Vývoj Aplikačního nástroje s využitím měření WAMS a neuronových sítí pro klasifikaci provozních stavů v přenosové soustavě

Pracoviště: KEE
Číslo dokumentu: 22190-038-2024
Typ zprávy: Výzkumná zpráva
Řešitelé: Ing. Václav Mužík, Ph.D., doc Ing. Karel Noháč, Ph.D., Ing. Mikhail Olkhovskyi, Ph.D., Ing. et Ing. Martin Vinš, Ing. Filip Zmeko
Vedoucí projektu: doc. Ing. Karel Noháč, Ph.D.
Počet stran: 49
Datum vydání: 5. 2. 2025
Oborové zařazení: 2.2 Electrical engineering, Electronic engineering, Information engineering - Electrical and electronic engineering

Zadavatel / zákazník:

Zpracovatel / dodavatel:
Západočeská univerzita v Plzni
Research and Innovation Centre for Electrical Engineering
Univerzitní 8
306 14 Plzeň
Kontaktní osoba:
doc Ing. Karel Noháč, Ph.D.
tel. +420 377 634 343
nohac@fel.zcu.cz

Tato zpráva vznikla s podporou projektu TAČR NCK TN02000025 NCE II.

Anotace

Tato výzkumná zpráva popisuje vývoj algoritmu neuronových sítí pro detekci provozních stavů v přenosové soustavě na základě dat z PMU. Hlavní vývoj probíhal v pythonu, s MATLABem jako verifikačním nástrojem pro snadnější zpracování a vizualizaci dat. Algoritmus spolehlivě rozpoznává klíčové stavy, jako zkraty či skokové změny zatížení, přičemž výsledky jsou interpretovány prostřednictvím grafické vizualizace. Další zlepšení přesnosti vyžaduje rozšíření trénovacích dat, zejména pro omezení falešně pozitivních detekcí.

Klíčová slova

Algoritmus, Detekce, MATLAB, Neuronová síť, Porucha, python, WAMS

Report title

Development of an Application Tool using WAMS measurements and neural networks for classifying operating states in the transmission system

Abstract

This research report describes the development of a neural network algorithm for detecting operational states in a transmission system based on PMU data. The main development was done in python, with MATLAB as a verification tool for easier data processing and visualization. The algorithm reliably recognizes key states, such as short circuits or load step changes, and the results are interpreted via a graphical visualisation. Further improvement of accuracy requires expanding the training data, especially to reduce false positive detections.

Keywords

Algorithm, Detection, Fault, MATLAB, Neural Network, python, WAMS

Obsah

1 Úvod

V rámci projektu TAČR TN02000025 - “Národní centrum pro energetiku II (NCE II)”, konkrétně pracovního balíčku “Nové analytické nástroje pro využití dat z WAMS” byly provedeny rozsáhlé analytické, experimentální a tvůrčí činnosti zaměřené na inovativní přístup ke zpracovávání a využívání komplexních přesných měření WAMS (Wide Area Measurement System) dostupných aplikačním garantem a projektovým participantem, společností ČEPS, a.s.

Základní koncepční myšlenkou bylo inovativním způsobem přistoupit k potenciálu těchto měření s ohledem na pokrok v možnostech nabízených nejen vyšším výkonem, ale i pokročilými metodami, které jsou aktuálně dostupné informačními systémy. Klíčový rozdíl mezi zmíněnými měřeními WAMS a tradičními technologiemi (často oborově nazývanými SCADA) je vysoká přesnost jak velikosti vyhodnocených veličin, tak i časového zařazení (obvykle nazývaného časové razítko). Přesnost měření sama o sobě implikuje nové možnosti dalšího zpracování a dedukcí, které byly dříve s ohledem na špatnou podmíněnost úlohy neproveditelné. Časová synchronizace však otevírá prostor pro nacházení relevantních souvislostí měření z více lokalit, tedy integrovat měření do celků v rámci celé národní přenosové soustavy, nebo dokonce ze všech soustav propojených do kooperujících v rámci hlavní evropské soustavy.

V první fázi řešení bylo nezbytné zmapovat možnosti, které technologie nabízí a jaké jsou zkušenosti s teoretickým i praktickým využitím v akademickém i průmyslovém prostředí. Výsledkem tohoto procesu byla rozsáhlá analytická zpráva a databáze dostupných zdrojových dokumentů s vazbou na toto téma. Akademické pracoviště ZČU má se zpracováním dat WAMS již historickou zkušenost, v té době však možnosti zpracování byly technicky omezené a aplikační garant tehdy ještě neměl vlastní měřící jednotky instalované.

V druhé fázi bylo zhodnoceno, které z aktuálně nabízených možností mají význam s ohledem na potenciál inovativnosti a budoucí upotřebitelnosti u aplikačního garanta. Postupným zužováním při respektování možností akademického prostředí a technických podmínek aplikačního garanta byly vybrány dva hlavní směry dalšího zaměření postupu. Jedná se o tato témata:

- Detekce, identifikace a hodnocení událostí s možným vlivem na stabilitu ES
- Detekce a rozbor rozsáhlých nízkofrekvenčních výkonových kyvů v ES

Ačkoli při prvotním zpracování a získávání dat měly další technické práce určité kroky společné a šlo postupy sdílet, přesto většina dalšího řešení těchto témat probíhala spíše samostatně. V obou případech bylo klíčové získat relevantní data dokumentující výskyt příslušného jevu.

V dalším tetu bude rekapitulován postup a výsledky prvního rozsáhlejšího tématu, přičemž druhé téma je předmětem jiné doplňující zprávy s ID 43945503.

2 Koncept hodnocení událostí s možným vlivem na stabilitu ES

Jak již bylo řečeno, schopnost vyvinou relativně všeobecnou metodu, potažmo softwarový nástroj, se odvíjí od dostupnosti datových podkladů kvalitně vypovídajících o příslušném fenoménu. V rané fázi se vývoj nejprve opíral o data vytipovaná a získaná na základě odborné zkušenosti a intuice obsluhy originálního technického zázemí dodavatele měřících jednotek WAMS. Takový postup nepřinášel větší množství podkladů, byl časové náročný a výběr časových dat nebyl systematický.

Z principu technického řešení dodavatele a rozsáhlosti dat není možné pro selekci vhodných okamžiků používat přímo vlastní časová data. S respektem k těmto okolnostem bylo přistoupeno k tvorbě metodického mechanismu na základě výpisu událostí a hlášení, které jako komplikovaný doprovodný text je rovněž výstupem instalovaného systému. I v tomto případě se jedná o relativně rozsáhlý datový podklad, v kterém není možné se snadno zorientovat. Bylo tedy připraveno několik algoritmických postupů pro strojové zpracování, přičemž jako nejpraktičtější se ukázal koncept hodnocení unikátnosti (výjimečnosti v rámci hodnoceného období), která určovala prioritu zaměření další analýzy na příslušný časový interval a lokalitu. Na základě prvotní informace z výpisu hlášení byl připraven nástroj schopný na půdě aplikačního garanta vybrat relevantní časová data o přiměřeném datovém rozsahu obsahující s vysokou pravděpodobností záznam události, která je svým významem hodna dalšího zpracování.

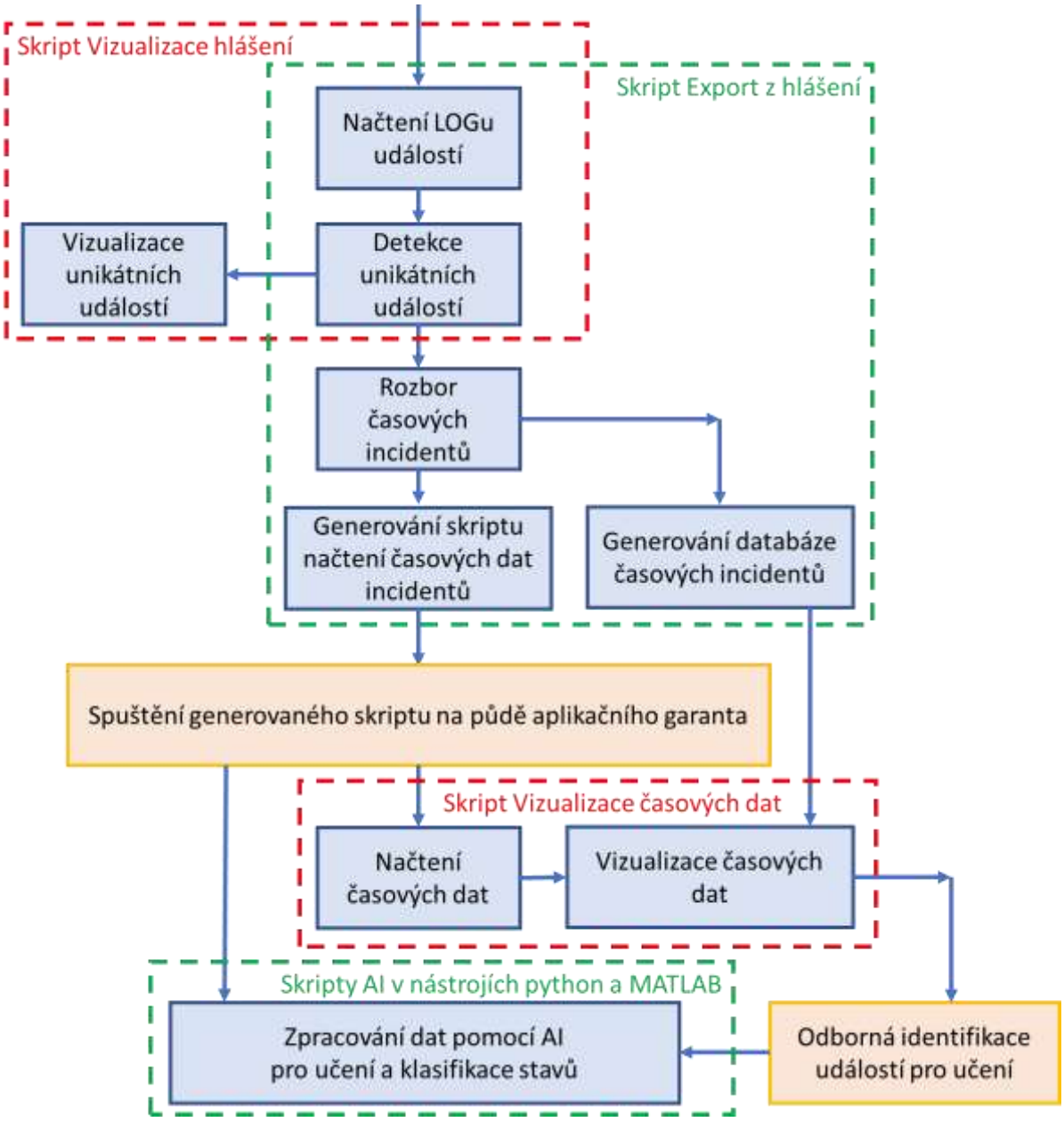
Základní myšlenkou finálního automatického hodnocení průběžných nebo kumulativních časových dat bylo využití nových postupů, které nabízejí technologie umělé inteligence AI (Artificial Intelligence), specificky metody učení a následného rozpoznávání ML (Machine Learning), resp. specificky DL (Deep Learning). Pro aplikaci těchto metod je nezbytné mít dostatek věrohodných dat pro učení. Získané časové údaje bylo tedy nezbytné objektivně odborně zhodnotit, než mohly být použity pro učící se mechanismy. Pro zpracování velkého množství případů byla vyvinuta specifická aplikace včetně grafického uživatelského rozhraní zpracovávající formát vstupních dat a vizualizující časové průběhy pro snadnou identifikaci událostí.

Poslední část spočívá v učení a následném samostatném identifikování událostí. Protože tyto technologie jsou zatím částečně na úrovni experimentální a naše zkušenosti takový stav potvrdily, byl další postup vyvíjen relativně nezávisle na dvou platformách, přičemž jednou bylo komplexní prostředí nástroje MATLAB a druhou, dominantní s ohledem budoucí integraci s podpůrnými aplikacemi, platforma programovacího jazyka python s jeho knihovnami pro ML/DL.

Další posílení učícího procesu a průběžné zkvalitňování klasifikace událostí bylo zamýšleno aplikací simulovaných dat na dispečerském systému aplikačního garanta. Taková data však byla zatím získána jen v omezením množství a variabilitě typu událostí. Do budoucna je tento směr velkou možností zkvalitnění identifikace pomocí AI.

3 Architektura zpracování, vizualizace a detekce provozních stavů

Tato kapitola vysvětluje jednotlivé bloky procesu sběru dat z prostředí ČEPS, identifikace mimořádných provozních událostí, jejich vizualizace a export. Exportovaná data jsou pak využita pro trénink neuronové sítě klasifikační úlohy. Celý proces je graficky znázorněn blokovým diagramem na Obr. 3.1.



Obr. 3.1 Architektura řešené klasifikační úlohy z pohledu načtení, zpracování, klasifikace a interpretace dat

Následující podkapitoly postupují po jednotlivých blocích architektury celého procesu a popisují každý krok zvlášť, tedy:

1) Načtení plus analýza zvoleného systémového výpisu hlášení (LOGu) s cílem vybrat vhodné časové úseky a vytvořit generovaný skript pro načtení těchto úseků alias incidentů z databáze archivovaných dat. Je tedy vytvořen soubor skriptu s implicitním názvem export_dat_WAMS_do_CSV.py pomocí analytického skriptu „Export z hlášení“ (soubor AnalyzaHlaseniExport.py). Současně je stejným skriptem vytvořena i databáze těchto incidentů pro pozdější využití (do souboru typu CSV pod implicitním názvem export_dat_WAMS_do_CSV.csv).
2) Souběžně s činností analýzy hlášení může být pro subjektivní zhodnocení prováděna i vizualizace vyhodnocení unikátnosti pomocí skriptu „Vizualizace hlášení“ (soubor AnalyzaHlaseniVizualizace.py).
3) Spuštění skriptu generovaného v prvním kroku v prostředí s dostupnou databází naměřených dat na půdě aplikačního garanta a získání požadovaných časosběrných dat.
4) Vizualizace vybraných dat pro odborný rozbor a přípravu podkladů učícího procesu nebo identifikace pomocí AI v případě nalezení kvalitních podkladů. Toto provádí skript „Vizualizace časových dat“ (soubor ZobrazeniDat.py).
5) Využití dat klasifikačního algoritmu s využitím neuronových sítí (nezávislé skripty v nástrojích python a MATLAB).
6) Prezentace výsledků klasifikace s využitím neuronových sítí

4 Popis skriptů pro získání a předzpracování WAMS dat

Tato kapitola představuje **kroky 1 až 3** uvedené v architektuře zpracování (kapitola 2.1), vizualizace a detekce provozních stavů.

4.1 Koncept skriptů

Jedná se o sestavu skriptů, které mají umožnit přehledné analyzování dat získaných z řídícího systému WAMS systému ve třech postupných krocích.

1) Načtení plus analýza zvoleného systémového výpisu hlášení (LOGu) s cílem vybrat vhodné časové úseky a vytvořit generovaný skript pro načtení těchto úseků z databáze archivovaných dat.
2) Spuštění skriptu generovaného v prvním kroku v prostředí s dostupnou databází naměřených dat na půdě aplikačního garanta a získání požadovaných časosběrných dat.
3) Vizualizace vybraných dat pro odborný rozbor a přípravu podkladů učícího procesu nebo identifikace pomocí AI v případě nalezení kvalitních podkladů.

Skripty jsou sestaveny v programovacím jazyce python a využívají jeho standardní osvědčené rozšiřující moduly. V dodaném datovém balíčku je pro spuštění optimalizovaná aktuální verse (prosinec 2024) těchto komponent. Alternativně je možné po nainstalování všech potřebných součástí začlenit skripty do vlastní širší funkční sestavy postavené na instalaci jazyka python. Níže je v příloze uvedena potřebná sestava modulů a doporučení pro instalaci.

4.2 Analýza systémového výpisu hlášení (LOGu)

Výpis událostí je poměrně složitý tabulkový soubor, kde z mnoha sloupců (44) jsou důležité zejména časové údaje, hlášení a tzv. entita. Originální struktura je v příloze níže.

Skript opatřený komentáři postupuje při zpracování v následujících krocích:

1) Nastavení důležitých systémových proměnných určujících parametrizaci další činnosti skriptu.
2) Načtení vlastního LOGu do proměnné modulu pandas s převedením textu do formátu UTF8 a vytvoření nových sloupců pro efektivní zpracování časových údajů.
3) Vyhodnocení četnosti hlášení systému, výpis unikátních hlášení s jejich individuálním počtem.
4) Sestavení knihoven rozvoden, vedení a WAMS PMU jednotek.
5) Generování sekvence samostatných incidentů přesahujících zvolený práh unikátnosti hlášení s výpisem počtu zjištěných incidentů.
6) Generování sekvence časově souvislých událostí pro export do skriptu požadujícího časové údaje a také do databáze hlášení a entit (jako soubory s příponami `py` a `csv`). Databáze může být později využita skriptem vizualizace. V případě nemožnosti identifikovat odpovídající PMU je to zapsáno do výstupu. Na případné nepřijatelné a vyřazené požadavky je upozorněno hlášením.

Pro celý proces je kruciálním krokem nalezení vhodných časových úseků zúročitelných pro učení a následné rozbory algoritmy AI. Tyto úseky je velice těžké vyčlenit v obrovském množství málo vypovídajících naměřených dat. Jakmile bylo zřejmé, že tuto část nelze zajistit

intuitivní cestou za pomoci obsluhujícího personálu v dostatečném množství, bylo zahájeno úsilí vyvinout příslušný podpůrný software. Základní myšlenkou je klasifikovat hlášení systému do úrovní významové výjimečnosti, tedy zavedení pojmu indexu unikátnosti hlášení v rámci sledovaného časového období. Na základě tohoto faktoru je pak vybrána přiměřeně úzká skupina dostatečně ojedinělých hlášení dle zvolené uživatelské úrovně.

Následný výběr sledovaných období je podroben postupu výše tak, aby se načítaly jen relevantní data incidujících uzlů a větví topologie soustavy a současně byly věrohodně určeny dle kontextu příslušné měřící jednotky. Automaticky jsou také sloučeny vícenásobné podněty, které se ve skutečnosti týkají stejné kauzy, nebo vyplývají z anomálií potencionálně příčinně souvisejících.

Skript je doplněn dalším rozšiřujícím, který zobrazuje jednotlivá vstupní hlášení s ohledem na jejich unikátnost v grafu jako interaktivní www stránku:

Obr. 4.1 Zobrazení unikátnosti hlášení

Funkčnost tohoto skriptu vyžaduje dostupnost kompatibilního webového prohlížeče.

4.3 Vizualizace časosběrných dat WAMS

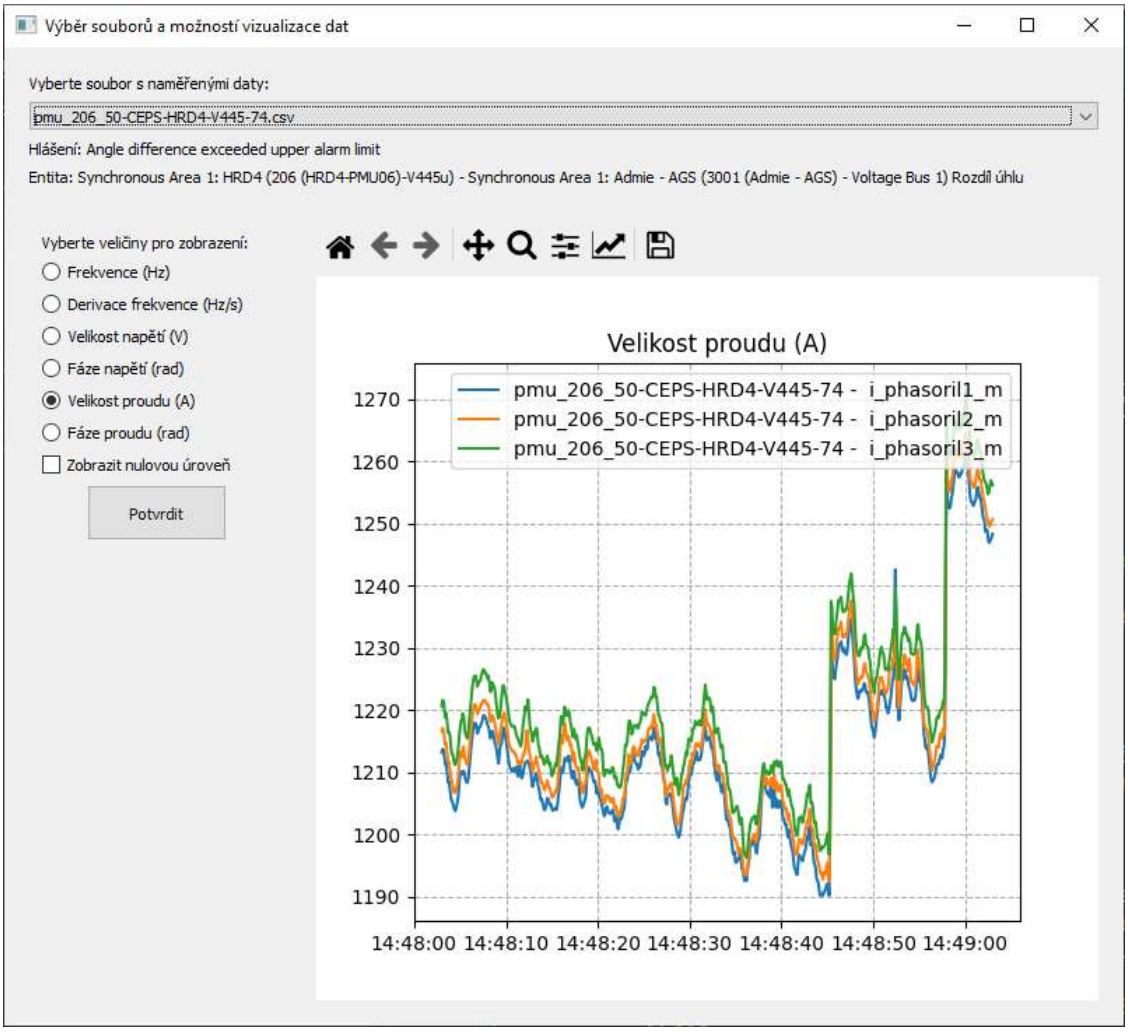
Data získaná činností skriptu generovaného v prvním kroku je pro účely rozboru vhodné zobrazit s možností zobrazení jednotlivých složek a rychlého přechodu mezi výstupy z různých PMU a časových úseků.

