function for neural network", in *2018 7th International Symposium on Next Generation Electronics (ISNE)*, máj 2018, s. 1–2. DOI: 10.1109/ISNE.2018.8394695.

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

[114] N. Kavvadias a K. Masselos, "Design of fixed-point rounding operators for the vhdl-2008 standard", in *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing*, okt. 2012, s. 1–8.

[115] R. Yates, "Fixed-Point Arithmetic: An Introduction", Digital Signal Labs, tech. spr., jan. 2013, s. 1–15. url: http://www.digitalsignallabs.com/downloads/fp.pdf.

[116] Xilinx, *Vivado Design Suite User Guide: Synthesis*, jún 2018. url: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug901-vivado-synthesis.pdf.

[117] 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.

[118] Xilinx, *UG474 7 Series FPGAs Configurable Logic Block User Guide*, sept. 2016. url: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf.

[119] ——, *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.

[120] ——, *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.

Prílohy

Príloha A

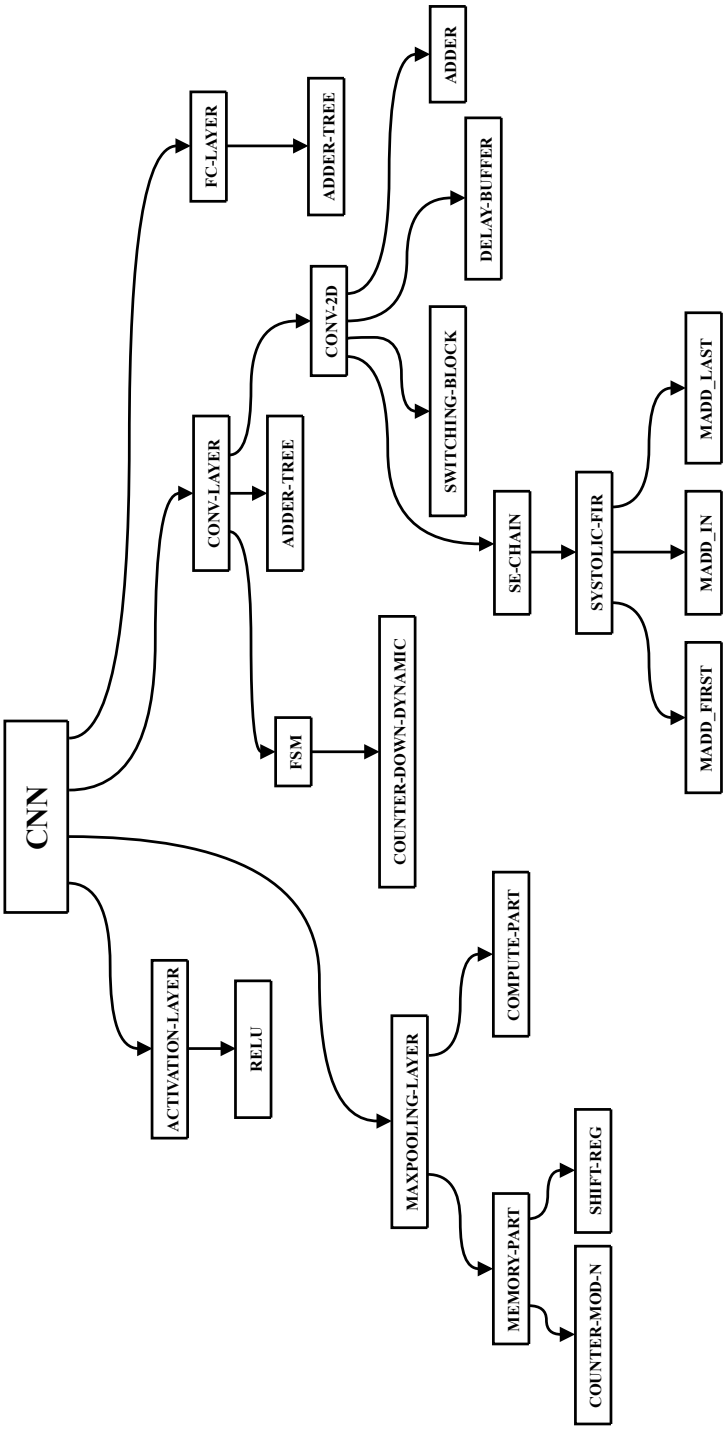
Obsah priloženého kompaktného disku

Priložený kompaktný disk obsahuje:

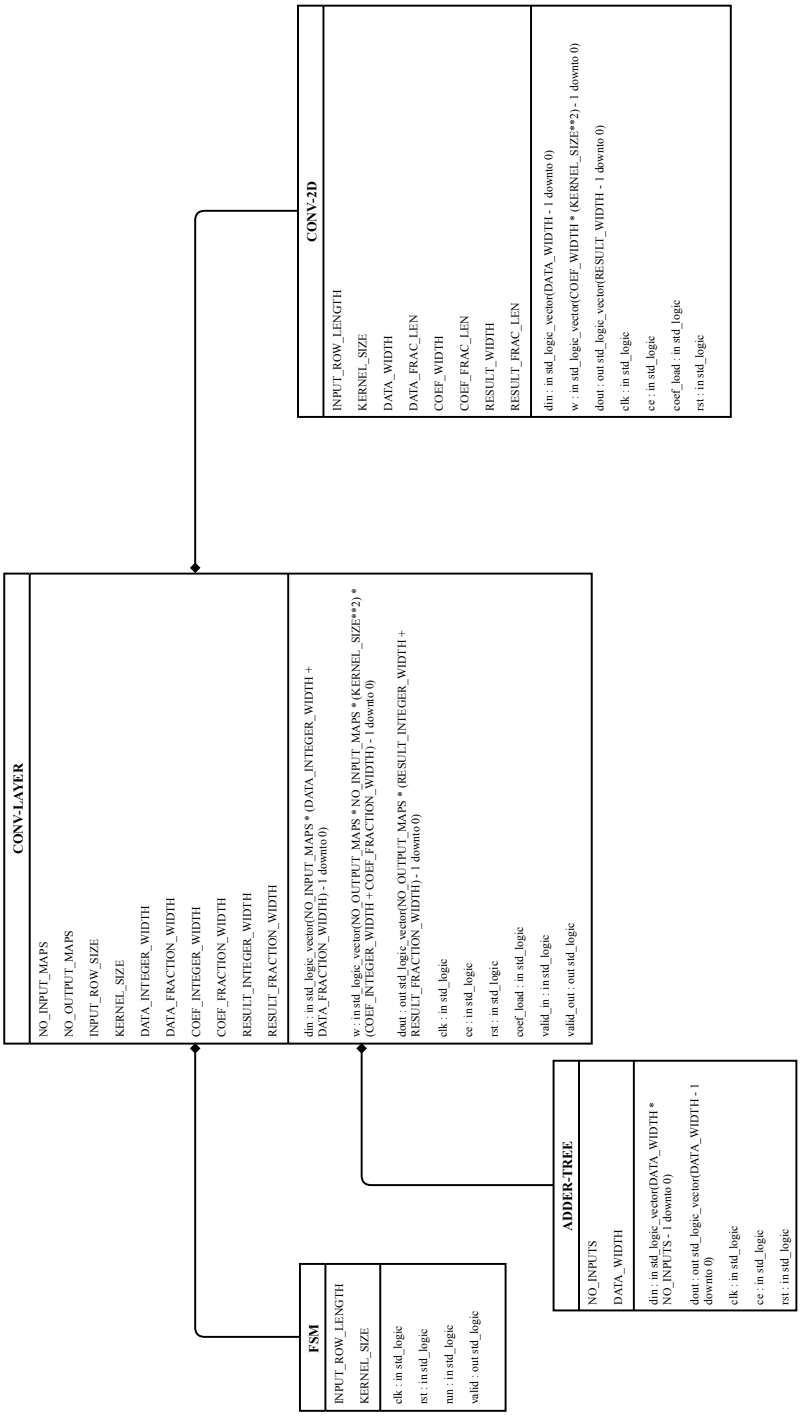
- Práca v elektronickej podobe (formát PDF)
- Zdrojové súbory Matlab/Simulink a VHDL, popisujúce v práci navrhnutý model konvolučnej siete
- Diagramy hierarchickej štruktúry entít a prislúchajúce RTL schémy