Výstupní zobrazitelné veličiny jsou:

- Frekvence (Hz)
- Derivace frekvence (Hz/s)
- Velikost napětí ve třech fázích (V)
- Fáze napětí ve třech fázích (rad)
- Velikost proudu ve třech fázích (A)
- Fáze proudu ve třech fázích (rad)

Alternativně je možno pro snadnější zhodnocení významnosti změn veličin zobrazit souběžně i fiktivní nulovou hodnotu tak, aby se malé změny velké absolutní hodnoty nejevily jako podstatné.

Příklad zobrazení velikosti proudů:



Obr. 4.2 Zobrazení absolutní velikosti proudů

Před spuštěním tohoto skriptu je kromě souborů s daty nutno mít ve stejném adresáři i soubor databáze hlášení a entit (standardně export_dat_WAMS_do_CSV.csv) získaný v prvním kroku.

5 Struktura klasifikačního algoritmu

Vývoj klasifikační úlohy běžel primárně ve vývojovém prostředí python, kterému je věnován následující kapitola. Paralelní vývoj klasifikační úlohy běžel v prostředí MATLAB a toolboxu Deep network designer, který umožňoval shodnou architekturu neuronových sítí, jednoduchou editaci vlastností celé úlohy a grafickou interpretaci. Hlavní výhodou prostředí MATLAB je práce s daty ve workspace, kdy je možné jednoduše kontrolovat mezivýpočty, jak graficky, tak hodnoty v jednotlivých proměnných. Princip fungování (a do jisté míry i kód) je naprosto shodný jak v prostředí MATLAB, tak python a přináší i stejné výsledky. **Pro finální software a aplikaci na straně aplikačního garanta byla klasifikační úloha implementována v pythonu, zatímco** MATLAB **sloužil výhradně pro verifikační účely.**

5.1 Vývoj v prostředí MATLAB

Kód klasifikace je rozdělen na 4 části, které jsou v MATLABu rozděleny do 4 m-file souborů. Každý m-file je pak volán jednotlivě prostřednictvím funkcí. Soubory jsou následující:

- main.m – volání všech funkcí
- data_prep.m – příprava dat pro aplikaci v neuronové síti
- train.m – vytvoření neuronové sítě a trénink prostřednictvím výstupů z funkce data_prep
- klasifikace.m – vizuální kontrola výstupních dat klasifikace

5.1.1 Funkce data_prep

1) **Načtení dat**
   - Skript původně obsahoval sekci pro načtení dat z jednotlivých CSV souborů reprezentujících různé provozní stavy (např. Z1F, BSP, SZZ), která byla nahrazena načtením předpřipraveného souboru loaded_csv.mat. Tento přístup zrychluje zpracování dat díky eliminaci potřeby opakovaného čtení z disku.
2) **Definice parametrů neuronové sítě**
   - Skript definuje počet tříd (4 třídy: Z1F, BSP, SZZ, CLR) a délku časového okna, která určuje velikost segmentů dat využívaných při tréninku.
3) **Normalizace dat**
   - Data pro každý provozní stav (Z1F, BSP, SZZ) jsou normalizována na škále <0, 1>.
     - **Min-max normalizace**: Každý sloupec dat je upraven tak, že minimální hodnota se stává 0 a maximální hodnota 1.
     - Alternativní možnost, která je zakomentována, nabízí normalizaci na základě průměru a směrodatné odchylky.

4) **Vytvoření tensorů**
   - Pro každý provozní stav jsou data rozčleněna na časově posunuté segmenty (shifting windows). Tyto segmenty jsou uloženy jako jednotlivé buňky tensorů.
5) **Výběr specifických segmentů pro trénink**
   - Vybrané časové úseky ze Z1F, BSP a SZZ jsou označeny jako reprezentativní pro dané provozní stavy.
   - Typ CLR (běžný stav) je vytvořen kombinací dat z normálních segmentů Z1F, BSP a SZZ.
6) **Randomizace dat**
   - Data z každé třídy jsou náhodně promíchána (permutace pořadí segmentů), aby se minimalizovalo riziko systematických chyb během tréninku.
7) **Rozdělení dat na trénovací, validační a testovací sady**
   - Data z každé třídy jsou rozdělena v poměru 70 % pro trénink, 20 % pro validaci a 10 % pro testování. Toto rozdělení zajišťuje vyváženost mezi tréninkem modelu, jeho laděním a testováním.
8) **Klasifikace**
   - Každý segment dat je označen příslušnou třídou v binární reprezentaci (např. [1 0 0 0] pro CLR). Tato označení jsou použita k propojení tensorů dat a jejich klasifikací.
9) **Konečná sestava trénovacích, validačních a testovacích dat**
   - Data pro všechny třídy jsou spojena do jednotných tensorů pro každou sadu (trénovací, validační a testovací) spolu s odpovídajícími tensorovými klasifikacemi. To usnadňuje jejich další využití při tréninku neuronové sítě.

Tento postup zajišťuje kvalitní přípravu dat pro strojové učení, minimalizuje chyby z nevhodně zpracovaných dat a umožňuje robustní trénink neuronové sítě.

5.1.2 Funkce train

1) **Načtení dat**
   - Skript přijímá trénovací, validační a testovací data ve formě tensorů (tensor_all_train, tensor_all_val, tensor_all_test) a odpovídajících tříd (tensor_class_all_train, tensor_class_all_val, tensor_class_all_test).
   - Tensorová data mají rozměry:
     - Trénovací: [1x708], každá buňka obsahuje matici [121x24].
     - Validační: [1x203].
     - Testovací: [1x104].
   - Převod tříd do kategorií:
     - Každý záznam ve třídách je převeden na kategorický formát (např. pro klasifikaci do čtyř tříd: CLR, Z1F, SZZ, BSP).
     - Hodnoty tříd jsou extrahovány jako indexy maximálních hodnot v matici tříd a převedeny na kategorické buňky.
2) **Definice architektury neuronové sítě**
   - Síť se skládá z následujících vrstev:
     - **Vstupní vrstva**: sequenceInputLayer přijímá data s rozměry [121x24] (počet časových vzorků a parametrů na vstupu).
     - **Konvoluční vrstvy**:
       - Celkem čtyři vrstvy s různými konfiguracemi:
         - Jádra o velikosti 7 a 3.

        - Počty filtrů: 32, 8, 32, 16.
        - Padding je nastaven na "same" pro zachování rozměrů po konvoluci.
      - Mezi konvoluční vrstvy jsou vloženy aktivační vrstvy reluLayer pro nelinearitu.
    - **Zplošťovací vrstva**: flattenLayer převádí výstup konvolučních vrstev na jednorozměrný vektor.
    - **Plně propojené vrstvy**:
      - První má 256 neuronů s aktivací ReLU.
      - Druhá má 4 neurony odpovídající čtyřem třídám (CLR, Z1F, SZZ, BSP).
    - **Softmax vrstva**: Převádí výstupy na pravděpodobnosti tříd.
    - **Klasifikační vrstva**: Určuje finální předpověď.
3) **Nastavení trénovacího procesu**
    - **Optimalizační algoritmus**: Adam s počáteční rychlostí učení 0.001.
    - **Počet epoch**: 30 (plných průchodů trénovacím datasetem).
    - **Velikost dávky**: 32 vzorků na dávku.
    - **Validační data**: Používají se ke sledování výkonu během tréninku.
    - **Frekvence validace**: Každých 50 iterací.
    - **Vizualizace pokroku**: Grafický výstup průběhu tréninku.
4) **Trénování neuronové sítě**
    - Funkce trainNetwork trénuje síť s použitím trénovacích dat a nastavení:
      - Trénovací data (X_train, Y_train) jsou použita k optimalizaci váh sítě.
      - Validační data (X_val, Y_val) slouží k monitorování výkonnosti modelu během tréninku.
    - Výstupem je natrénovaná síť net, která je připravena k použití na testovacích datech.

Tento proces kombinuje moderní architekturu neuronové sítě s efektivním optimalizačním algoritmem a pečlivým řízením trénovacího procesu, což zajišťuje kvalitní výstup pro klasifikaci provozních stavů.

5.1.3 Funkce klasifikace

1) **Klasifikace na testovacích a dalších datech**
    - Funkce classify se používá k aplikaci natrénované neuronové sítě (net) na různé datové sady:
      - X_test: Testovací data, která slouží k vyhodnocení celkového výkonu sítě.
      - tensor_csv_data_Z1F: Data odpovídající třídě Z1F (jednofázový zkrat).
      - tensor_csv_data_CLR: Data běžného provozního stavu (CLR).
      - tensor_csv_data_SZZ: Data skokové změny zatížení (SZZ).
      - tensor_csv_data_BSP: Data blízkého spínacího prvku (BSP).
    - Výstupem klasifikace je pravděpodobnostní přiřazení každého vzorku ke konkrétní třídě (CLR, Z1F, SZZ, BSP).
2) **Převod predikovaných hodnot**
    - Predikované hodnoty (YPred) i skutečné třídy (Y_test) jsou převedeny na numerické hodnoty pomocí funkce cellfun. To umožňuje jednodušší manipulaci a vizualizaci:

        - YPred_numeric_test: Predikované hodnoty pro testovací data.
        - Ytest_numeric: Skutečné třídy testovacích dat.
        - Podobný převod je proveden i pro predikované hodnoty z datasetů Z1F, CLR, SZZ a BSP.

3) **Vizualizace výsledků**
    - **Grafy**: Pro každou datovou sadu je vytvořen samostatný graf v matici 5×1 (pět podgrafů). Každý graf obsahuje predikované hodnoty vizualizované jako značky "x":
        - **Graf 1: CLR** – Zobrazuje predikce běžného provozního stavu.
        - **Graf 2: Z1F** – Predikce jednofázových zkratů.
        - **Graf 3: SZZ** – Predikce skokových změn zatížení.
        - **Graf 4: BSP** – Predikce blízkých spínacích prvků.
        - **Graf 5: Testovací data** – Porovnává predikované (YPred_numeric_test) a skutečné hodnoty (Ytest_numeric).
    - Popis grafů:
        - **Popisky os y**: Označují třídy (0 - CLR, 1 - Z1F, 2 - SZZ, 3 - BSP).
        - **Rozsah osy y**: Nastaven na hodnoty -0.5 až 5 pro přehlednost.
        - **Popisky a názvy grafů**: Umožňují snadnou interpretaci výsledků.
        - **Mřížka**: Přidána pro lepší čitelnost.
4) **Porovnání testovací sady**
    - Poslední graf (subplot(5,1,5)) zobrazuje společně predikované (YPred_numeric_test) a skutečné třídy (Ytest_numeric).
    - Použité značky:
        - "x" pro predikované hodnoty.
        - "o" pro skutečné hodnoty.
    - Toto zobrazení usnadňuje hodnocení přesnosti klasifikace.

Skript umožňuje klasifikaci různých datových sad pomocí neuronové sítě a poskytuje grafickou interpretaci výsledků. Vizualizace predikcí a jejich porovnání se skutečnými hodnotami umožňuje rychlou analýzu výkonu modelu a identifikaci případných chyb.

5.1.4 Hlavní soubor main.m

Hlavní soubor main.m slouží k řízení celého procesu přípravy dat, trénování neuronové sítě a klasifikace. Je rozdělen do tří hlavních sekcí:

1) **Příprava dat (Data preparation function)**
    - **Velikost časového okna:** Proměnná window_size definuje délku časového okna (v tomto případě 120), která je klíčová pro segmentaci vstupních dat.
    - Funkce data_prep_241203 provádí kompletní zpracování vstupních dat:
        - Načtení a normalizace dat pro jednotlivé třídy (Z1F, BSP, SZZ, CLR).
        - Rozdělení dat na trénovací, validační a testovací sady.
        - Generování tensorů, které jsou připravené pro vstup do neuronové sítě.
    - Výstupy funkce:
        - Tensorová data (tensor_all_train, tensor_all_val, tensor_all_test) a odpovídající klasifikace (tensor_class_all_train, tensor_class_all_val, tensor_class_all_test).
        - Datové sady pro jednotlivé třídy: tensor_csv_data_Z1F, tensor_csv_data_BSP, tensor_csv_data_SZZ, tensor_csv_data_CLR.

2) **Trénování neuronové sítě (Training)**
   - Funkce train provádí:
     - Optimalizaci neuronové sítě na základě trénovacích dat (tensor_all_train, tensor_class_all_train).
     - Sledování výkonnosti pomocí validační sady (tensor_all_val, tensor_class_all_val).
   - Výstupy funkce:
     - Natrénovaná neuronová síť net.
     - Testovací data:
       - X_test: Tensorová data testovací sady.
       - Y_test: Odpovídající skutečné klasifikace.

3) **Klasifikace (Classification)**
   - Funkce klasifikace:
     - Aplikuje natrénovanou neuronovou síť net na testovací data (X_test) a na data jednotlivých tříd (Z1F, CLR, SZZ, BSP).
     - Vytváří predikce pro jednotlivé třídy i celkovou testovací sadu.
   - Výsledky klasifikace jsou zobrazeny v přehledných grafech:
     - Zvlášť pro každou třídu (CLR, Z1F, SZZ, BSP).
     - Porovnání predikovaných hodnot a skutečných tříd na testovací sadě.

Soubor main.m spojuje klíčové části zpracování dat, tréninku a vyhodnocení neuronové sítě. Zajišťuje:

1. Přípravu dat pro strojové učení.
2. Trénování neuronové sítě na reálných datech z provozu.
3. Vyhodnocení výkonu sítě pomocí klasifikace a vizualizace výsledků.

Tento soubor je vstupním bodem celého procesu a jeho jednoduchá struktura umožňuje snadnou úpravu parametrů či funkcionalit. **Detailní kód je dostupný v přílohách tohoto dokumentu.**

5.1.5 Vytvoření neuronové sítě v nástroji Deep network designer

Architektura sítě je sekvenční a byla optimalizována pro analýzu časových dat. Hlavní charakteristiky jednotlivých vrstev jsou následující:

Architektura sítě

1. **Vstupní vrstva**:
   - sequenceInputLayer je konfigurována pro vstupy o velikosti [121, 24], což odpovídá počtu časových vzorků (121) a počtu parametrů měření (24) v jednom časovém okně.
2. **Konvoluční vrstvy**:
   - Síť obsahuje čtyři konvoluční vrstvy s různými konfiguracemi:
     - První vrstva: jádro velikosti 7, 32 filtrů.
     - Druhá vrstva: jádro velikosti 7, 8 filtrů.
     - Třetí vrstva: jádro velikosti 7, 32 filtrů.
     - Čtvrtá vrstva: jádro velikosti 3, 16 filtrů.
   - Každá konvoluční vrstva využívá padding typu "same", což zachovává rozměry vstupních dat.

3. **Aktivační vrstvy**:
    - Po každé konvoluční vrstvě je zařazena aktivační vrstva reluLayer, která přidává nelinearitu a zlepšuje schopnost sítě modelovat složité vzory.
4. **Zplošťovací vrstva**:
    - flattenLayer převádí výstupy konvolučních vrstev na jednorozměrný vektor pro vstup do plně propojených vrstev.
5. **Plně propojené vrstvy**:
    - První plně propojená vrstva obsahuje 256 neuronů s aktivací ReLU.
    - Druhá plně propojená vrstva má čtyři neurony, které odpovídají čtyřem výstupním třídám (CLR, Z1F, SZZ, BSP).
6. **Softmax a klasifikační vrstva**:
    - softmaxLayer převádí výstupy na pravděpodobnostní hodnoty.
    - classificationLayer určuje finální přiřazení vstupu k jedné ze čtyř tříd.

Nastavení trénovacího procesu

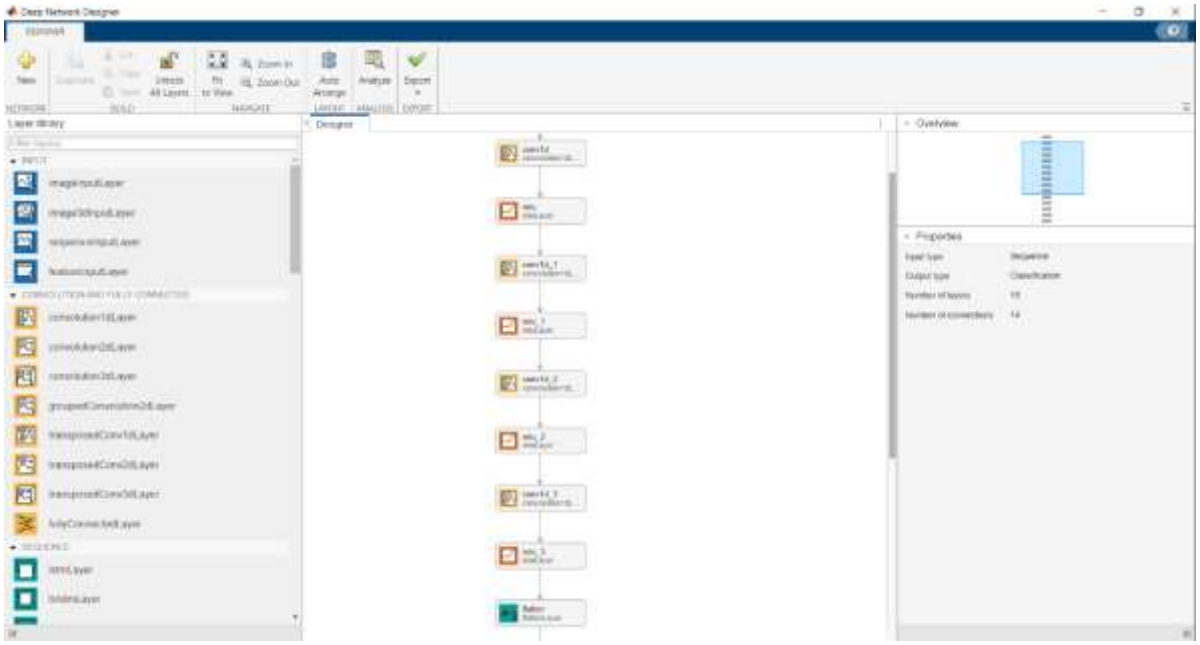
Trénink neuronové sítě byl proveden s následujícím nastavením:

- **Optimalizační algoritmus**: Adam, který je efektivní pro učení neuronových sítí a minimalizuje chyby s použitím adaptivních rychlostí učení.
- **Počet epoch**: 30 (plných průchodů trénovacím datasetem).
- **Velikost dávky (mini-batch)**: 32 vzorků na iteraci, což vyvažuje rychlost a přesnost tréninku.
- **Počáteční rychlost učení**: 0.001, což zajišťuje stabilní konvergenci.
- **Validační data**: Použita pro sledování výkonu modelu během tréninku s frekvencí validace každých 50 iterací.
- **Vizualizace pokroku**: Trénink byl doplněn o grafický výstup průběhu učení, který umožnil sledování zlepšení přesnosti a poklesu chyby v reálném čase.

Neuronová síť byla navržena a testována v prostředí MATLAB s využitím nástroje **Deep Network Designer**. Tento nástroj nabízí několik zásadních výhod:

- **Grafická interpretace**: Jasné zobrazení struktury sítě a jednotlivých vrstev.
- **Možnost ukládání a načítání modelů**: Síť lze snadno uložit do workspace a znovu načíst pro další experimenty nebo nasazení.
- **Intuitivní ladění**: Umožňuje snadnou modifikaci parametrů jednotlivých vrstev bez nutnosti zásahů do kódu.

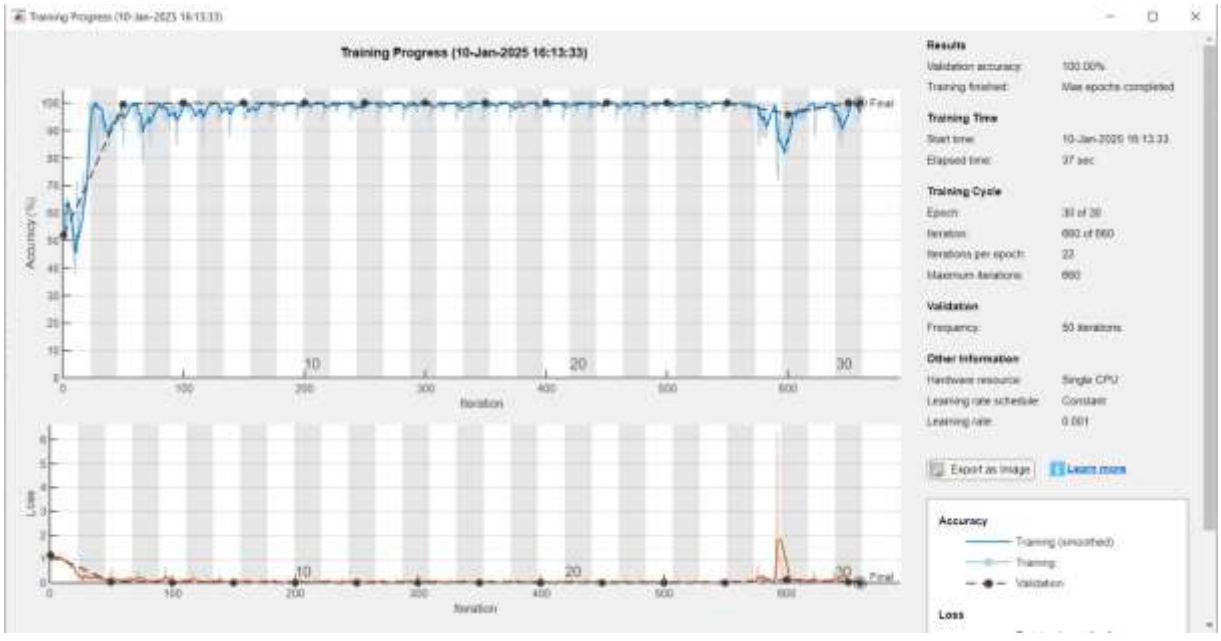
Následující obrázek je snímkem obrazovky z toolboxu Deep Network Designer v prostředí MATLAB.



Obr. 5.1 Deep network designer toolbox v prostředí MATLAB, konfigurace neuronové sítě

5.1.6 Vytvoření neuronové sítě v nástroji Deep network designer

Vývoj úlohy v prostředí MATLAB umožňuje uživatelům vizualizovat průběh tréninku neuronové sítě prostřednictvím okna **Training Progress**, které se automaticky zobrazí po spuštění tréninkového procesu. Toto okno poskytuje cenné informace o vývoji přesnosti a chyby modelu v čase, což umožňuje průběžné sledování výkonu sítě (viz následující Obr. 5.2).



Obr. 5.2 Ukázka trénovací sekvence neuronové sítě

Graf přesnosti (Accuracy) zobrazuje křivky pro přesnost klasifikace na trénovacích a validačních datech.

- Křivky:
  - **Training (smoothed)**: Vyhlazená křivka přesnosti na trénovacích datech, která eliminuje krátkodobé výkyvy a ukazuje celkový trend.
  - **Training**: Surová křivka přesnosti na trénovacích datech po jednotlivých iteracích.
  - **Validation**: Přesnost modelu na validačních datech. Tato křivka ukazuje, jak dobře síť generalizuje mimo tréninkovou sadu.
- Interpretace:
  - Rostoucí přesnost na validačních datech je indikátorem zlepšujícího se výkonu sítě.
  - Rozdíl mezi křivkami tréninkové a validační přesnosti může indikovat přetrénování, pokud je validační přesnost výrazně nižší.

Graf ztráty (Loss) sleduje hodnotu ztrátové funkce, která slouží jako měřítko chyby modelu.

- Křivky:
  - **Training (smoothed)**: Vyhlazená křivka ztráty na trénovacích datech.
  - **Training**: Hodnota ztráty po jednotlivých iteracích na trénovacích datech.
  - **Validation**: Hodnota ztráty na validačních datech.
- Interpretace:
  - Snižující se ztráta na validačních datech značí zlepšování modelu.
  - Stagnace nebo nárůst ztráty na validačních datech při současném poklesu na trénovacích datech může být známkou přetrénování.

Další informace v okně Training Progress jsou:

- **Validation Accuracy**: Nejlepší dosažená přesnost na validačních datech.
- **Training Finished**: Informace o ukončení tréninku, včetně důvodu (např. dosažení maximálního počtu epoch).
- **Training Time**: Celková doba tréninku.
- **Training Cycle**: Informace o průběhu epoch a iterací:
  - **Epoch**: Počet kompletních průchodů trénovacím datasetem.
  - **Iteration**: Počet provedených iterací během tréninku.
  - **Iterations per Epoch**: Počet iterací potřebných k dokončení jedné epochy.
  - **Maximum Iterations**: Celkový počet iterací při zadaném počtu epoch.
- **Validation Frequency**: Frekvence validace (např. každých 50 iterací).
- **Informace o CPU**: Přehled o využití procesoru během tréninku.

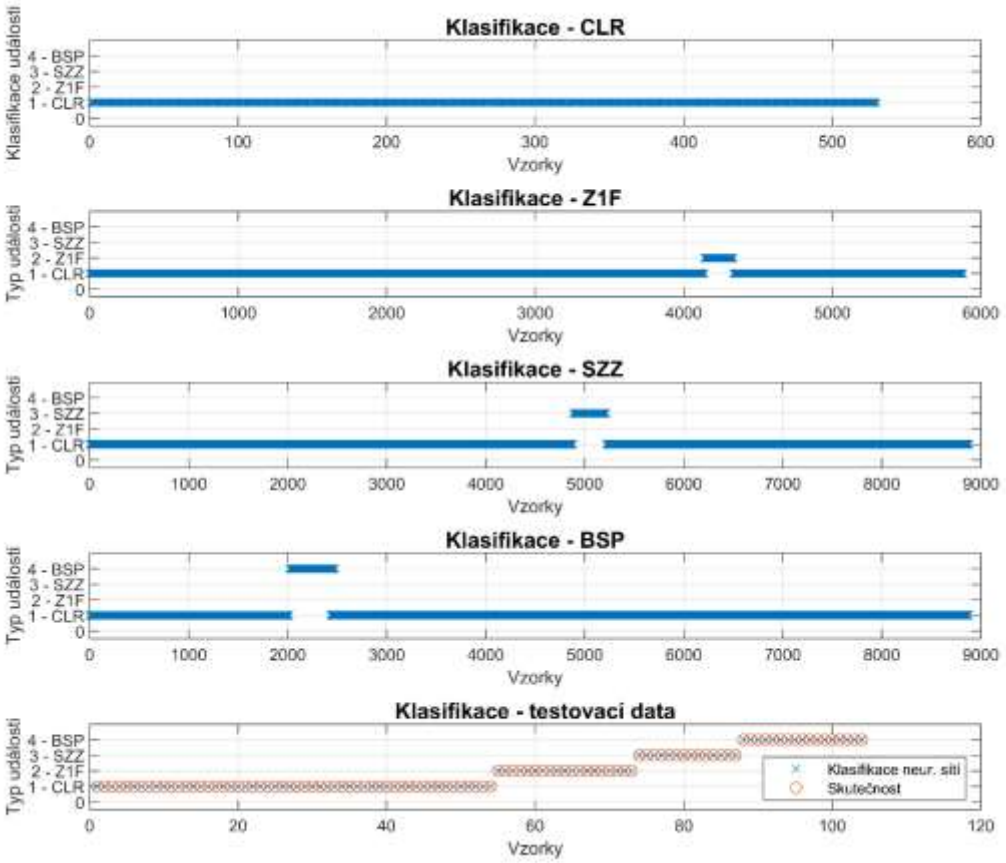
Tato vizualizace poskytuje snadno interpretovatelné zpětné vazby o výkonu modelu a je důležitým nástrojem pro ladění hyperparametrů a identifikaci problémů, jako je přetrénování nebo nedostatečný výkon modelu.

5.1.7 Interpretace výsledků

Následující obrázek ukazuje funkčnost vytrénované neuronové sítě při klasifikaci provozního stavu na měřených datech. Poslední graf na Obr. 5.3 (Klasifikace – testovací data) ukazuje, zda na balíku testovacích dat byl při tréninku neuronová síť úspěšná ve všech klasifikacích. V této konfiguraci parametrů a vstupních dat byl síť 100% úspěšná – to je základní předpoklad pro správnou detekci v grafech 1–4. Na prvním grafu síť správně klasifikuje stav CLR, kdy na

vstupních datech skutečně nebyla žádná událost. Na grafu druhém dochází ke správné identifikaci stavu Z1F. Stejně tak na grafu třetím a čtvrtém je vidět, že neuronová síť správně klasifikovala stav SZZ a BSP po dobu trvání tohoto stavu.

Je nutné zmínit, že naladit sadu hyperparametrů a správnou velikost okna včetně správně randomizovaných dat je časově náročná úloha. S rostoucím balíkem vstupních dat a větším počtem klasifikačních stavů bude nutné vyvinout optimalizační úlohu pro volbu parametrů a velikosti časového okna.



Obr. 5.3 Interpretace výsledků klasifikační úlohy vyvinuté v prostředí MATLAB

5.2 Vývoj v prostředí python

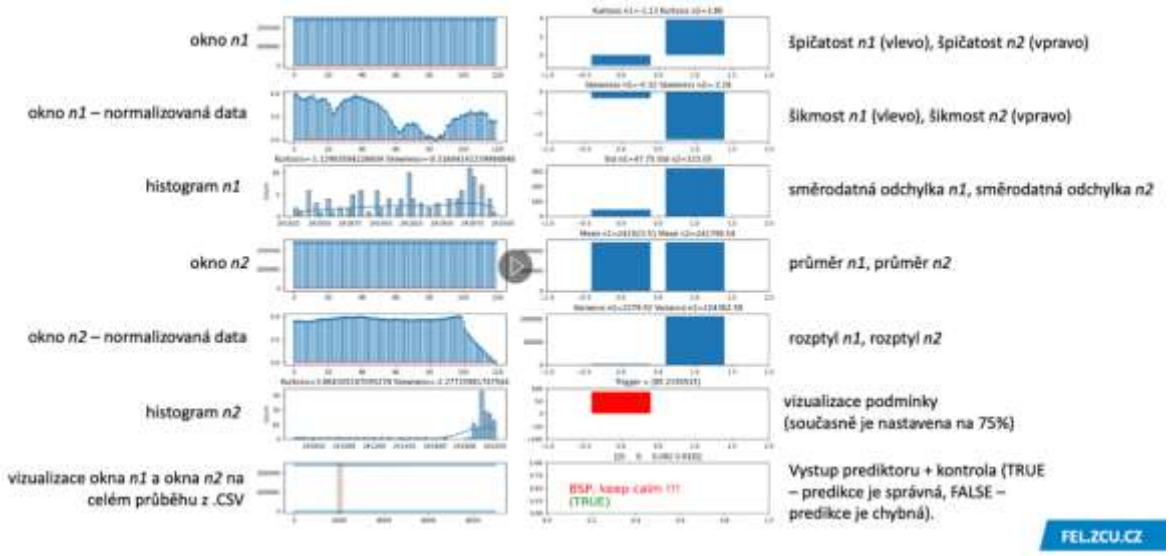
Přístup k řešení problému bude záviset na vlastnostech vstupních dat. Pokud data tvoří bloky s relativně dlouhou časovou posloupností, bude přístup odlišný od situace, kdy se předpokládá krátká časová posloupnost.

5.2.1 Délka okna

Pro správnou činnost neuronové sítě je nutné zvolit vhodnou délku okna, v jehož rámci budou data přiváděna na její vstup. Zároveň je žádoucí, aby vstupní data byla normalizovaná. V případě časové posloupnosti lze použít normalizaci v intervalu ⟨0, 1⟩ na základě maximálních a minimálních hodnot relevantních pro zkoumaný jev.

Pokud však v aktuálním okně chybí časově proměnný signál, může normalizace zesílit šum. V takovém případě je vhodné nejprve detekovat nepřítomnost užitečného (nesoucí informaci o specifické zkoumané události) signálu ještě před normalizací. Alternativně, pokud je vstupní blok dat větší než okno, lze normalizovat celý blok předem a následně posouvat okno po datech. Tato metoda však není účinná, pokud je datový blok srovnatelný s velikostí okna, například při práci s daty v reálném čase. Situaci navíc komplikuje fakt, že při běžném provozu systému po většinu času nedochází k výrazným změnám signálu. Pokud tak má být problém řešen jako klasifikační úloha, je nutné zavést další třídu reprezentující normální stav systému k ostatním třídám reprezentujícím specifické zkoumané události.

Jedno z možných řešení je znázorněno na Obr. 5.4.



Obr. 5.4 Princip funkce neuronové sítě s použitím hodnot rozptylu

5.2.2 Identifikace provozních stavů s využitím třídy CLR

Pro použití klasifikační neuronové sítě je nutné předem definovat požadovaný počet tříd. Vzhledem k tomu, že ve většině dat nejsou zaznamenány žádné události, byla vytvořena třída CLR. Neuronová síť tedy klasifikuje vstupní data do čtyř tříd:

1. **Žádná událost – CLR**
2. **Jednofázový zkrat – Z1F**
3. **Blízký spínací prvek – BSP**
4. **Skoková změna zatížení – SZZ**

Jedná se tedy o jednorozměrnou konvoluční neuronovou síť, kde na vstupu jsou umístěny tři konvoluční vrstvy sloužící jako extraktor příznaků. Tyto vrstvy obsahují 8, 16 a 32 filtrů, přičemž velikost konvolučních jader je ve všech případech stejná (3). Pro aktivaci byla použita funkce ReLU.

Následuje vrstva Flatten, která převádí výstup konvolučních vrstev do jednorozměrné podoby. Výstupní klasifikátor obsahuje jednu skrytou vrstvu se 128 neurony a výstupní vrstvu se 4 neurony. Pro klasifikaci je použita aktivační funkce Softmax.

Neuronová síť byla trénována standardním způsobem, avšak data pro trénink byla vybírána podle principu referenčního okna, který je detailně popsán níže. Stejný postup byl použit i při testování účinnosti neuronové sítě, kdy byla vstupní data nejprve předzpracována algoritmem referenčního okna.

Princip referenčního okna:

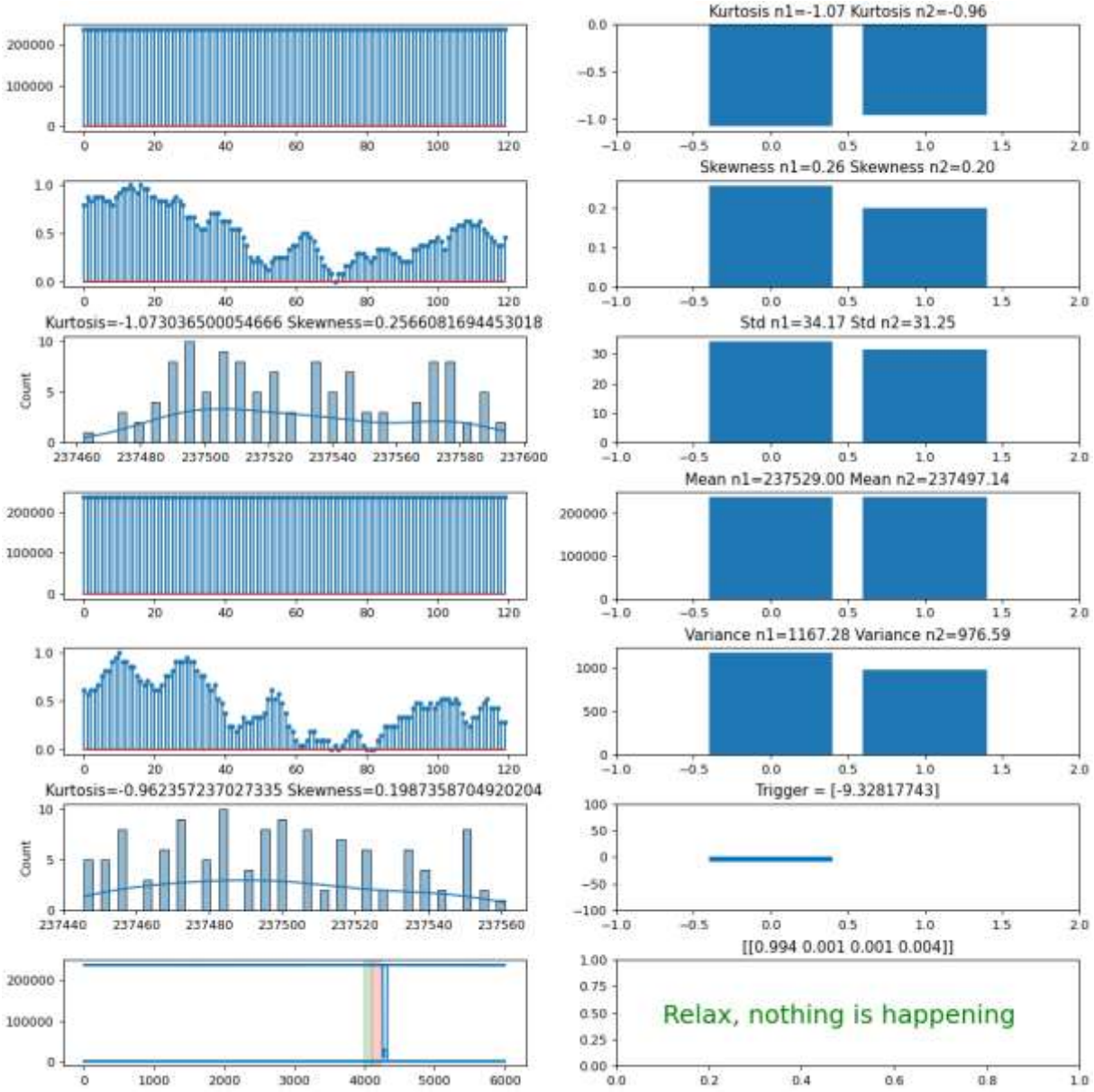
- Jsou použita dvě časová okna, **n1** a **n2**, přičemž každé má velikost **120 vzorků**.
- Okna plavou po časové ose, přičemž okno n1 je za normálních podmínek zpožděno za oknem n2 o 120 vzorků.
- Vypočítává se **rozptyl (variance)** hodnot signálu v oknech **n1 a n2**.
- Následně se určuje, o kolik procent je rozptyl n2 větší, než rozptyl n1 (předpokládá se, že okno n2 zachytí užitečný signál jako první).

Podmínky a následné kroky:

1. **Pokud je rozptyl n2 o více než 75 % větší, než rozptyl n1** (hodnota **75 %** byla stanovena experimentálně), pak:
    - Data v **n2** se normalizují následovně:
        - **Skutečné maximum → 1**
        - **Skutečné minimum → 0**
    - Okno **n2** se uloží do složky odpovídající třídě (**Z1F, BSP nebo SZZ**).
    - Okno n1 se přestává posouvat („zmrazí se“) až do doby, kdy rozptyl n2 opět klesne pod 75 %.
2. **Pokud je rozptyl n2 menší nebo rovný 75 % rozptylu n1, pak**:
    - Data v **n2** se normalizují následovně:
        - **Skutečné maximum → 1**
        - **Hodnota 0 → 0**
    - Okno **n2** se uloží do složky s názvem třídy **"CLR"** (reprezentující normální stav).