Príloha B

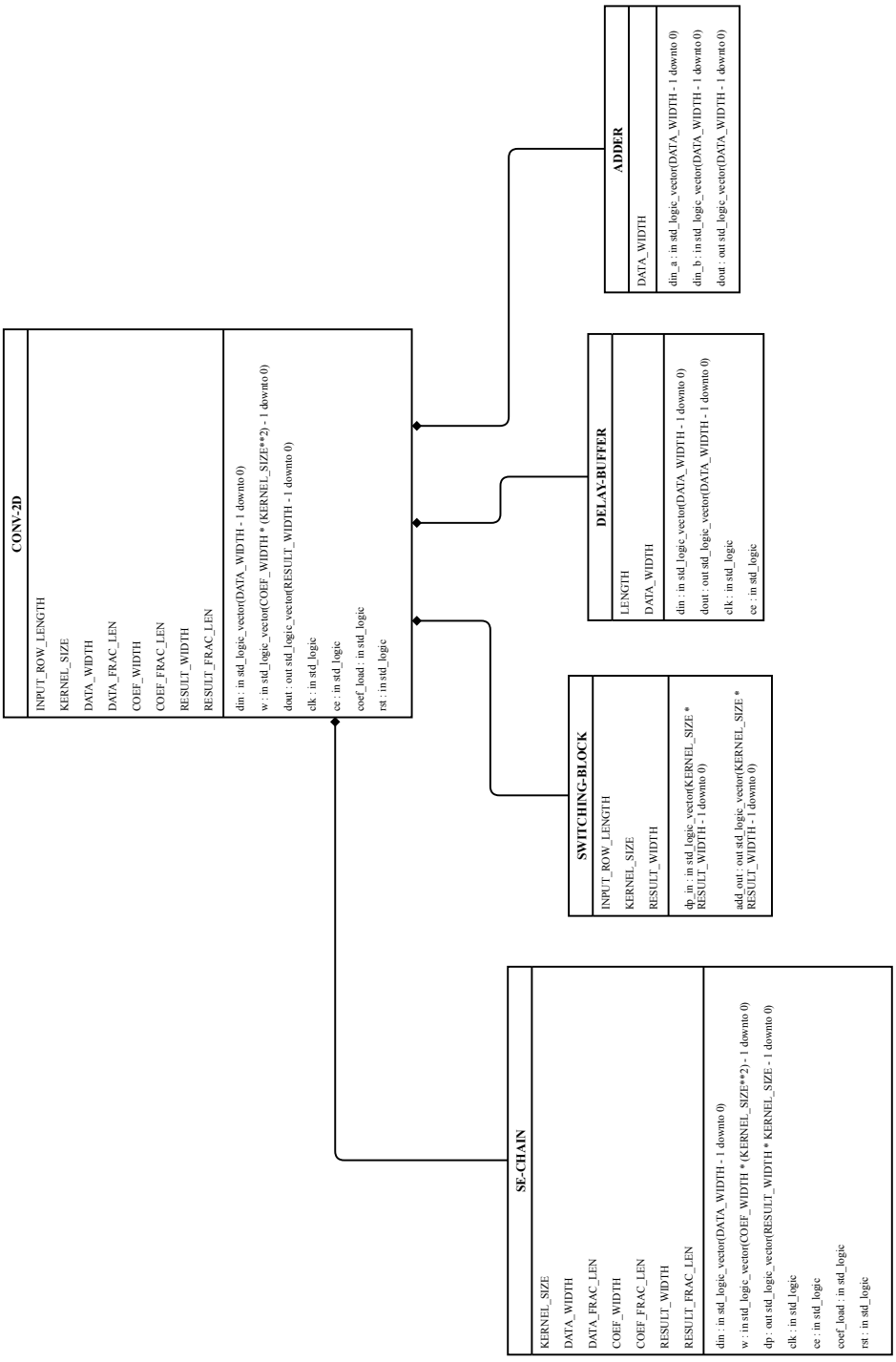
Diagramy entít a ich rozhrania



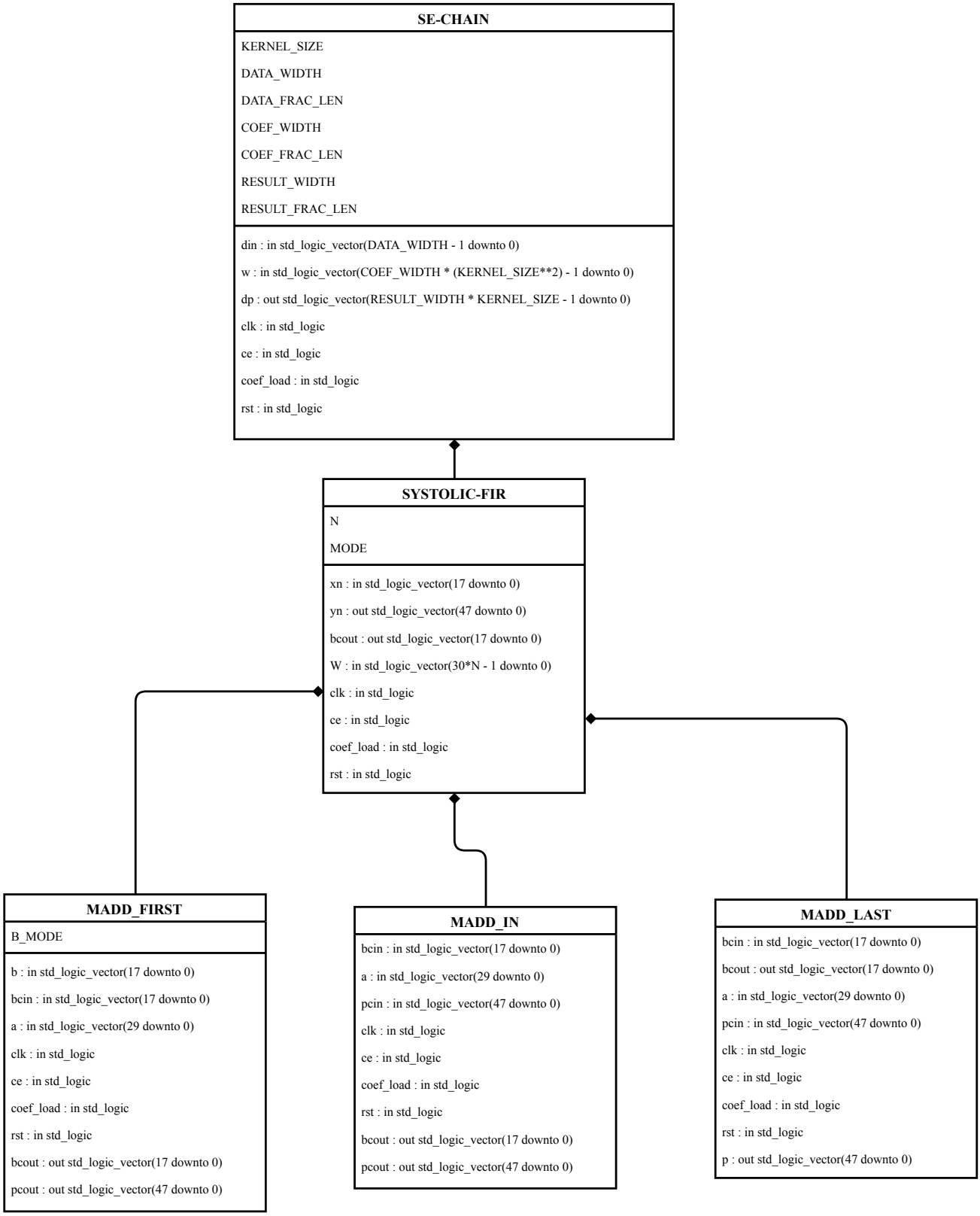
Obr. B.1: Hierarchická štruktúra navrhnutého VHDL modelu konvolučnej siete



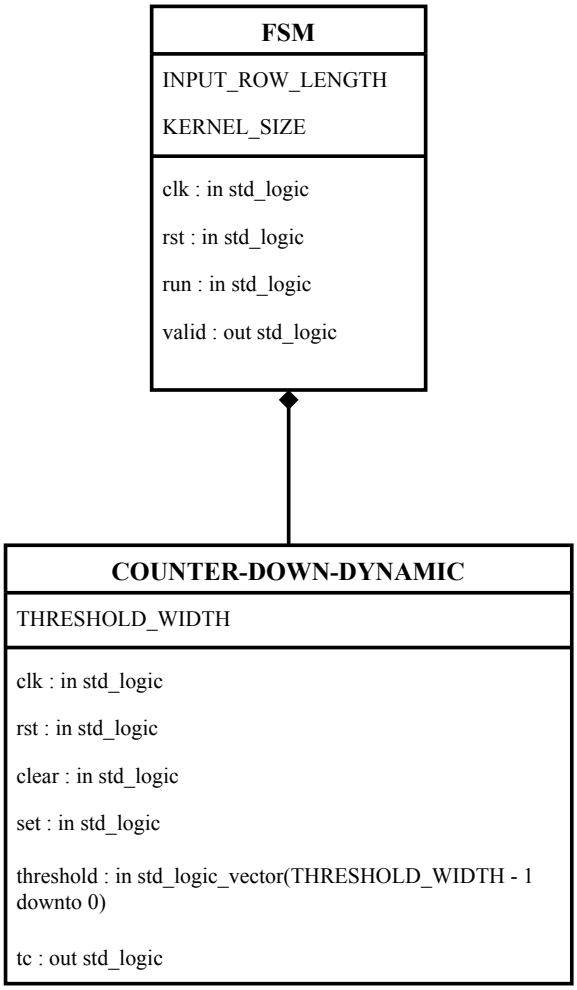
Obr. B.2: Rozhrania entít konvolučnej vrstvy



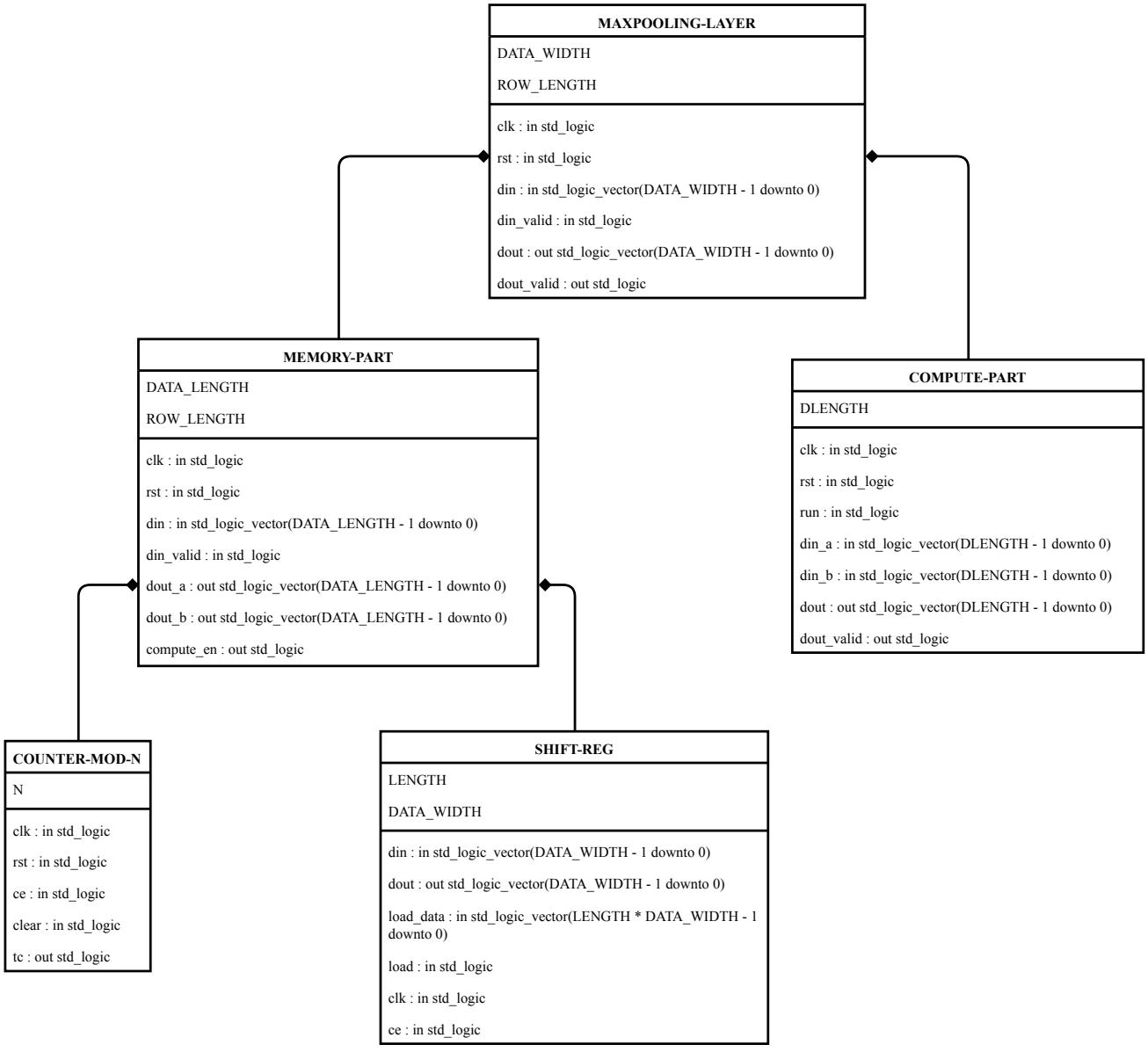
Obr. B.3: Rozhrania entít 2D konvolútora