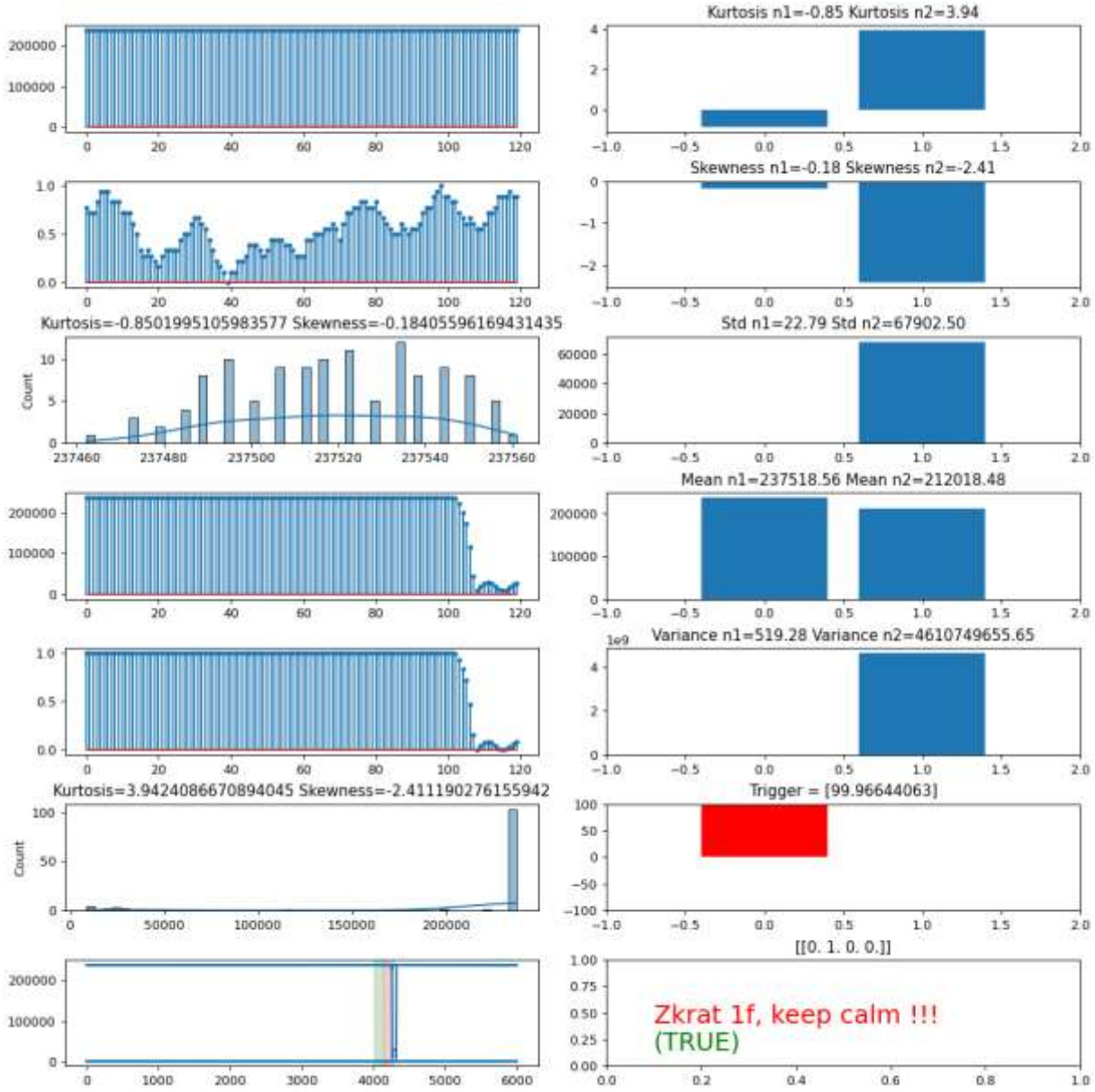
Extrahovaná data se následně využívají k **trénování neuronové sítě**, která na výstupu klasifikuje vzory do již zmíněných **čtyř tříd**.

Z Obr. 5.5 je patrné, že obě okna jsou mimo specifickou událost, neuronová síť tedy klasifikuje příchozí data jako **CLR**.



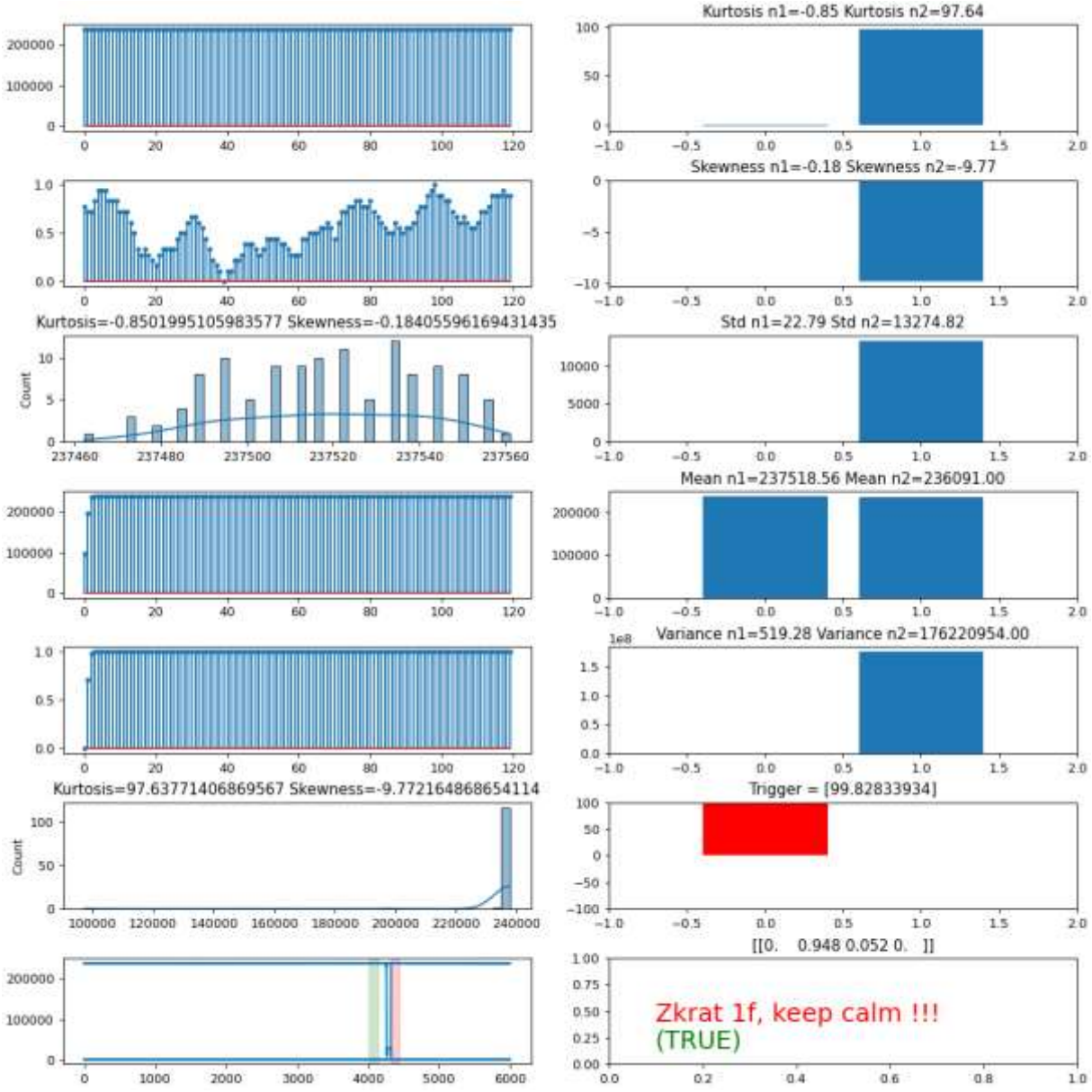
Obr. 5.5 Průběh klasifikační úlohy s událostí mimo časové okno

Z Obr. 5.6 je patrné, že druhé okno n2 (červené) vstupuje do zóny události, což způsobí, že rozptyl v okně n2 začne výrazně převyšovat rozptyl v okně n1. V důsledku toho se aktivuje trigger, kdy dochází k normalizaci dat. Neuronová síť následně klasifikuje tuto událost jako jednofázový zkrat (Z1F).



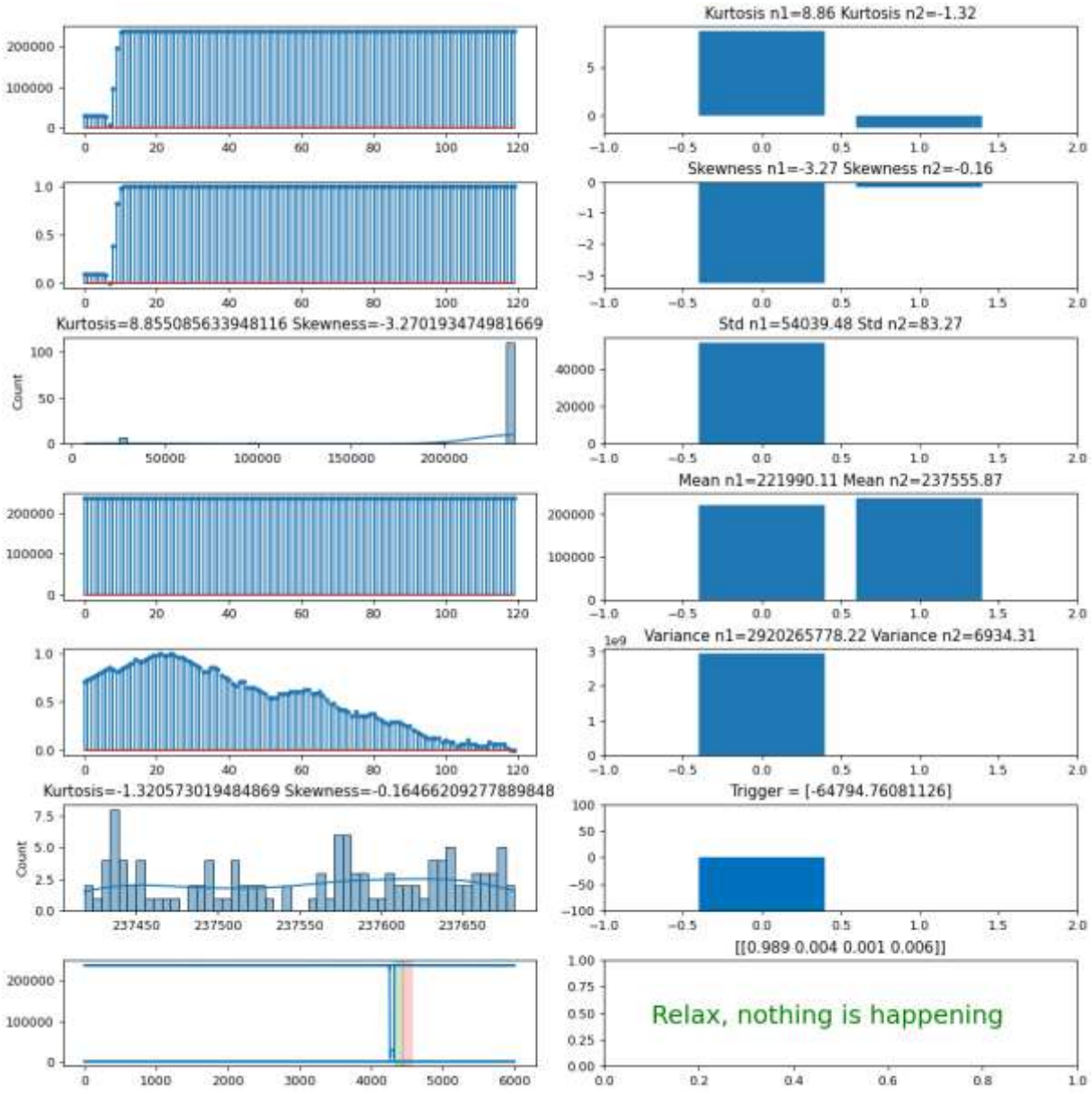
Obr. 5.6 Průběh klasifikační úlohy s událostí uvnitř zóny časového okna

Z Obr. 5.7 je jasně patrné, že s posouváním okna n2, okno n1 zůstává na místě v okamžiku aktivace triggeru.



Obr. 5.7 Průběh klasifikační úlohy s událostí uvnitř časového okna – vlastnosti referenčního okna

Jakmile přestanou být triggery aktivní, okno n1 se vrátí do původní pozice (bez zpoždění). Tento stav je znázorněn na Obr. 5.8.



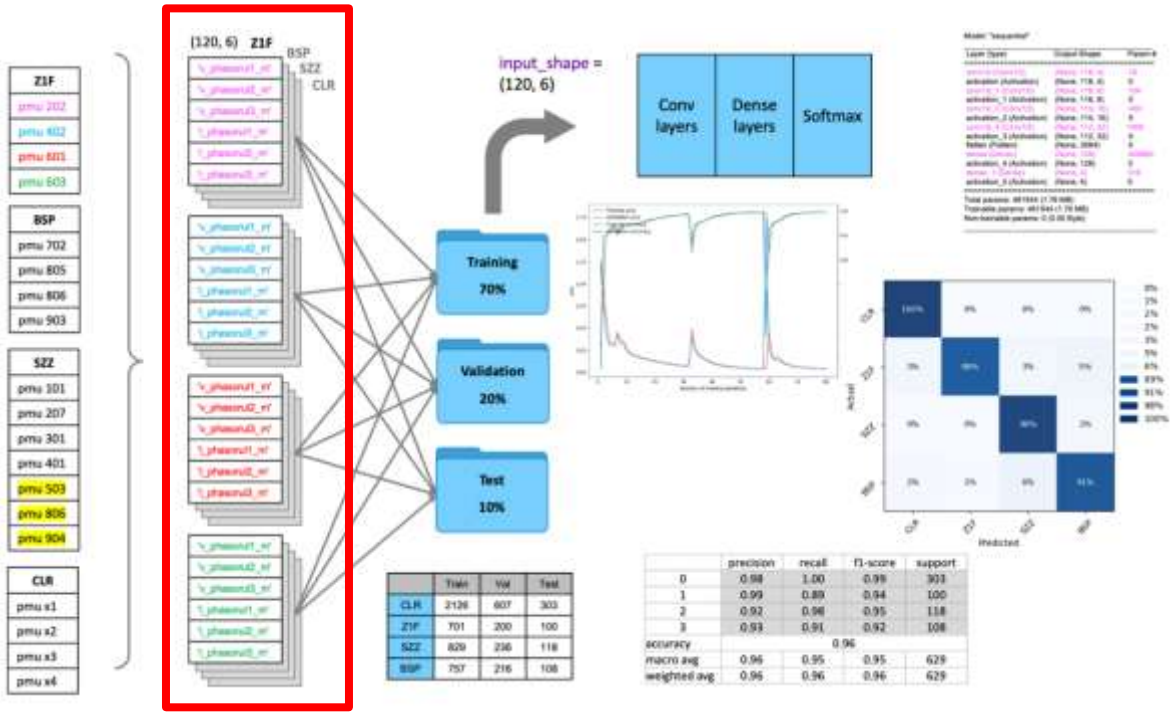
Obr. 5.8 Průběh klasifikační úlohy s událostí uvnitř zóny časového okna – návrat referenčního okna do původní pozice

5.2.3 Porovnání PMU jako nezávislé datové bloky vs. PMU sdružené do 2D tensoru

Pro třídu Z1F (jednofázový zkrat) je predikce 100 % správná v celém sledovaném spektru (pro predikci bylo použito měření z PMU 202). Pro třídu BSP (blízký spínací prvek) je predikce 100 % správná v místě výskytu sledované události, ale chybná v intervalu Frame 244 – Frame 260, kde je chybně nahlášena skoková změna zatížení. Pro predikci bylo použito měření z PMU 702.

Pro třídu SZZ (skoková změna zatížení) je predikce 100 % správná v místě výskytu sledované události (Frame 119 – Frame 205), ale falešně pozitivní SZZ v intervalu Frame 340 – 362, kde se žádná SZZ nevyskytovala. Dále jsou také chybně hlášeny BSP a Z1F (postupně) v intervalu Frame 363 – Frame 425. Pro predikci bylo použito měření z PMU 101.

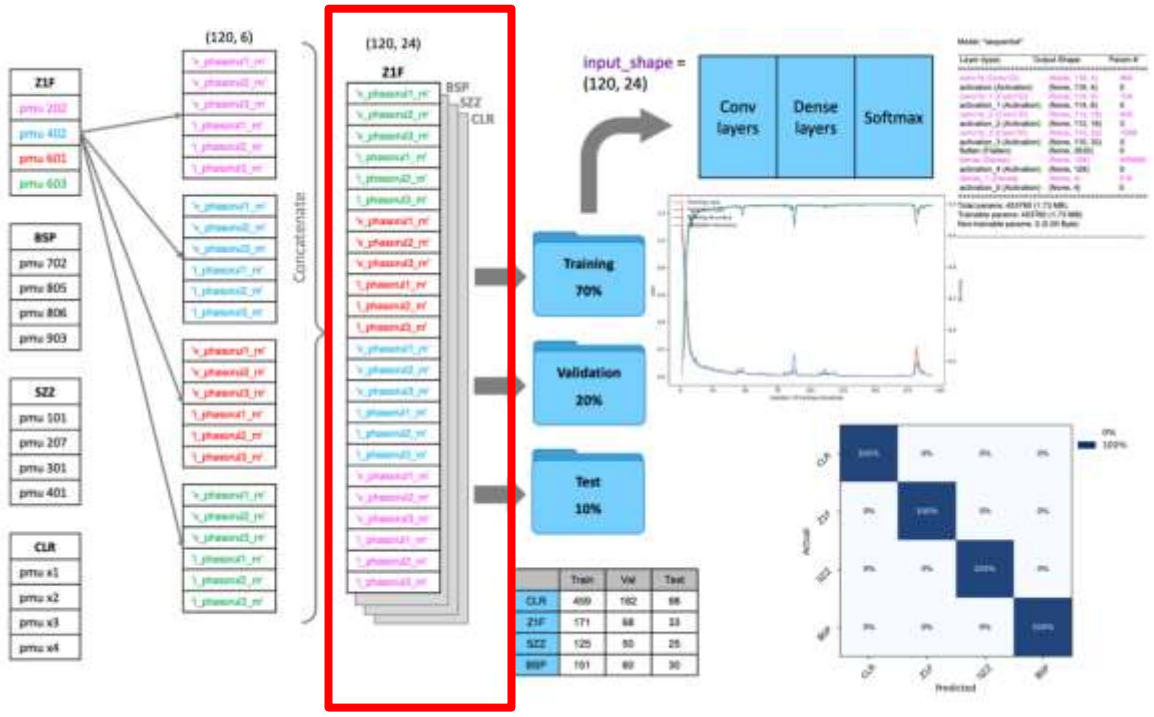
Je třeba poznamenat, že v tomto experimentu nebyla data z jednotlivých PMU seskupena, tj. v souboru dat byla každá PMU nezávislým blokem dat. To platilo jak pro trénování, tak pro testování neuronové sítě. Diskutovaná struktura (PMU jako nezávislé datové bloky zvýrazněny červeně) dat je znázorněna na Obr. 5.9.



Obr. 5.9 Způsob předzpracování dat – PMU jako nezávislé datové bloky

Je však možné aplikovat i rozdílný přístup, kdy jsou data ze všech PMU zapojených do detekce událostí seskupena do 2D tenzorů (viz Obr. 5.10), a teprve poté jsou tyto tenzory rozděleny do datových sad pro trénování, validaci a testování neuronové sítě. Experiment ukázal, že takový přístup přináší vyšší přesnost klasifikace než nezávislé datové bloky PMU.

Na Obr. 5.9 a 5.10 je rovněž zobrazena matice záměny (konfuzní matice), z níž je patrné, že v prvním případě byla klasifikační přesnost pro Z1F 89 %, SZZ 98 % a BSP 90 % (Obr. 5.9). Zatímco v druhém případě byla přesnost klasifikace ve všech případech 100 % (Obr. 5.10). Data z jednotlivých PMU jsou tedy seskupena do jednoho 2D tensoru (zvýrazněno červeně na Obr. 5.10).



Obr. 5.10 Způsob předzpracování dat – PMU sdružené do 2D datových tensorů

V Tab. 1 jsou uvedena všechna poskytnutá data .CSV a dále data, která z nich byla vybrána pro použití v neuronové síti. Při uvažování čtyř PMU a šesti parametrů v každém z nich s velikostí okna 120 vzorků, získáme 2D tensor o velikosti (24, 120).