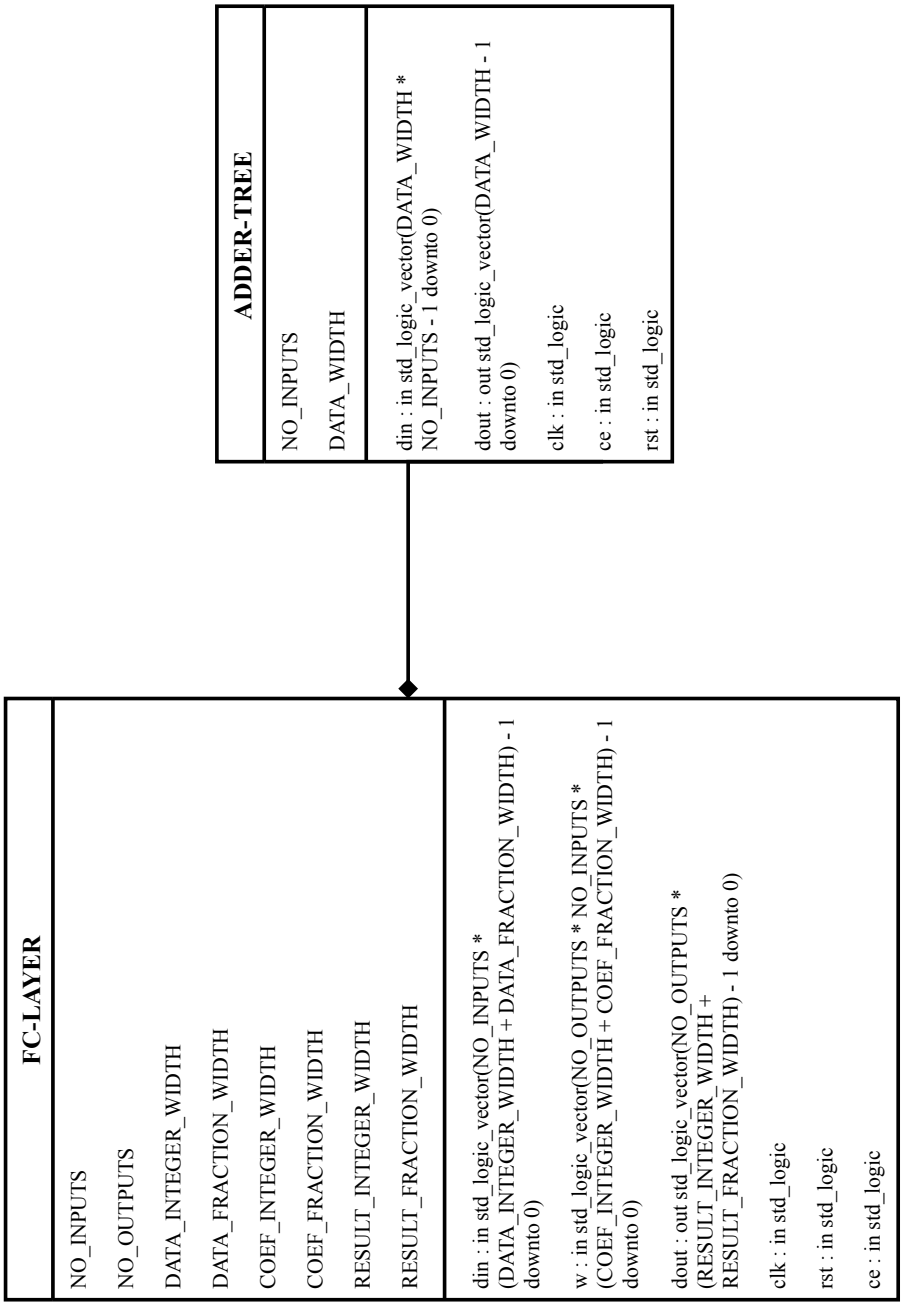
Obr. B.4: Rozhrania entít reťaze 1D konvolútorov



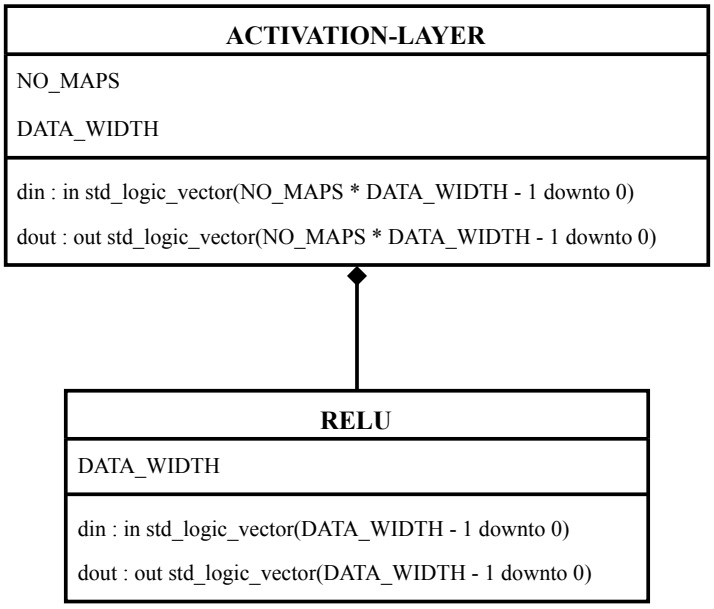
Obr. B.5: Rozhrania entít riadiacej časti 2D konvolútora