Tab. 1 Výběr dat pro trénování neuronové sítě

Všechny poskytnuté datové sady	Vybrané datové sady
tbl.ts tbl.f tbl.dfdt tbl.v_phasorul1_m tbl.v_phasorul1_a tbl.v_phasorul2_m tbl.v_phasorul2_a tbl.v_phasorul3_m tbl.v_phasorul3_a tbl.i_phasoril1_m tbl.i_phasoril1_a tbl.i_phasoril2_m tbl.i_phasoril2_a tbl.i_phasoril3_m tbl.i_phasoril3_a	tbl.v_phasorul1_m tbl.v_phasorul2_m tbl.v_phasorul3_m tbl.i_phasoril1_m tbl.i_phasoril2_m tbl.i_phasoril3_m

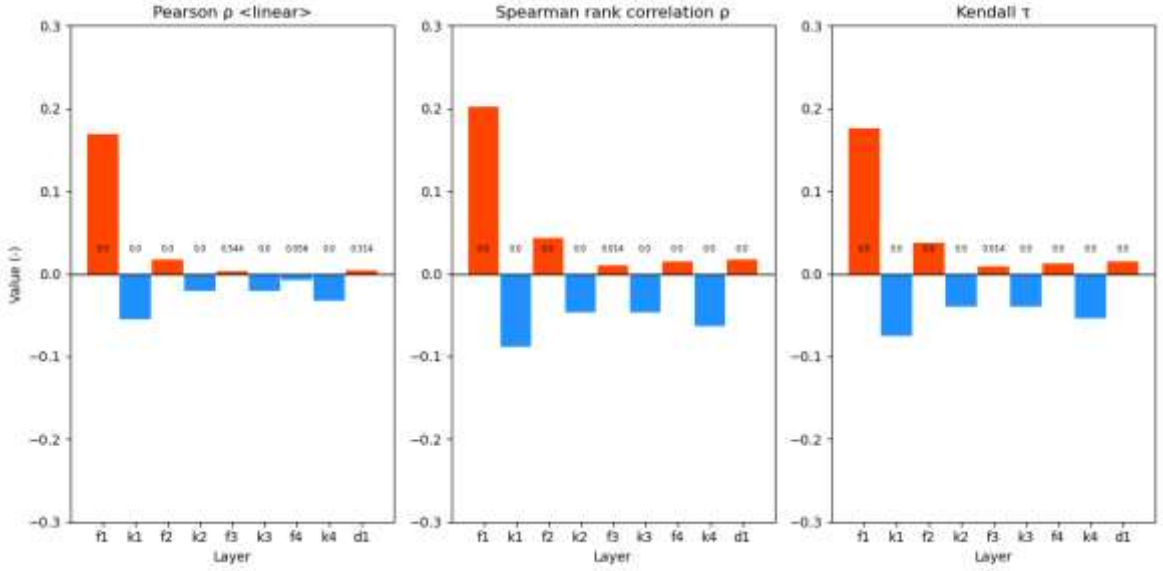
5.2.4 Optimální nastavení konfigurace neuronové sítě

K určení optimální konfigurace neuronové sítě byla použita metoda hrubé síly (brute force). Trénování bylo provedeno s použitím grafického procesoru NVIDIA na 62 208 vygenerovaných konfiguracích neuronové sítě (parametry níže). Na Obr. 5.11 je znázorněna existence přímé úměry mezi velikostí počtu filtrů f1, f2, f3, f4 a d1 a přesností klasifikace. Naopak mezi velikostí konvolučních jader k1, k2, k3 a k4 existuje nepřímá úměra. Přesnost klasifikace pro všechny konfigurace je znázorněna na Obr. 5.12. Pro lepší přehlednost je prezentována pouze velikost

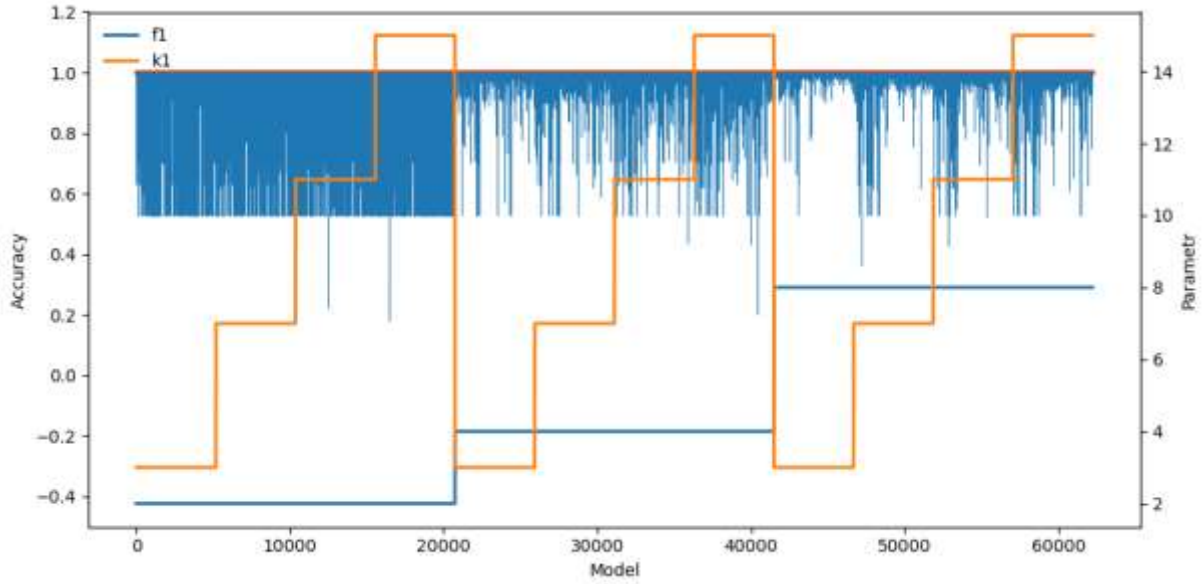
filtru f1 a velikost jádra k1, ze kterých je jasně patrné, že se zvětšující se velikostí f1 se přesnost klasifikace zvyšuje, zatímco se zvětšující se velikostí jádra k1 se přesnost klasifikace snižuje.

Parametry pro první generování konfigurací neuronové sítě:

f1 = [2, 4, 8]
k1 = [3, 7, 11, 15]
f2 = [4, 8, 16]
k2 = [3, 7, 11, 15]
f3 = [8, 16, 32]
k3 = [3, 7, 11, 15]
f4 = [16, 32, 64]
k4 = [3, 7, 11, 15]
d1 = [32, 64, 128]



Obr. 5.11 Korelační koeficienty pro každou vrstvu a velikost jádra (konfigurace 62208)

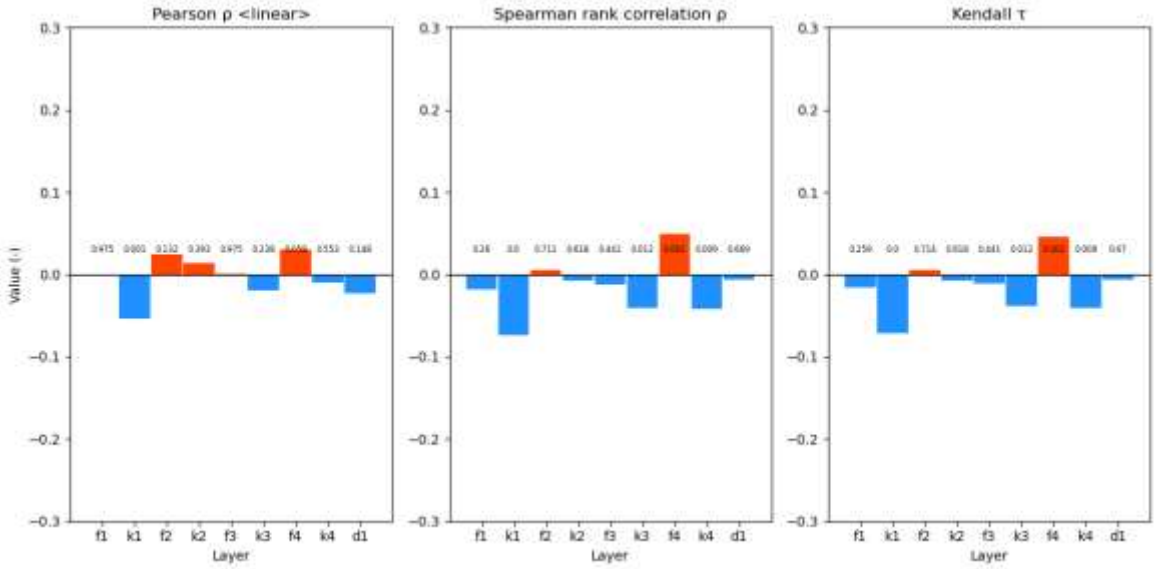


Obr. 5.12 Přesnost klasifikace pro všechny konfigurace neuronové sítě (konfigurace 62208)

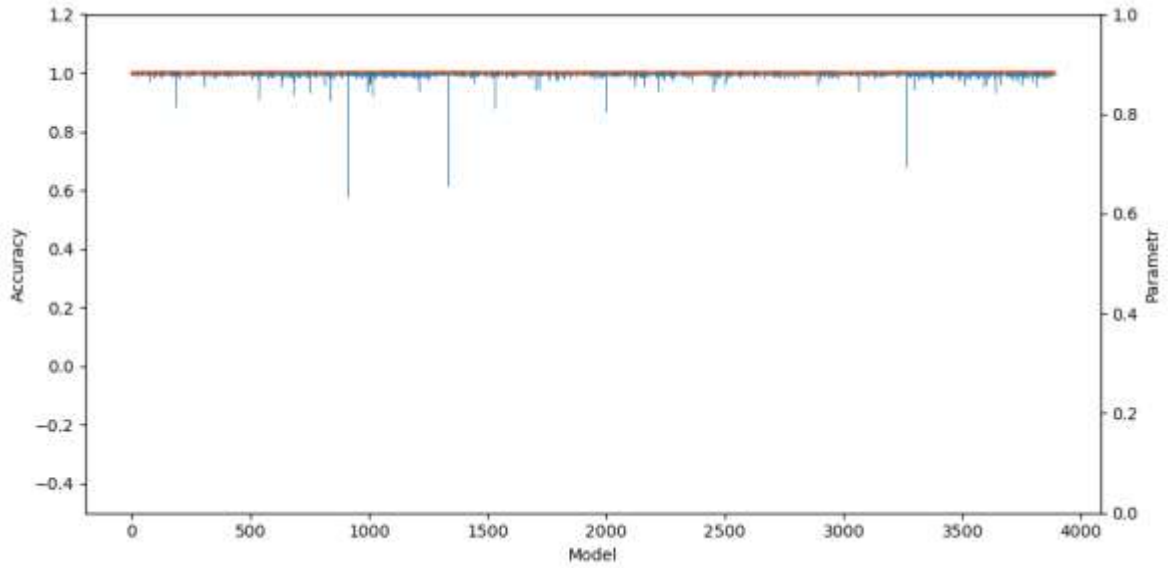
S ohledem na nové poznatky byly upraveny parametry neuronové sítě a následně byla provedena další fáze hledání optimálních parametrů, tentokrát však už pouze na 3 888 vygenerovaných variantách. Parametry těchto 3 888 variant jsou uvedeny níže. Na Obr. 5.13 jsou znázorněny korelační koeficienty pro jednotlivé vrstvy a velikosti jádra. Na Obr. 5.14 je pak patrné, že vzhledem k novým parametrům má téměř každá konfigurace neuronové sítě v rámci získaných 3 888 variant velmi vysokou klasifikační přesnost.

Parametry pro druhé generování konfigurací neuronové sítě:

f1 = [8, 16, 32]
k1 = [3, 7]
f2 = [8, 16, 32]
k2 = [3, 7]
f3 = [8, 16, 32]
k3 = [3, 7]
f4 = [16, 32, 64]
k4 = [3, 7]
d1 = [64, 128, 256]



Obr. 5.13 Korelační koeficienty pro každou vrstvu a velikost jádra (konfigurace 3888)



Obr. 5.14 Přesnost klasifikace pro všechny konfigurace neuronové sítě 3888

5.2.5 Zvýšení přesnosti klasifikace pomocí smíšených trénovacích, validačních a testovacích dat

Níže jsou uvedeny výsledky ověřovacího experimentu v rámci tréninku s použitím souboru dat druhého zkratu. Smyslem experimentu je tedy využít data, která popisují stejný děj, ale jsou získána z jiného časového úseku a/nebo jiného PMU. Teoreticky lze předpokládat, že při využití smíšených dat by mělo dojít k zvýšení přesnosti klasifikace neuronové sítě.

Experiment však přinesl smíšené výsledky při dané konfiguraci časového okna a randomizaci dat. Použití smíšených dat může vést k 100% přesnosti klasifikace, ale také k naprosto nesprávné identifikaci.

Všechny varianty experimentu jsou uvedeny v Tab 2. Specifické události použité pro testování jsou:

Zkrat „03.08.2023“
pmu: 202, **402**, **601**, 603 – Třída **Z1F**

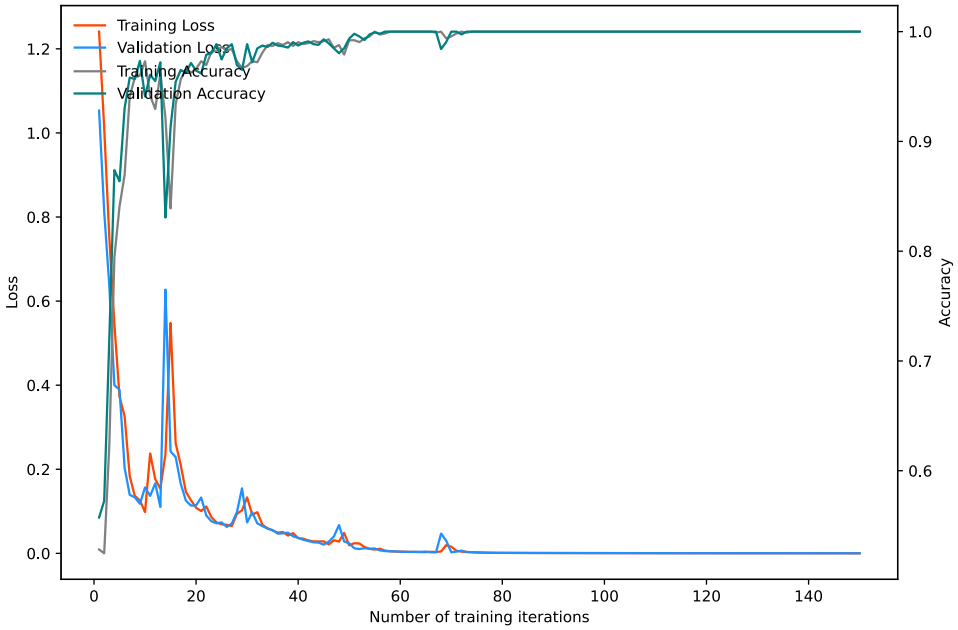
Zkrat „02.08.2023“
pmu: **402**, 401, 403, 301 – Třída **Z1F1**
pmu: **501**, 502, 503, 504 – Třída **Z1F2**

Tab. 2 Varianty smíšení trénovacích, validačních a testovacích dat

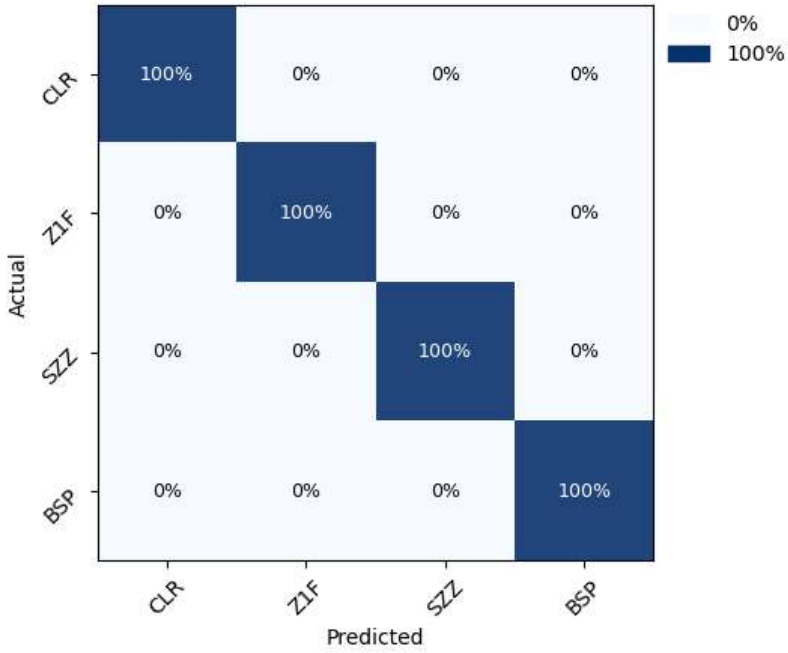
Variant	Train	Validation	Test
A	Z1F	Z1F	Z1F
B	Z1F	Z1F	Z1F1
C	Z1F	Z1F	Z1F2
D	Z1F	Z1F1	Z1F
E	Z1F	Z1F1	Z1F1
F	Z1F	Z1F1	Z1F2
G	Z1F	Z1F2	Z1F

H	Z1F	Z1F2	Z1F1
I	Z1F	Z1F2	Z1F2

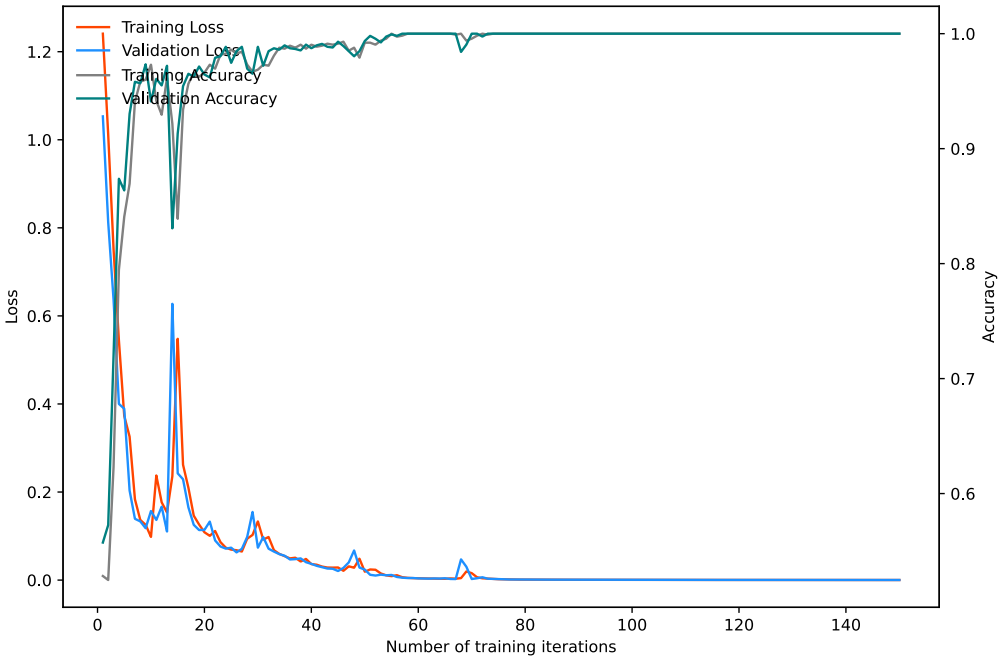
5.2.5.1 Grafické znázornění výsledků klasifikace pomocí smíšených dat



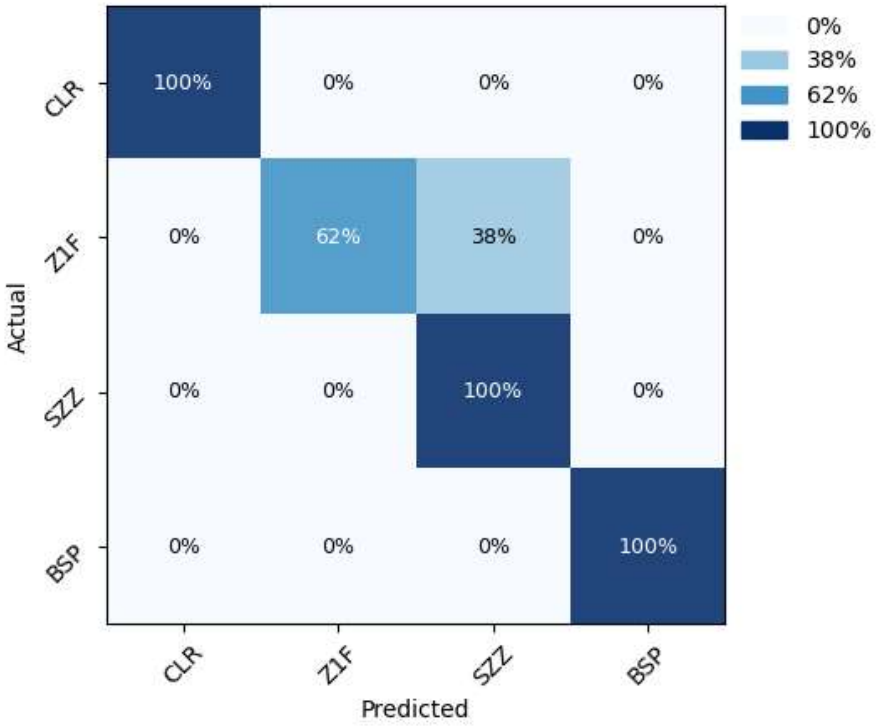
Obr. 5.15 Průběh trénování neuronové sítě – Varianta A