Obr. B.6: Rozhrania entít zlučovacej vrstvy



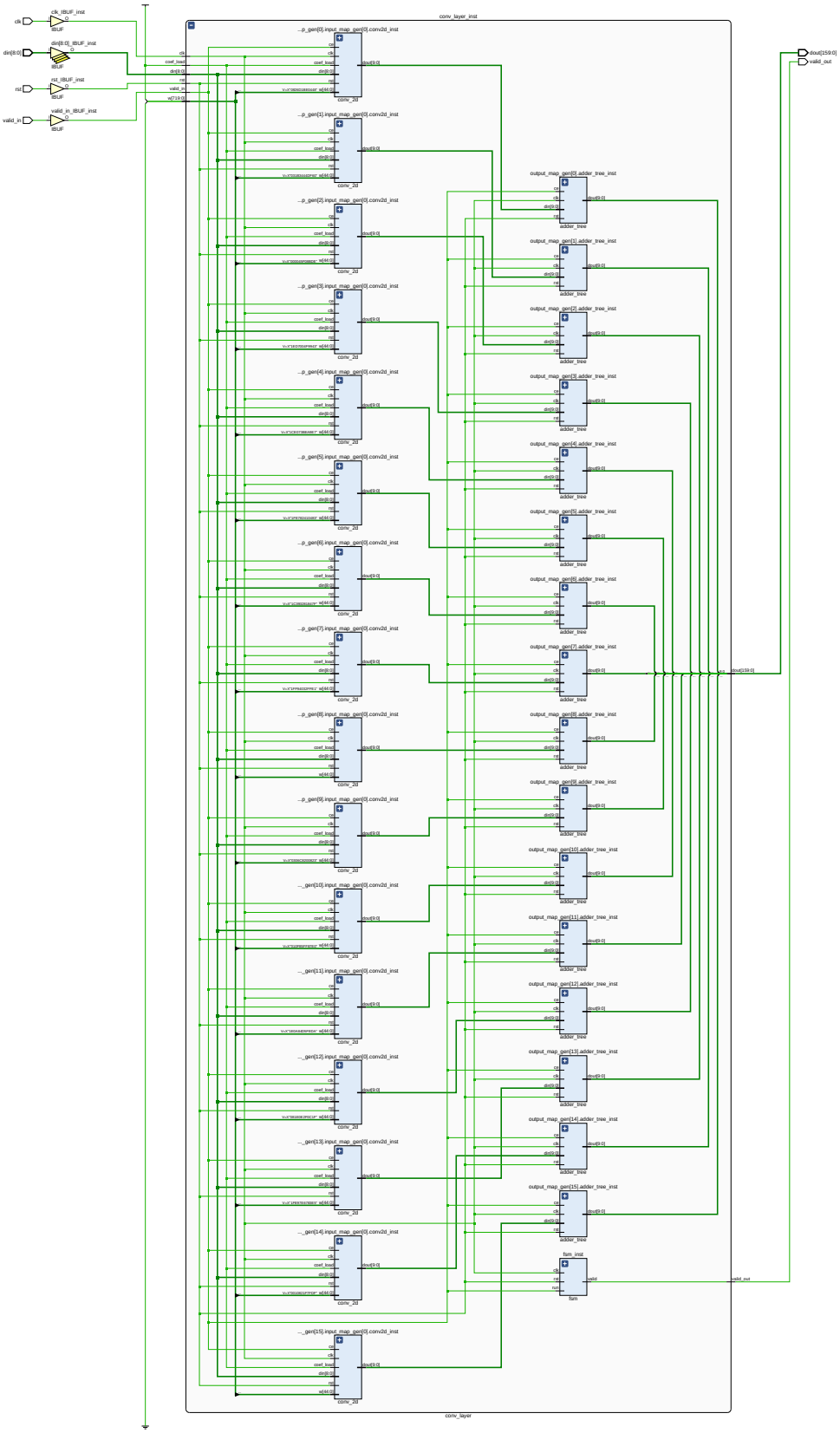
Obr. B.7: Rozhrania entít plne-prepojenej vrstvy



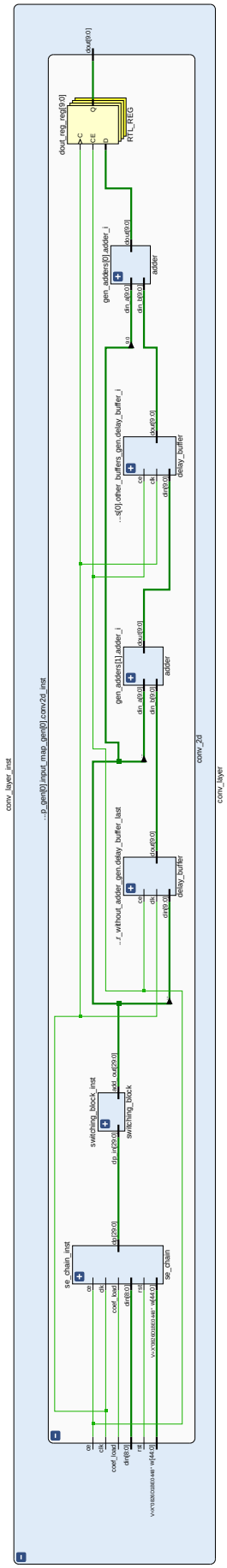
Obr. B.8: Rozhrania entít aktivačnej vrstvy

Príloha C

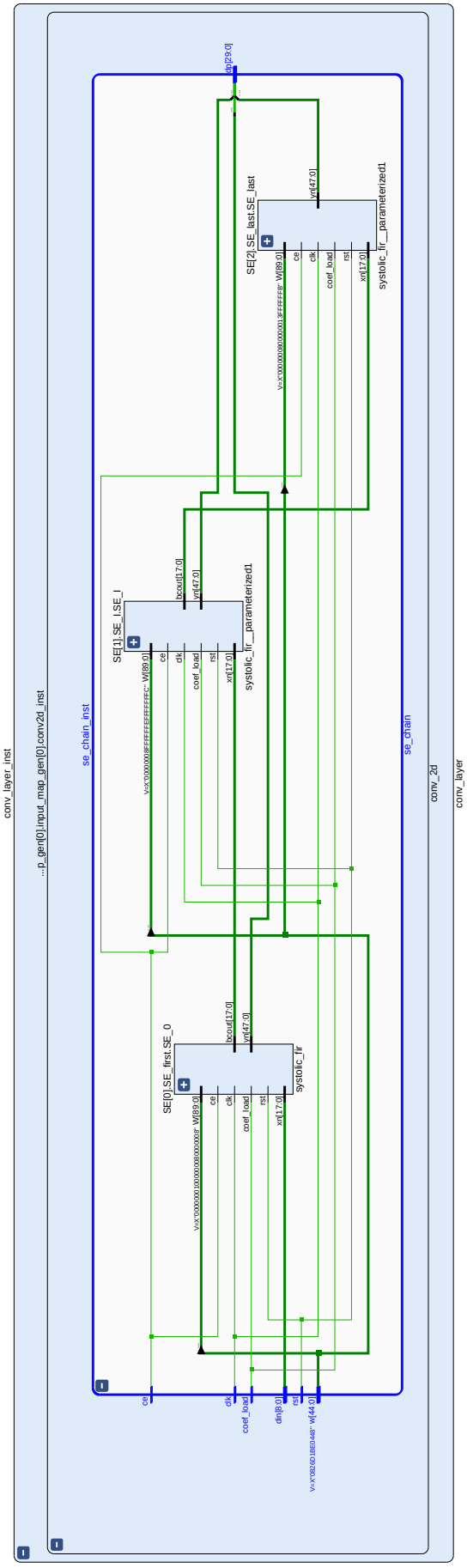
RTL schémy entít



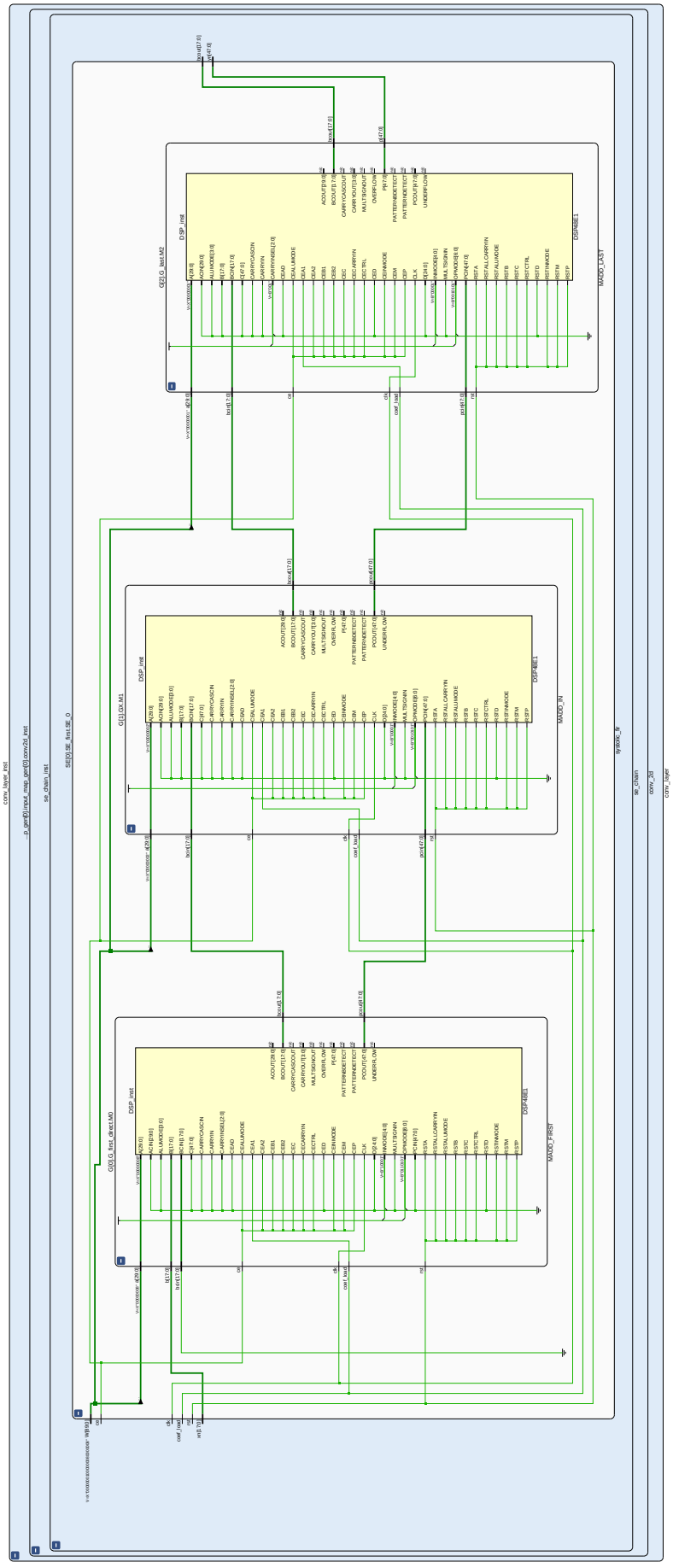
Obr. C.1: RTL schéma subsystému konvolučnej vrstvy