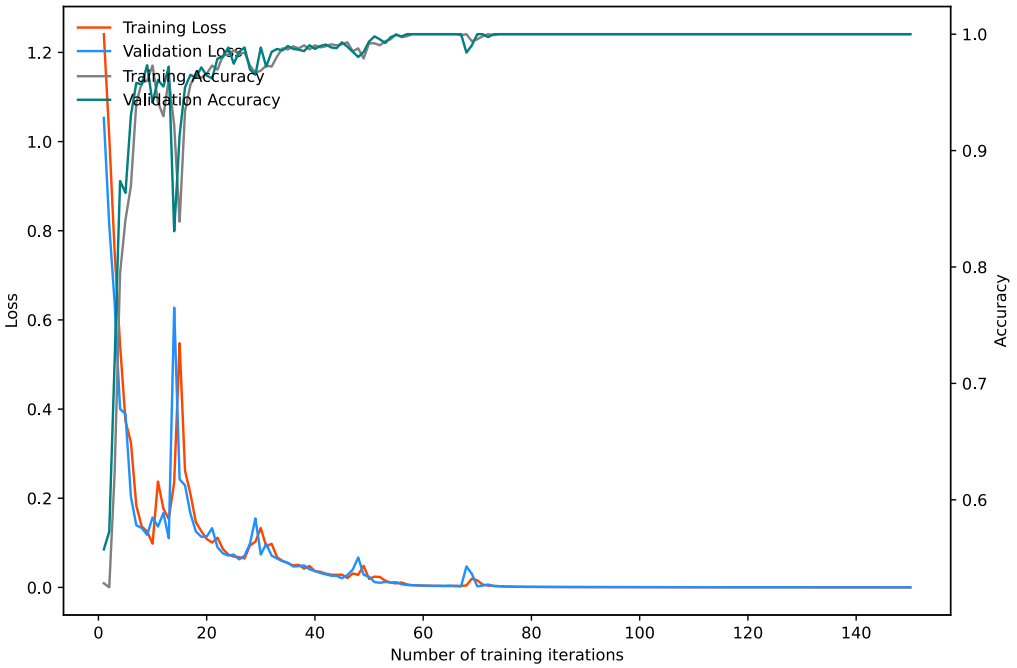
Obr. 5.16 Přesnost detekce provozních stavů – Varianta A



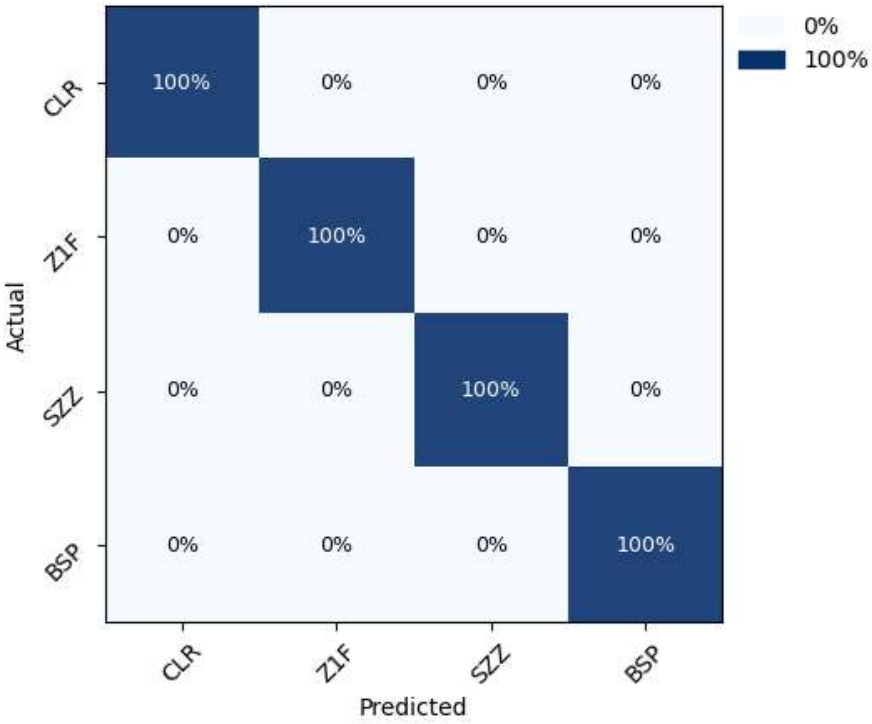
Obr. 5.17 Průběh trénování neuronové sítě – Varianta B



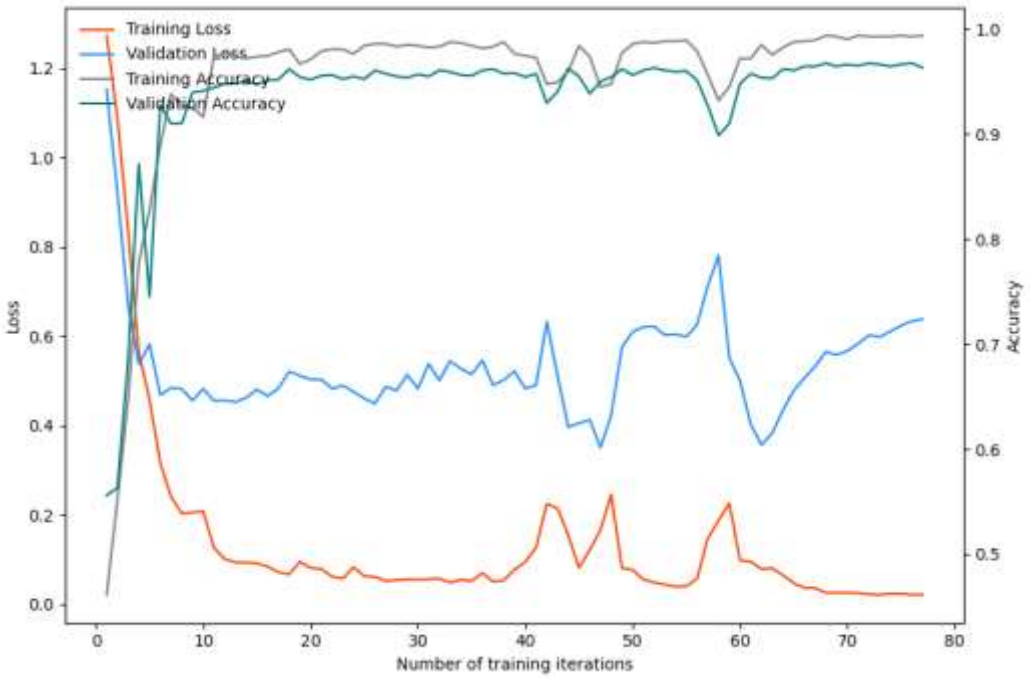
Obr. 5.18 Přesnost detekce provozních stavů – Varianta B



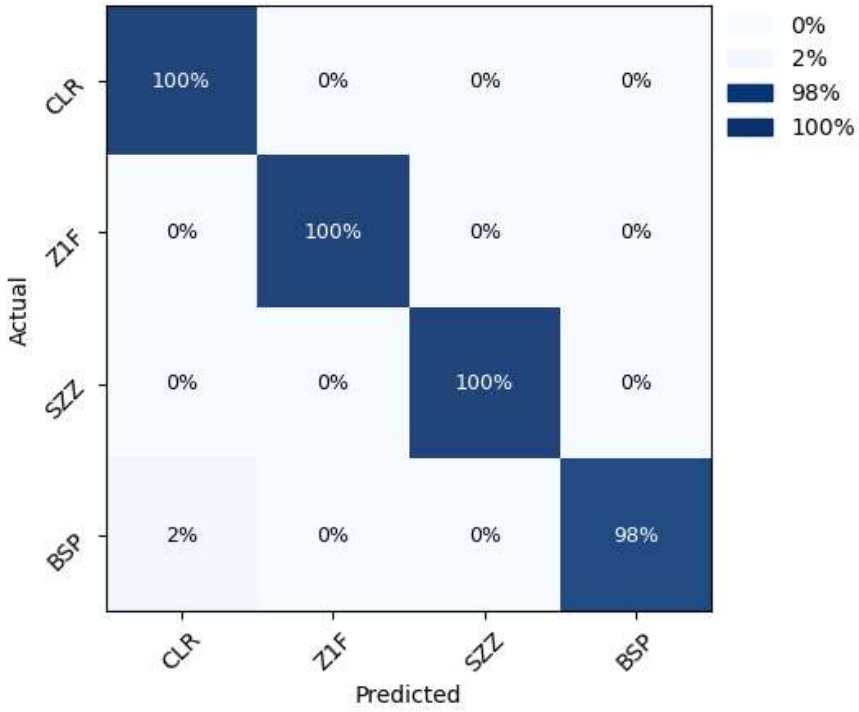
Obr. 5.19 Průběh trénování neuronové sítě – Varianta C



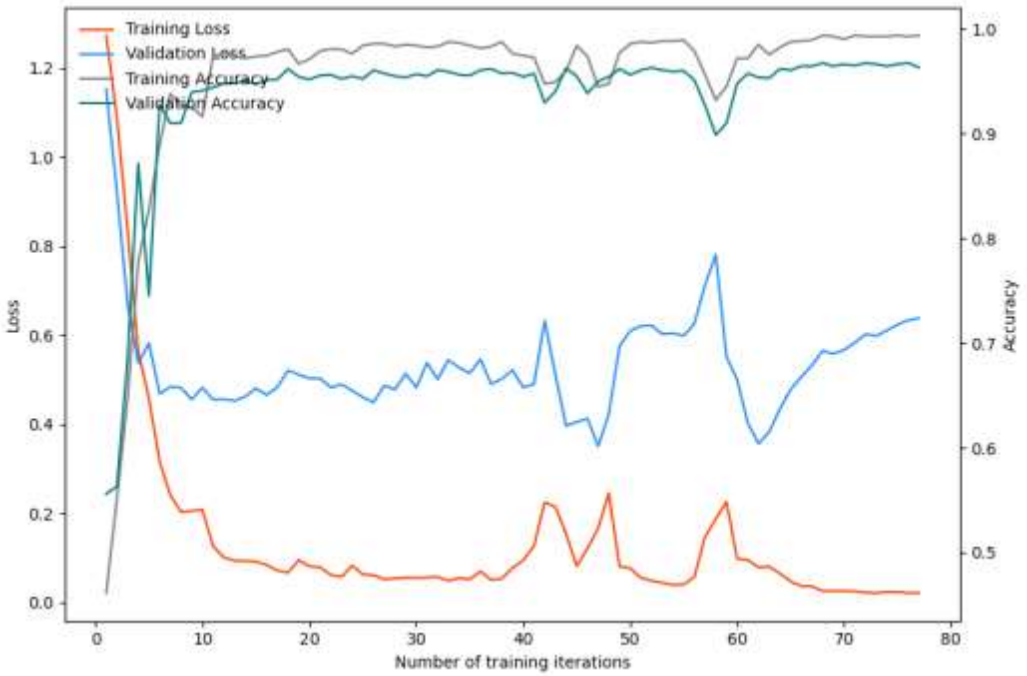
Obr. 5.20 Přesnost detekce provozních stavů – Varianta C



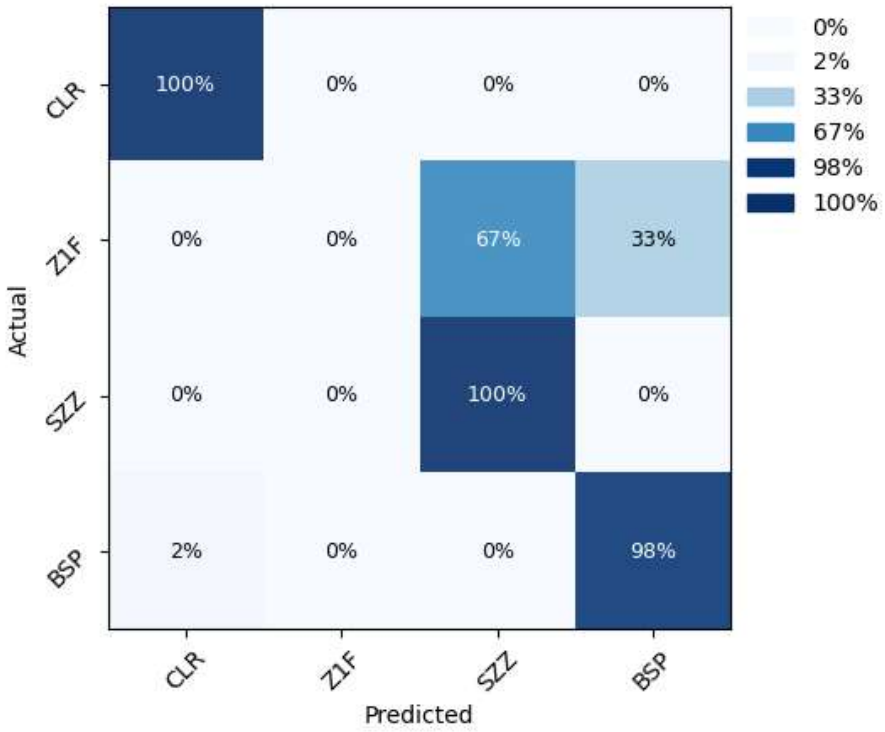
Obr. 5.21 Průběh trénování neuronové sítě – Varianta D



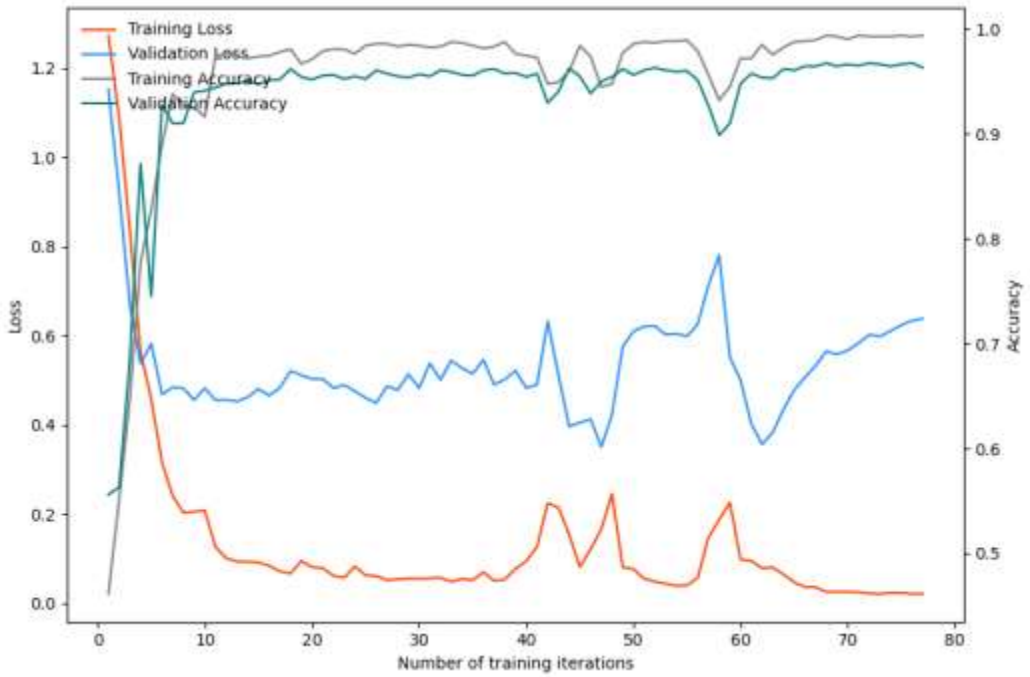
Obr. 5.22 Přesnost detekce provozních stavů – Varianta D



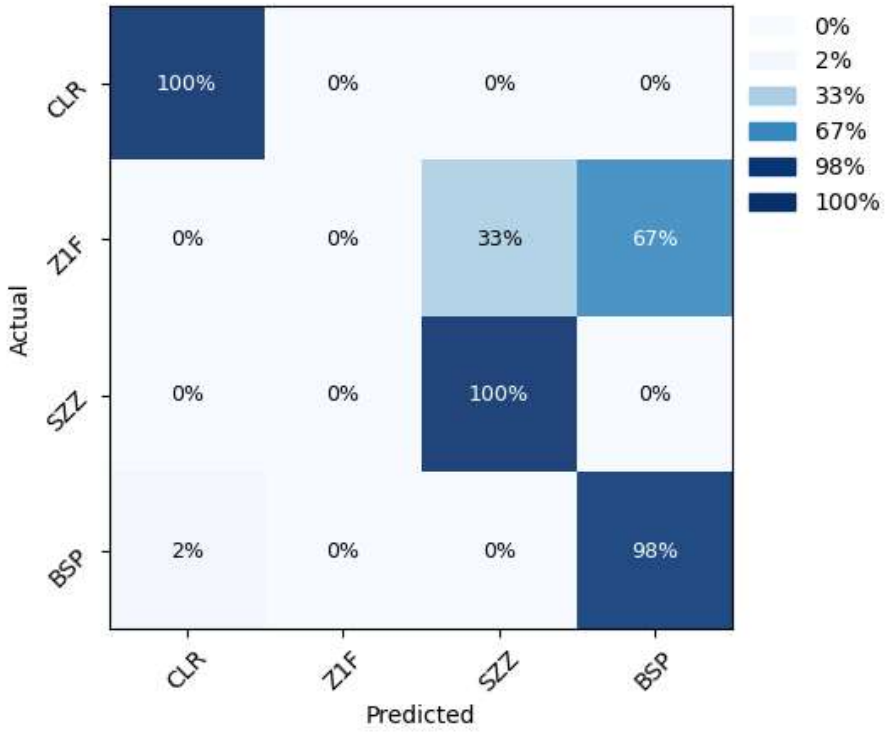
Obr. 5.23 Průběh trénování neuronové sítě – Varianta E



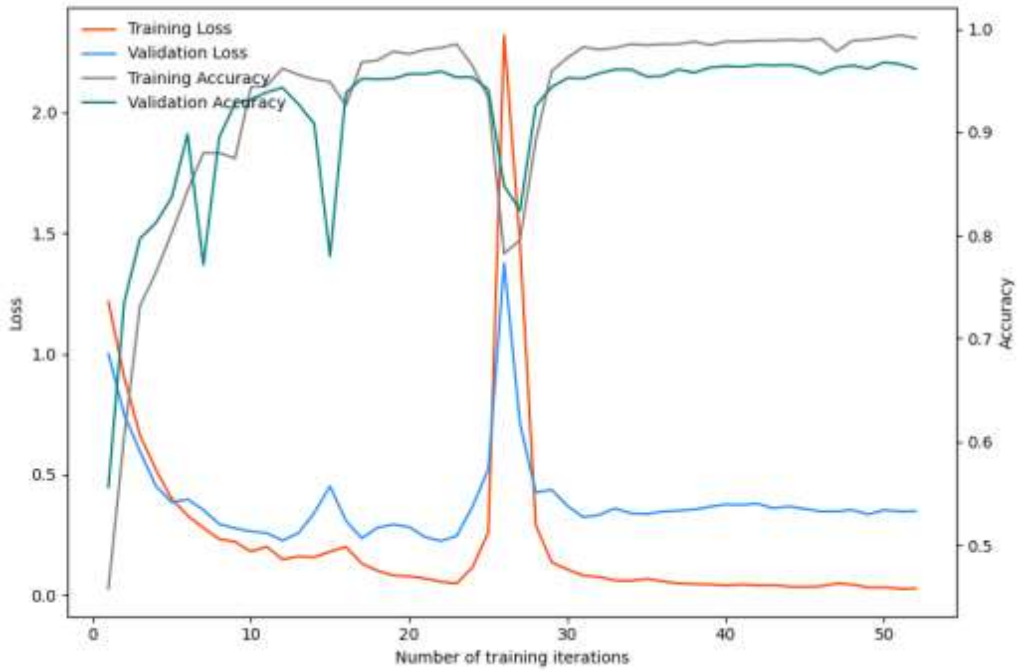
Obr. 5.24 Přesnost detekce provozních stavů – Varianta E



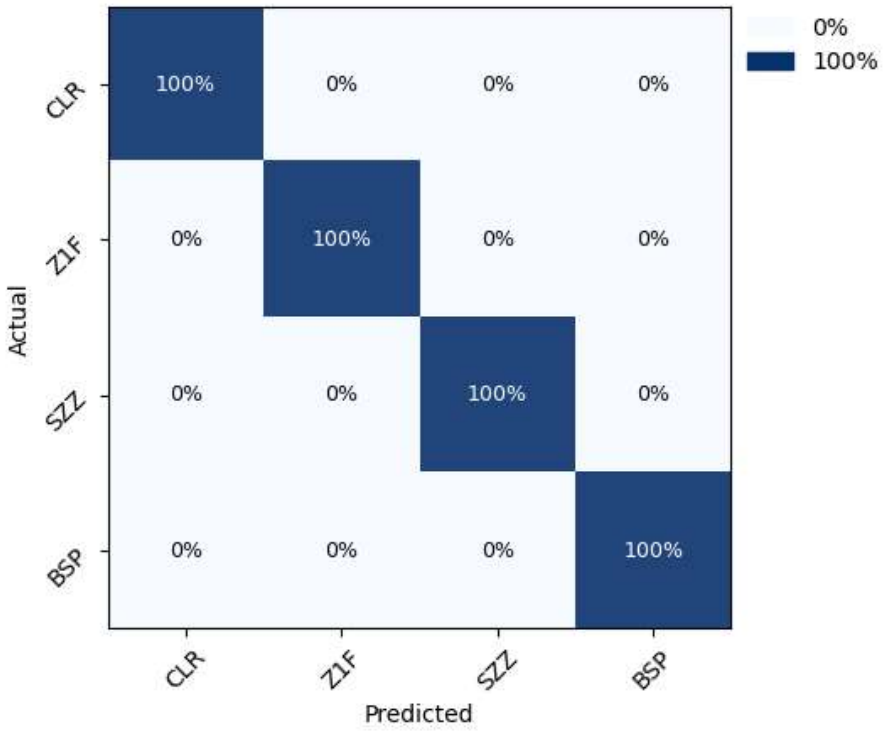
Obr. 5.25 Průběh trénování neuronové sítě – Varianta F



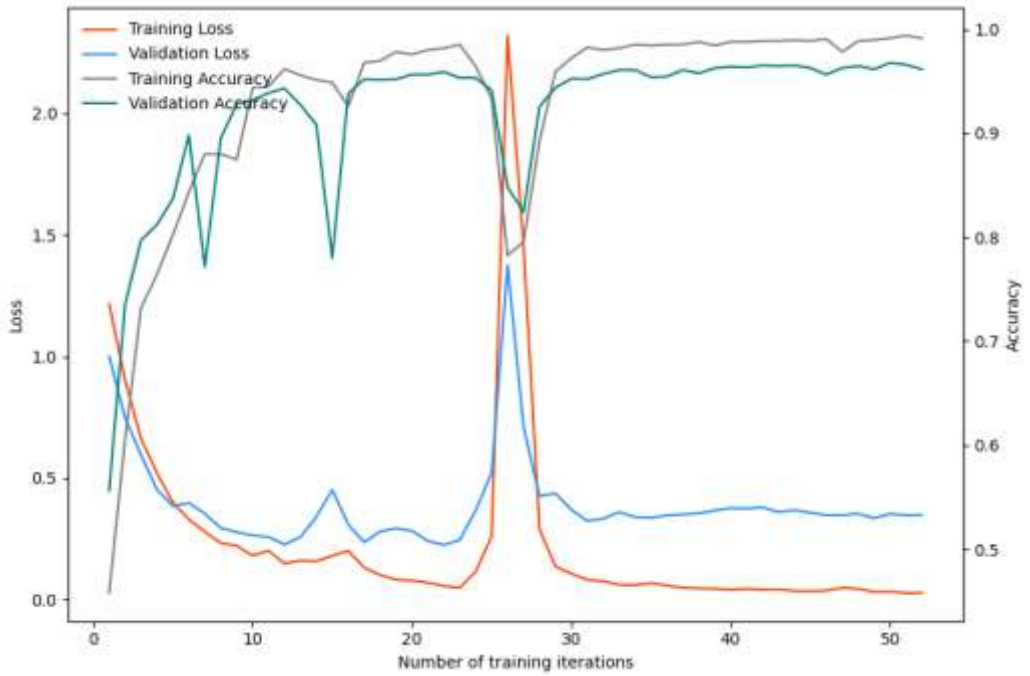
Obr. 5.26 Přesnost detekce provozních stavů – Varianta F



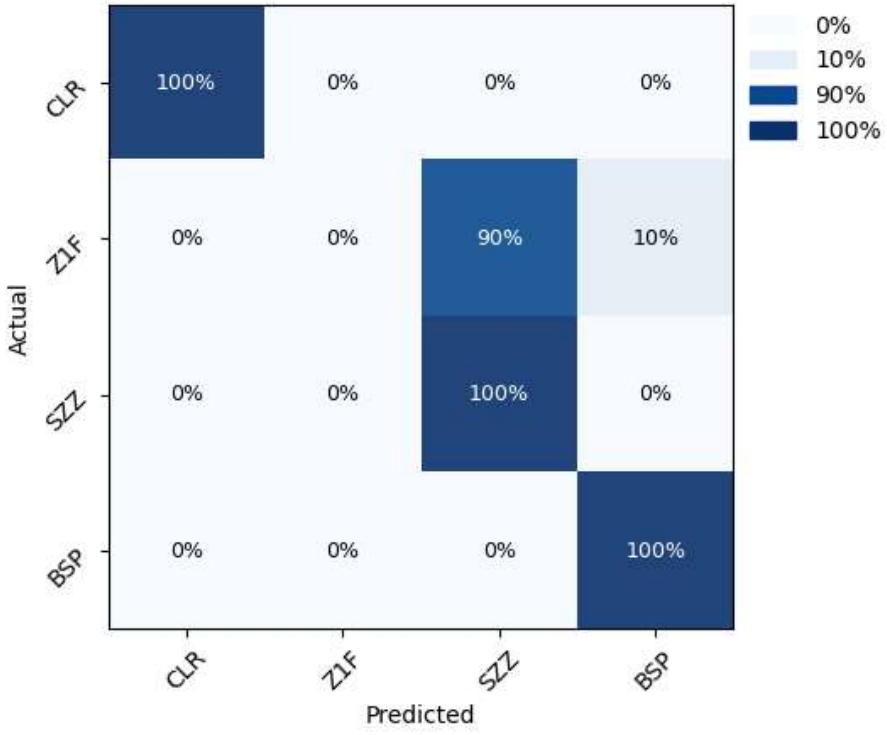
Obr. 5.27 Průběh trénování neuronové sítě – Varianta G



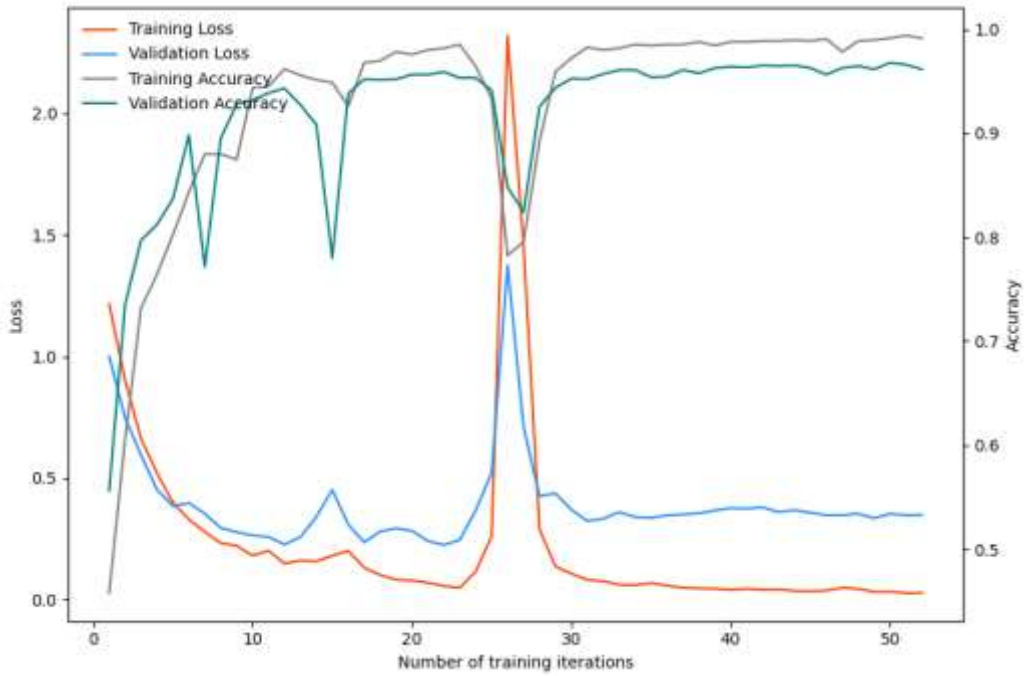
Obr. 5.28 Přesnost detekce provozních stavů – Varianta G



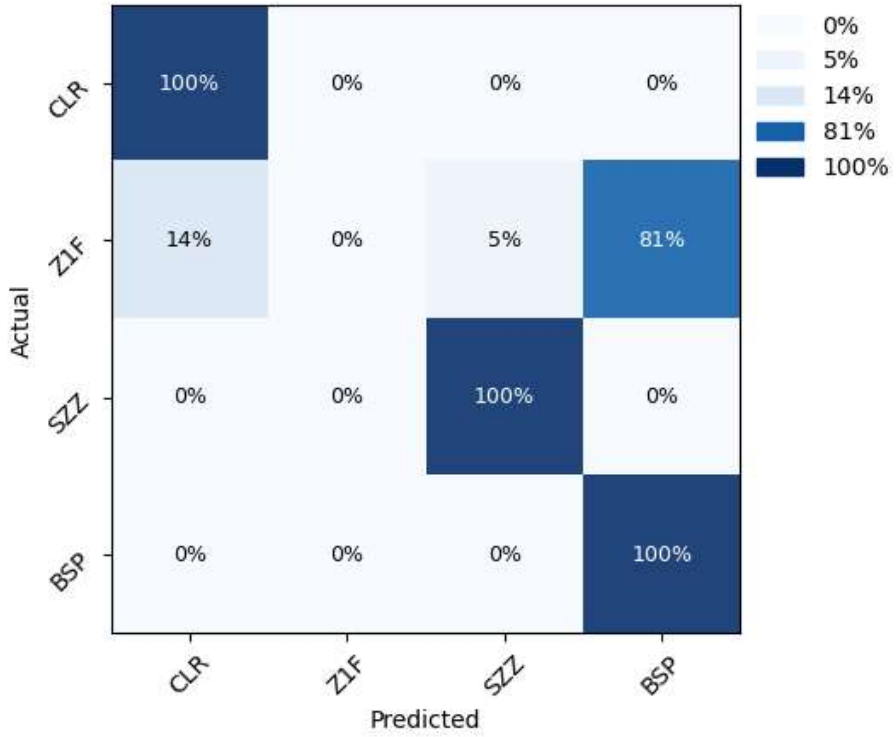
Obr. 5.29 Průběh trénování neuronové sítě – Varianta H



Obr. 5.30 Přesnost detekce provozních stavů – Varianta H



Obr. 5.31 Průběh trénování neuronové sítě – Varianta I



Obr. 5.32 Přesnost detekce provozních stavů – Varianta I

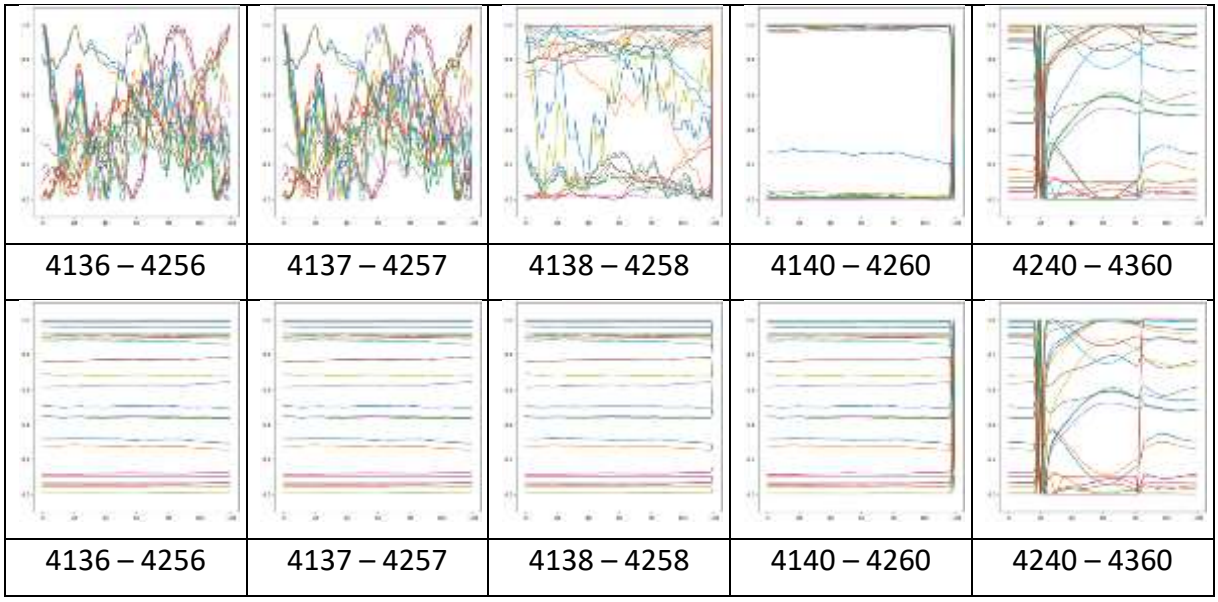
5.2.6 Normalizace signálu

Při srovnání normalizace v krátkém okně a normalizace celé časové posloupnosti, lze z Obr. 5.33, 5.34 a 5.35 vypozorovat, že při normalizaci v krátkém časovém okně dochází k zesílení šumu. To může negativně ovlivnit kvalitu signálu a jeho následnou analýzu. Tento efekt se projevuje v důsledku většího vlivu ,,náhodných rušení“, která jsou jinak obvyklou součástí spektra signálu.

Na druhé straně, normalizace celé časové posloupnosti umožňuje lépe zohlednit globální charakteristiky signálu, čímž se rušení lépe vyhlazuje a výsledná normalizovaná hodnota poskytuje plynulejší a stabilnější signál.

Obr 5.33 poskytuje vizuální srovnávací pohled na efekt různých metod normalizace při události Z1F.

- Horní řádek – normalizace v okně
- Spodní řádek – normalizace celé časové posloupnosti

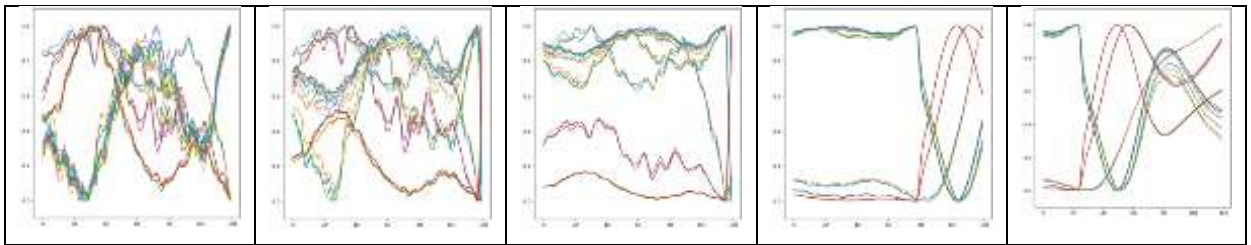


4136 – 4256	4137 – 4257	4138 – 4258	4140 – 4260	4240 – 4360
4136 – 4256	4137 – 4257	4138 – 4258	4140 – 4260	4240 – 4360

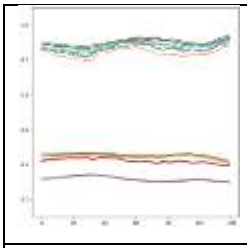
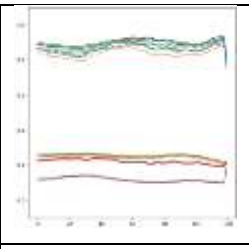
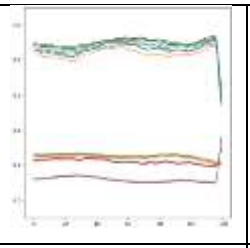
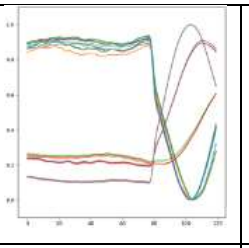
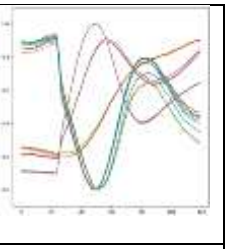
Obr. 5.33 Normalizace dat z události Z1F

Obr 5.34 poskytuje vizuální srovnávací pohled na efekt různých metod normalizace při události BSP.

- Horní řádek – normalizace v okně
- Spodní řádek – normalizace celé časové posloupnosti



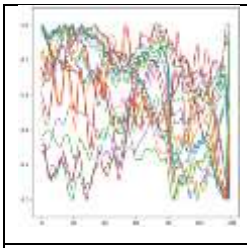
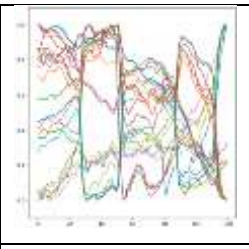
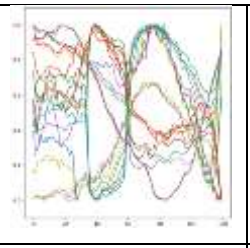
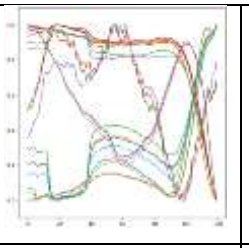
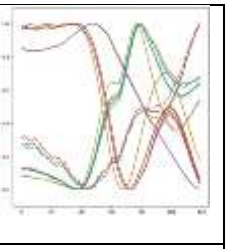
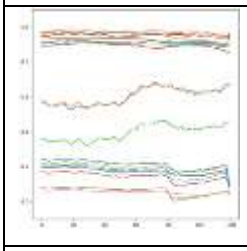
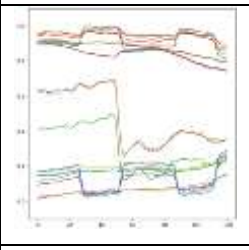
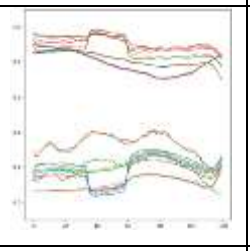
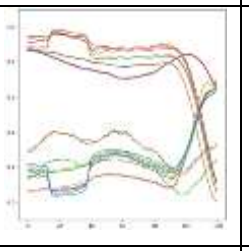
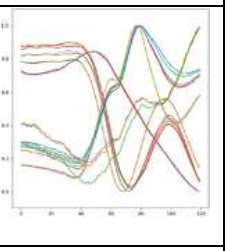
2015 – 2135	2017 – 2137	2019 – 2139	2057 – 2177	2111 – 2231

				
2015 – 2135	2017 – 2137	2019 – 2139	2057 – 2177	2111 – 2231

Obr. 5.34 Normalizace dat z události BSP

Obr 5.34 poskytuje vizuální srovnávací pohled na efekt různých metod normalizace při události SZZ.

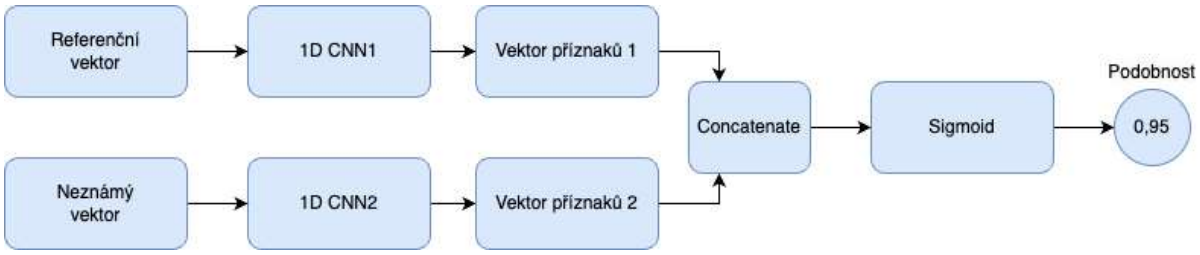
- Horní řádek – normalizace v okně
- Spodní řádek – normalizace celé časové posloupnosti

				
4779 – 4899	4870 – 4990	4923 – 5043	4944 – 5064	4997 – 5117
				
4779 – 4899	4870 – 4990	4923 – 5043	4944 – 5064	4997 – 5117

Obr. 5.35 Normalizace dat z události SZZ

5.2.7 Způsob trénování neuronových sítí

Pro řešení problému se šumem při normalizaci dat v krátkém časovém okně bylo rozhodnuto použít jiný typ trénování neuronových sítí, tedy učení typu few-shot (few-shot learning). V podmínkách nedostatečného množství tréninkových dat může tento přístup výrazně zvýšit efektivitu rozpoznávání vzorů. Struktura navržené neuronové sítě je znázorněna na Obr. 5.36.



Obr. 5.36 One(few) shot 1D CNN: Contrastive Loss

Datové sady Z1F, SZZ a BSP byly použity k vytvoření vektorových párů pro trénování neuronové sítě. Datová sada CLR nebyla použita vůbec, jelikož obsahuje náhodné šumy, které ji činí nevhodnou pro daný typ úlohy (viz Obr. 5.33, 5.34, 5.35).