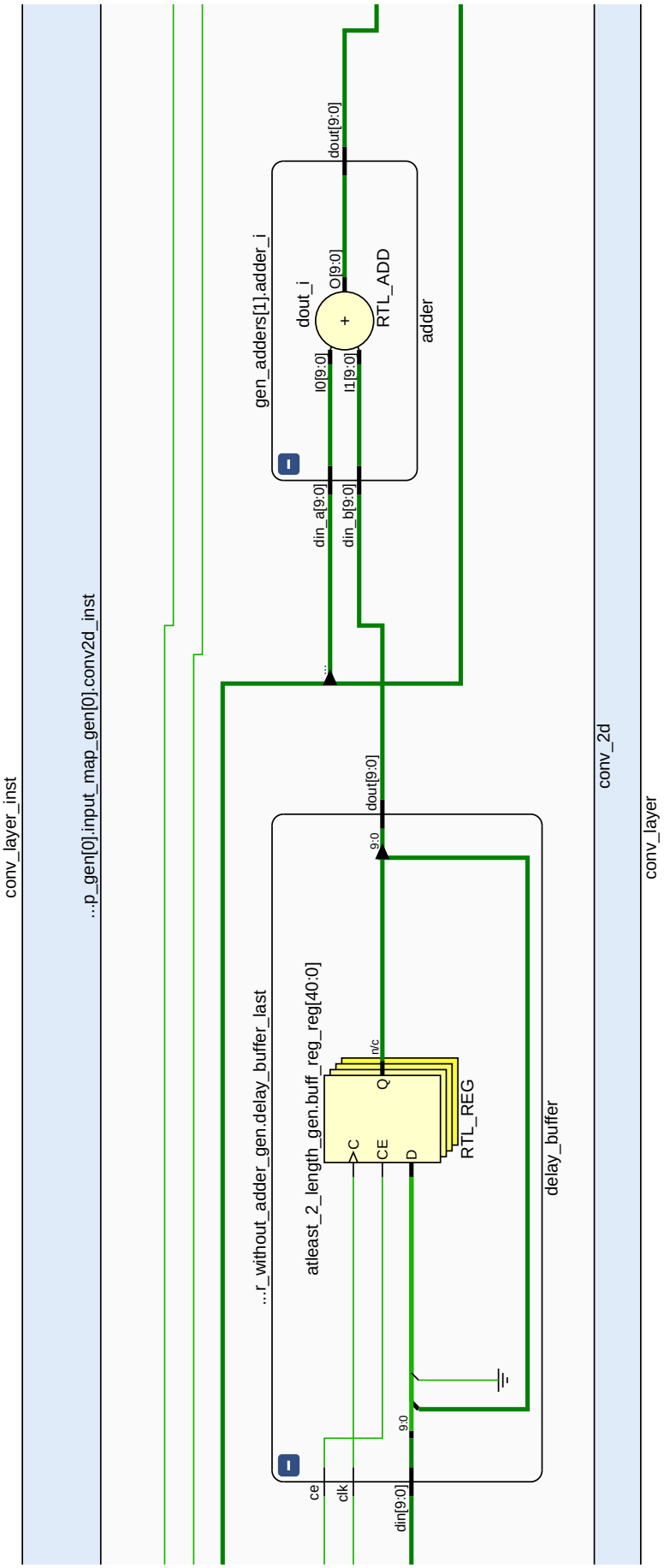
Obr. C.2: RTL schéma 2D konvolútora



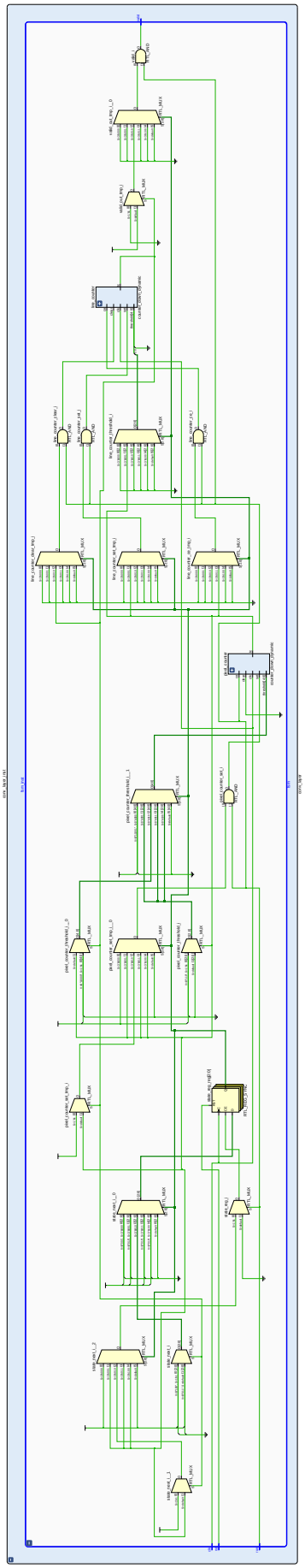
Obr. C.3: RTL schéma prvej časti 2D konvolútora - reťaz 1D konvolútorov



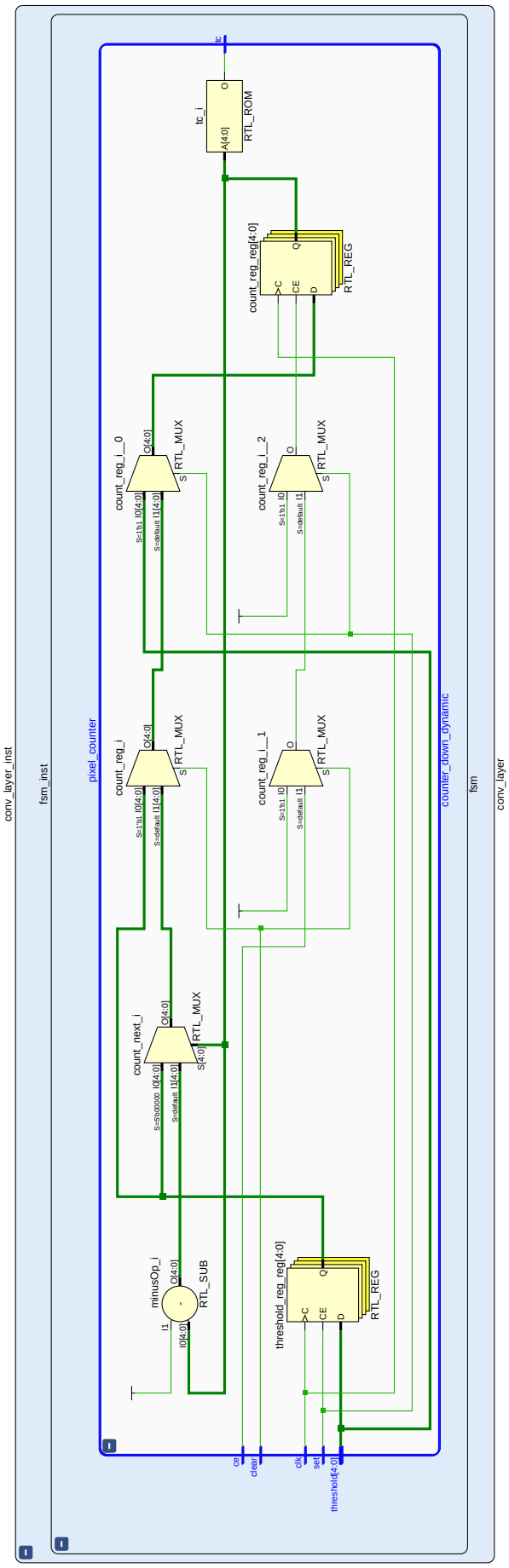
Obr. C.4: RTL schéma 1D konvolútora



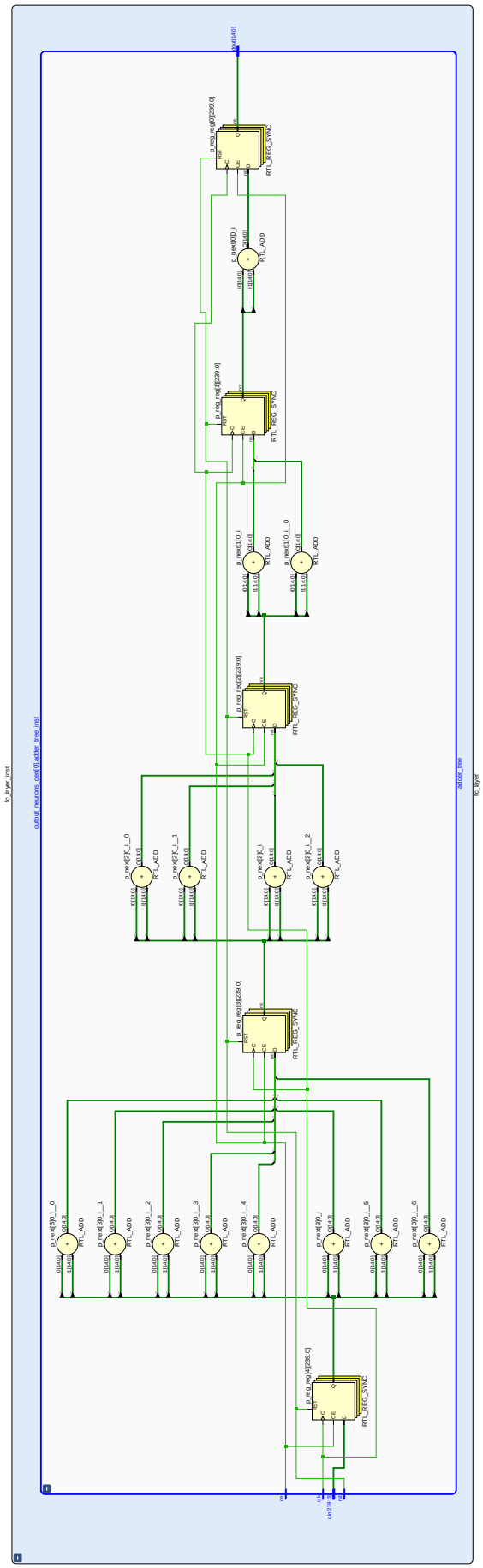
Obr. C.5: RTL schéma druhej časti 2D konvolútora - oneskorovacie registre a sčítačky