5.2.8 Podrobný popis navrženého výsledného řešení

K rozpoznání události je využita neuronová síť natrénovaná metodou few-shot. Tento přístup umožňuje efektivně generalizovat síť na nové úkoly nebo data i po vystavení pouze malému množství příkladů (tzv. "few-shots").

5.2.8.1 Popis algoritmu pro předzpracování dat

Za implementaci tohoto algoritmu je zodpovědná třída „ShiftSamples“, která se nachází ve skriptu „B1_shift_samples“.

Ze všech proměnných obsažených ve vybraných souborech CSV (které byly vybrány pomocí dříve popsaných algoritmů) je použito šest. Jedná se o tři hodnoty proudu v každé ze tří fází a hodnoty napětí v každé ze tří fází. Délka okna je 120 vzorků.

Proměnná, která definuje tyto kanály, se nazývá „make_channels“:

```
self.make_channels = ['tbl.v_phasorul1_m', 'tbl.v_phasorul2_m', 'tbl.v_phasorul3_m',
                      'tbl.i_phasoril1_m',                      'tbl.i_phasoril2_m',
'tbl.i_phasoril3_m']
```

Proměnná, která určuje velikost okna, se nazývá „window“:

```
self.window = 120
```

Metoda „pmu_csv_to_npy“ ve třídě „ShiftSamples“ převádí výchozí soubory s příponou CSV na soubory s příponou NPY.

Metoda „get_pmu_n“ vytváří seznam všech souborů CSV obsažených ve vstupním adresáři a generuje proměnnou „pmu_n“ obsahující cesty k těmto souborům.

Metoda „concatenate_pmu“ načte proměnnou „pmu_n“, sloučí všechny kanály do jednoho poolu a normalizuje hodnoty obsažené v kanálech do intervalu 1-0. Získaný výsledek se zapíše do proměnné „channels_concatenated“.

Metoda „make_shifted_samples“ obdrží proměnnou „channels_concatenated“ a provede proces postupného posunu okna podél časové osy získaných šesti kanálů. Okno se vždy posune o jeden vzorek, výsledek se zkopíruje a uloží na disk do složky /input_csv/raw_samples. Výsledek se ukládá ve dvou formátech, grafickém s příponou PNG a ve formátu NPY.

Výše uvedený postup by měl být proveden pro každou událost, která má být identifikována. Poté je nutné provést proces výběru nejvhodnějších vzorků a vybrané vzorky umístit např. do složky „event_BSP“, která bude dále reprezentovat celou třídu. Všechny třídy pak budou tímto způsobem umístěny do adresáře „event“:

```
/event/event_BSP
/event/event_SZZ
/event/event_Z1F
```

Výše popsaný postup koreluje s kapitolou 1. Kontrola údajů (**1. Data inspection**) v příručce (Manual).

Následující text se vztahuje ke kapitole 2. Tvorba dat a trénování modelu. Třída DataShaping, která se nachází ve skriptu B2_data_shaping.py a která obsahuje všechny potřebné metody, je zodpovědná za proces generování vstupních dat pro neuronovou síť a její vlastní trénování

Aby tato třída pracovala správně, je třeba jí poskytnout následující vstupní parametry:

- Cesta do hlavního adresáře – path
- počet epoch – epoch
- velikost dávky – batch
- velikost okna – window
- počet kanálů – channels
- počet PMU – number_of_pmu

Standardní hodnoty jsou již přednastaveny:

```
path='C:/Users/om/Desktop/FinalCut_Shot/‘
epoch=1
batch=20
window=120
channels=6
number_of_pmu=4
```

Jinak skript funguje zcela automaticky. Název metody „separate_train_test“ hovoří sám za sebe, je zodpovědná za rozdělení dat na trénovací a testovací datovou sadu. Získané soubory ve formátu NPY se ukládají do adresáře event_npy:

```
/event_npy/x_train_samples_pmu.npy
/event_npy/y_train_samples_pmu.npy
/event_npy/x_test_samples_pmu.npy
/event_npy/y_test_samples_pmu.npy
```

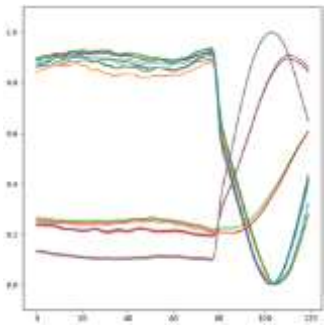
Neuronová síť se trénuje pomocí metody model_one_shot_training. Natrénovaný model s upravenými parametry se následně uloží do složky /network_dir/MODEL1D.h5.

V této fázi lze přikročit k implementaci procesu detekce událostí. Za to je zodpovědná třída FinalCut, která se nachází ve skriptu s názvem B3_predict_one_shot.py. Soubory CSV s detekovanou a rozpoznávanou událostí jsou umístěny v adresáři /input/.

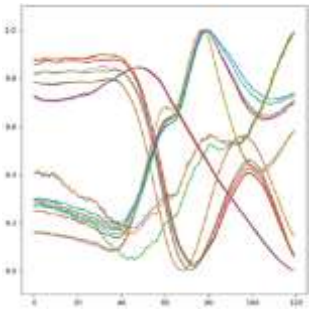
Za identifikaci události je zodpovědná metoda model_predict.

Protože se jedná o učení one-shot learning, je nutné poskytnout neuronové síti vzorky událostí, které má detekovat a rozpoznat. K tomuto účelu slouží adresář /anker/, který obsahuje následující podadresáře obsahující příslušné vzorky:

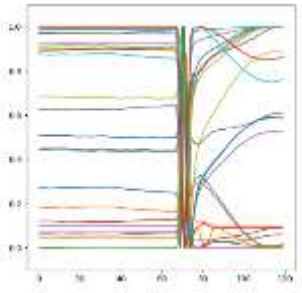
```
/anker_BSP/anker.npy (vzor udalosti BSP)
/anker_SZZ/anker.npy (vzor udalosti SZZ)
/anker_Z1F/anker.npy (vzor udalosti Z1F)
```



Obr. 5.37 Vzor BSP pro porovnání s neznámým vzorem



Obr. 5.38 Vzor SZZ pro porovnání s neznámým vzorem



Obr. 5.39 Vzor Z1F pro porovnání s neznámým vzorem

Metoda postupně načte každý vzorek události (kotvu) a postupně jej porovná s každým oknem ve zdrojovém souboru CSV (což bude nyní NPY). A určí, jak daleko je z hlediska podobnosti aktuální vzorek, který má být porovnán, se vzorovým vzorkem. Poté metoda vezme další vzorek události a rovněž jej porovná s analyzovaným souborem. Takto postupuje v případě všech detekovaných událostí. Výsledek porovnání se zapíše do souboru TXT *output_header_index.txt.*

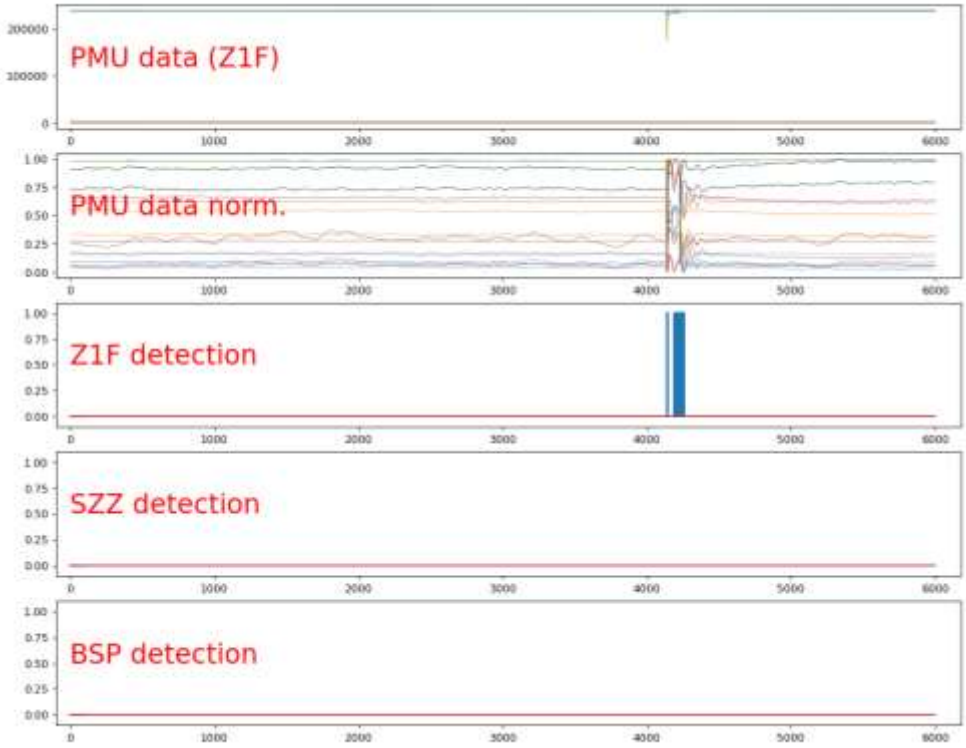
Formát je následující:

	Anker 0	Anker 1	Anker 2	DATE_TIME
0	6.192094e-07	9.993420e-01	3.420331e-04	2023-08-03 03:37:00.000
1	6.576857e-07	9.992684e-01	3.780920e-04	2023-08-03 03:37:00.020
2	7.139409e-07	9.992281e-01	3.853670e-04	2023-08-03 03:37:00.040
3	7.282632e-07	9.991923e-01	4.011645e-04	2023-08-03 03:37:00.060
4	6.500097e-07	9.989304e-01	5.464040e-04	2023-08-03 03:37:00.080
5	5.781649e-07	9.987609e-01	6.409663e-04	2023-08-03 03:37:00.100
6	5.331153e-07	9.987133e-01	6.723304e-04	2023-08-03 03:37:00.120
7	4.837078e-07	9.987929e-01	6.428040e-04	2023-08-03 03:37:00.140
…				
N				

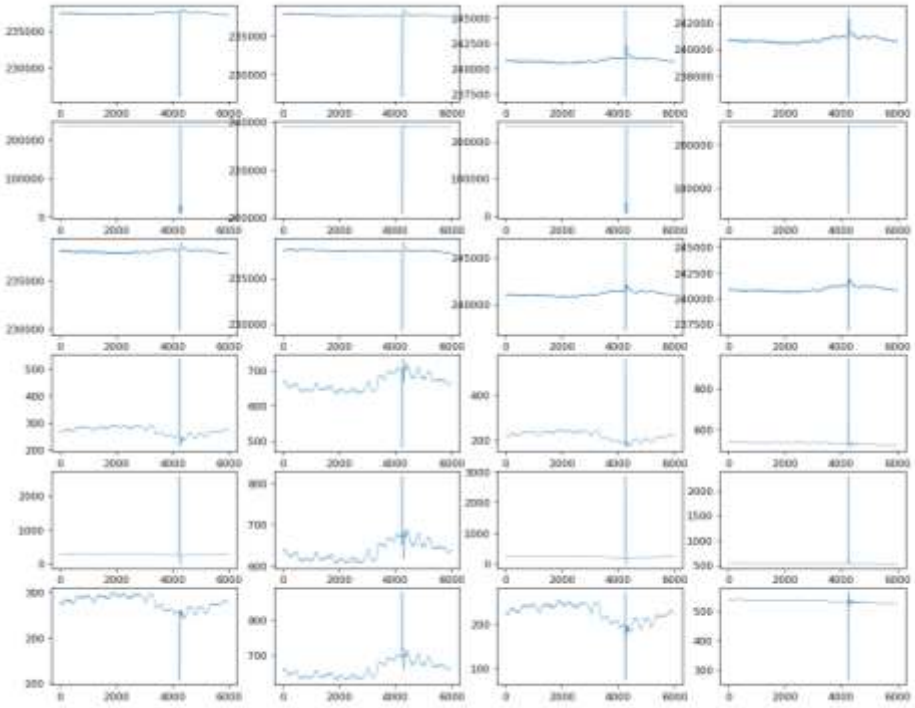
Tento soubor může být následně využit k určenému účelu. Příkladem využití je vizualizace dat v grafu. Za vizualizaci je zodpovědná třída FinalVisual, která patří do skriptu *B4_visualisation.py.*

Jak je patrné z Obr. 5.40, algoritmus úspěšně detekoval událost - jednofázový zkrat (Z1F) v místě časové osy, kde k němu skutečně došlo. Je však třeba poznamenat, že tohoto výsledku bylo dosaženo při velmi vysoké cut-off hodnotě (0,9999). Při nižších hodnotách se na grafech BSP a SZZ objevuje falešně pozitivní spouštění. To lze pravděpodobně vysvětlit omezeným množstvím dat pro trénink neuronové sítě.

Na Obr. 5.40 je první graf shora Z1F bez normalizace, druhý graf je Z1F s normalizací celé délce okna, třetí graf je výsledek klasifikace pomocí kotvy (anchor) Z1F, čtvrtý graf je výsledek klasifikace pomocí kotvy SZZ a pátý graf je výsledek klasifikace pomocí kotvy BSP.



Obr. 5.40 Porovnání klasifikace s různými způsoby normalizace a kotvy



Obr. 5.41 Vizualizace signálu ze všech kanálů a PMU jednotek

6 Závěr

V průběhu projektu byly úspěšně vyvinuty a implementovány algoritmy umožňující efektivní sběr, analýzu a klasifikaci dat z WAMS systémů. Využití neuronových sítí v prostředích python a MATLAB umožnilo navržení aplikovatelného řešení pro identifikaci mimořádných provozních stavů. Hlavní výstupy zahrnují:

- **Efektivní architektura pro analýzu a vizualizaci dat**: Skripty v pythonu byly navrženy pro sběr a zpracování časosběrných dat z PMU, jejich následnou vizualizaci a přípravu pro trénink klasifikačního modelu.
- **Využití neuronových sítí**: Hlavní implementace v pythonu nabídla flexibilitu a škálovatelnost, zatímco verifikace v MATLABu poskytla vizuální interpretaci a možnost ladění parametrů s využitím grafických nástrojů. Experimentální část odhalila klíčové faktory, jako je délka časového okna a metody normalizace dat, které zásadně ovlivňují přesnost modelu.
- **Výsledky klasifikace**: Algoritmus úspěšně klasifikoval klíčové provozní stavy, přičemž některé experimenty dosáhly přesnosti až 100 %. Nicméně analýza ukázala, že falešně pozitivní detekce zůstávají výzvou zejména u méně jasných stavů.
- **Plán dalšího vývoje**: Budoucí práce se zaměří na metody few-shot learning, které zvýší robustnost modelu při omezeném množství trénovacích dat. Rovněž je plánováno rozšíření a diverzifikace datasetu, což umožní lepší adaptaci algoritmu na reálné provozní podmínky.
    - Sběr dat bude posílen o další principy selekce časových intervalů dle průběžně získávaných relací mezi výpisem hlášení a časovými daty.
    - Učící data budou obohacena o data získaná simulačními výstupy nad modelem elektrizační soustavy.

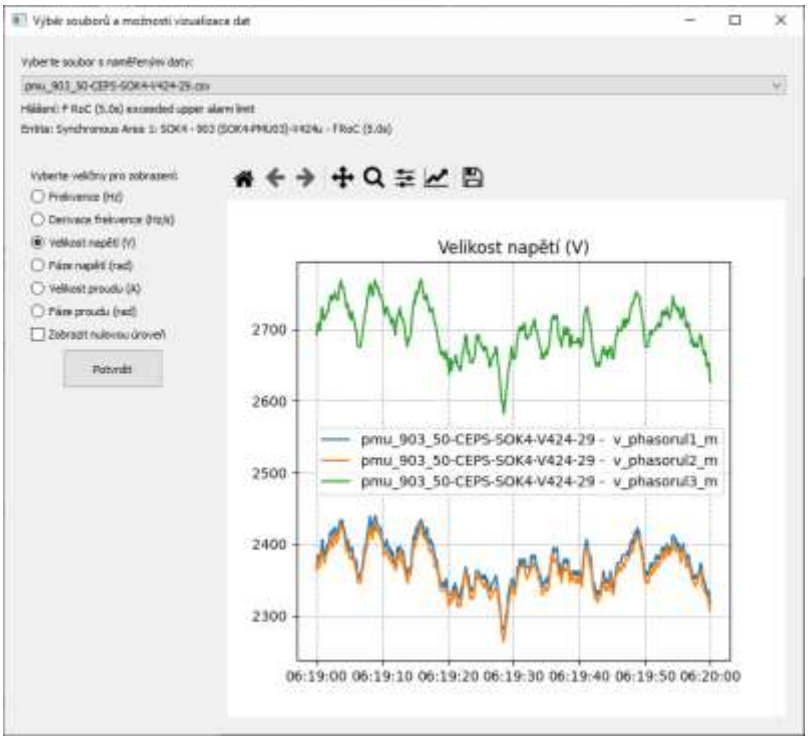
Přílohy

Příloha A – Ukázka výstupu skriptu načtení a analýzy zvoleného systémového výpisu hlášení (LOGu)

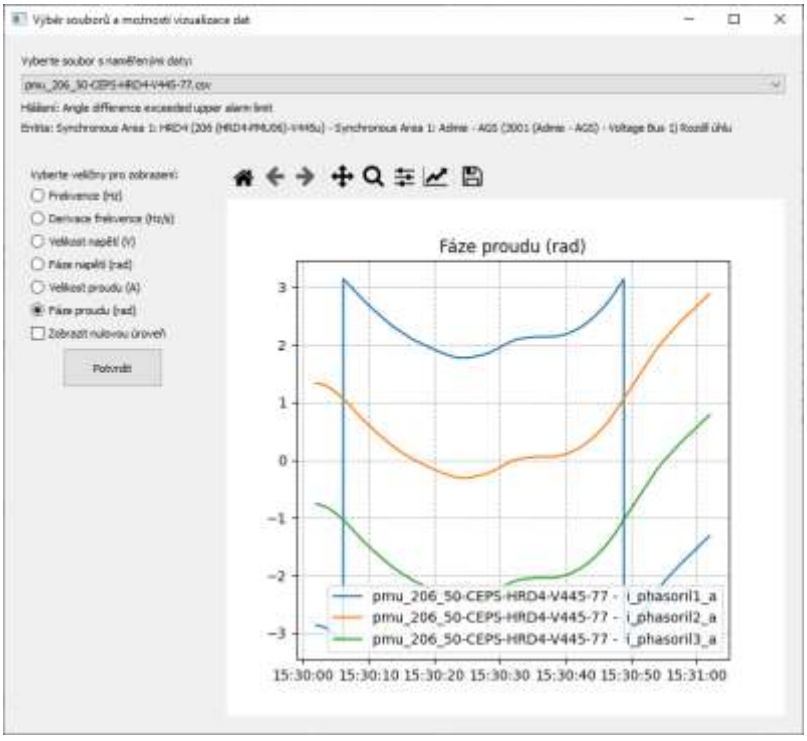
```
Počty hlášení dle četnosti:
Message
Positive Sequence Angle RoC (0.39s) fell below lower alarm limit                                27302
Positive Sequence Angle RoC (0.39s) exceeded upper alarm limit                                  27244
PDX1-3 event status alarm                                                                         890
Positive sequence voltage magnitude exceeded upper alarm limit                                    848
Angle difference fell below lower alarm limit                                                     766
PDX1-3 detected line down                                                                          75
Calculated exceeded upper alarm limit                                                              73
__EMS_ISLAND_ALARM exceeded upper alarm limit                                                      68
F RoC (5.0s) fell below lower alarm limit                                                          21
Positive sequence voltage magnitude fell below lower alarm limit                                  11
Angle difference exceeded upper alarm limit                                                        10
PV Power stability margin exceeded alarm limit                                                      8
Angle Disturbance Event                                                                             4
Islanding condition detected for synchronous area Synchronous Area 1                                2
F RoC (5.0s) exceeded upper alarm limit                                                             1
Synchronous Area 1: Seps - LEM - 21082 (QLEMEV478) - V478i - Positive Sequence Angle RoC (0.39s)    1
Frequency exceeded upper alarm limit                                                                1
Frequency fell below lower alarm limit                                                              1
Name: count, dtype: int64

Počet unikátních hlášení:  18
Počet incidentů:  280
Interval prilis rozsahly pro incident cislo  55  v casech  2024-06-07 15:56:45 2024-06-07 17:48:05
```

Příloha B – Ukázka výstupu skriptu pro zobrazení vizualizace vybraných dat pro odborný rozbor a přípravu podkladů



Obr. 0.1 Vizualizace naměřených dat detailu velikosti napětí



Obr. 0.2 Vizualizace naměřených dat detailu velikosti napětí

Historie revizí

Rev.	Kapitola	Popis změny	Datum	Jméno
0	Všechny	Publikování dokumentu	5. 2. 2025	M. Vinš