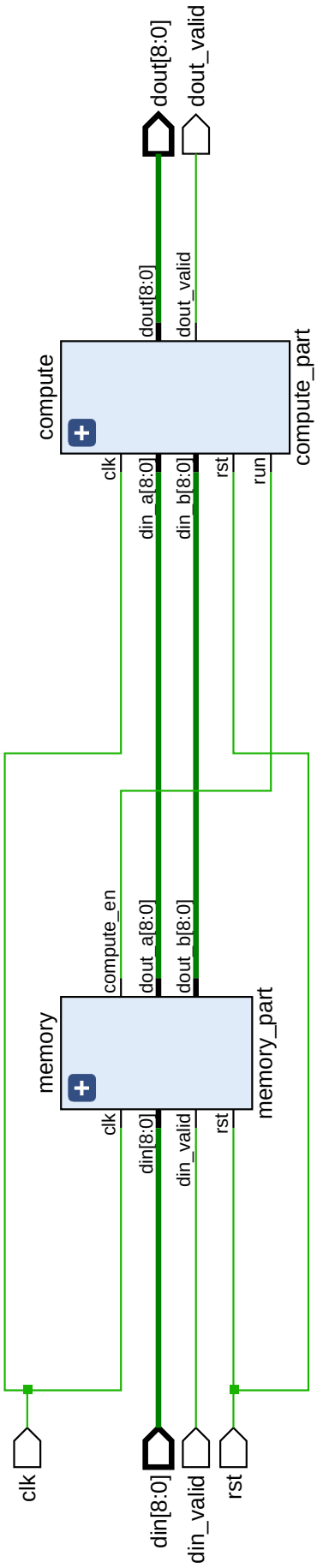
Obr. C.6: RTL schéma riadiacej časti 2D konvolútora



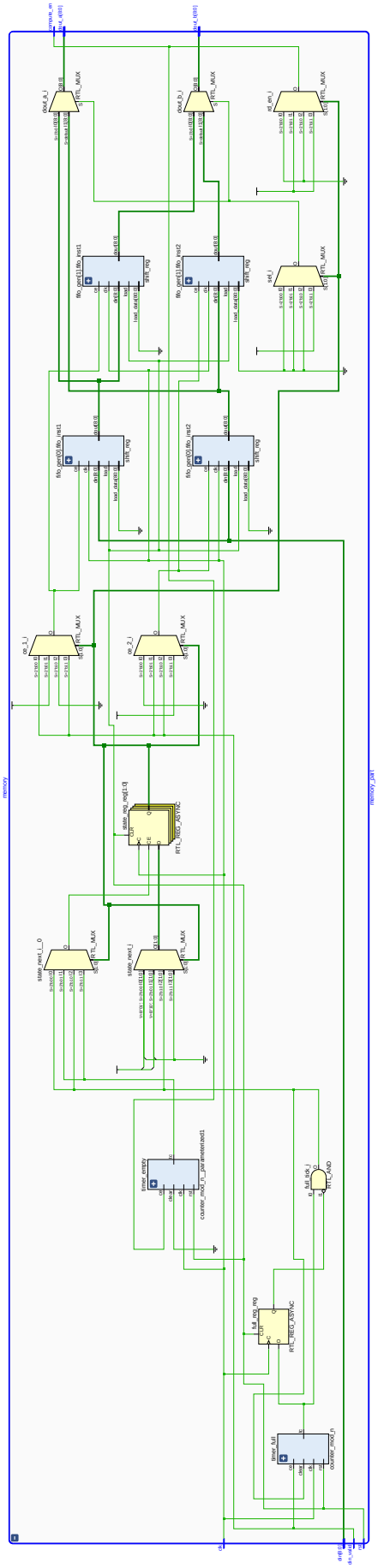
Obr. C.7: RTL schéma interne-použitého počítadla v FSM 2D konvolútora



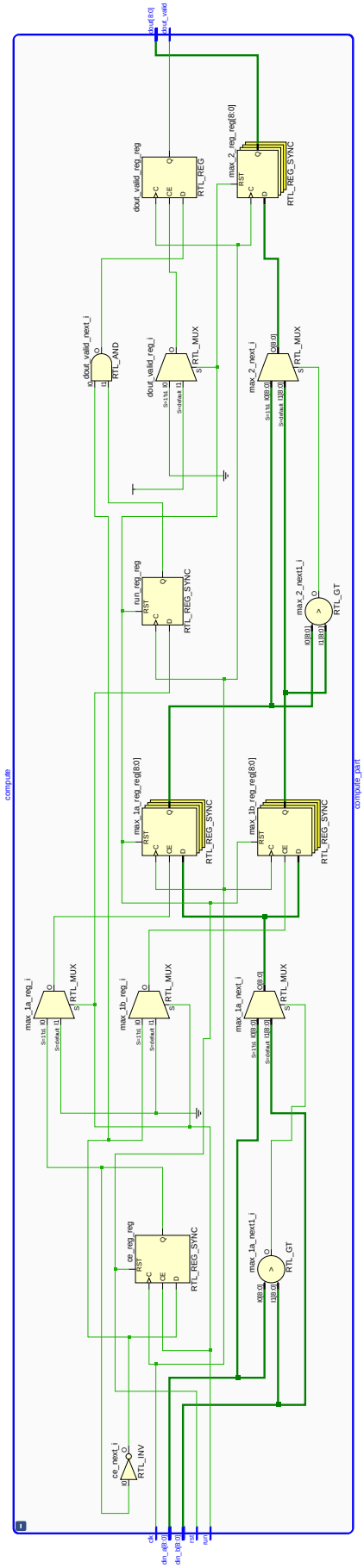
Obr. C.8: RTL schéma súčtového stromu



Obr. C.9: RTL schéma zlučovacej vrstvy



Obr. C.10: RTL schéma pamäťovej časti subsystému zlučovacej vrstvy



Obr. C.11: RTL schéma výpočtovej časti subsystému zlučovacej vrstvy