
**Budapesti Műszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Makai Lajos Bence

# MOBILHÁLÓZATI KAPACITÁS-IGÉNY ELŐREJELZÉSE A JELZÉSHÁLÓZATI FORGALOM ELEMZÉSÉVEL

KONZULENSEK

Dr. Varga Pál

Tánczos László

BUDAPEST, 2022

# Tartalomjegyzék

# 1 Kivonat

A mobilhálózatokban jelentkező erőforrásigények a felhasználói mobilitástól és eszközhasználattól függően változatosan jelennek meg. A szolgáltatók folyamatosan fejlesztik a hálózat képességeit új technológiák bevezetésével, jelenleg többek között épp a hálózati funkciók virtualizálásával, mellyel képesek lehetnek az erőforrásokat a korábbinál jobban optimalizálni. A hálózati erőforrás-elosztást a tapasztalt forgalmi adatok alapján méretezik, ami mindaddig optimális, míg a jövőbeli forgalom hálózati igényei hasonlóak valamilyen korábbi eseményhez. Általában elmondható, hogy a társadalom nagy része a napi rutinja szerint él – például reggel munkába utazik az agglomerációból, majd délután haza – így az általuk generált mobilhálózati forgalom könnyen előre jelezhető. Ezzel szemben, az egyre jobban elterjedő home office kultúra következtében nem könnyű előre jelezni az egyes napok hálózati forgalmának volumenét és típusát. Az IoT (Internet of Things) végpontok forgalmának előrejelzési lehetősége is változó: a periodikus forgalmat generáló szenzorok forgalma, és a rendezvényekhez vagy kampányokhoz kötött IoT forgalom nehezen becsülhető. Ha képesek vagyunk megbecsülni, hogy a felhasználók statisztikailag hol, mikor milyen számítási-, memória- és sávszélesség-igényű szolgáltatásokat fognak használni, akkor a hálózati erőforrásokat hatékonyan tudjuk elosztani és allokálni a jövőben jelentkező igényekhez.

A maghálózati forgalom monitorozásával és feldolgozásával meg lehet határozni a felhasználók mozgását, amiből előre lehet becsülni a TA-k (Tracking Area) leterheltségét és az átlagforgalom trendszerű változásait. A jelzésváltás üzenetek várható értékének becslésére több módszer is használható, viszont a felhasználók viselkedése sokszor kiszámíthatatlan – például váratlan események bekövetkezése esetén – így a hagyományos matematikai idősoros elemzések várhatóan nem adnak elég pontos eredményt. Ezért gépi tanulási módszer alkalmazásával próbálok meg pontosabb becslést előállítani a várható jelzésváltási forgalom intenzitásáról a hálózatban. Az adatfeldolgozó neurális hálózat képes rejtett mintázatot is felfedezni a bemeneti adatsoron, melyeket figyelembe vesz a becslés során.

A mobilhálózatokban a RAN (Radio Access Network) és CN (Core Network) között a vezérlő- és adatforgalom külön interfészeken halad, és más hálózati entitásokat köt össze. Jelenleg a publikus 5G mobilhálózat még non-standalone képességekkel rendelkezik, így az LTE S1AP protokollon modellezem azt, amit majd később az 5G SA

hálózat NGAP protokollján tapasztalhatunk. A feladat során a publikus LTE mobilhálózat S1AP interfészének monitorozásával gyűjtök mintákat az adatfolyamból, ami az eNB-t és a core-ban lévő MME-t (Mobility and Management Entity) köti össze. Ezen az interfészen keresztül menedzseli a hálózat a felhasználók mozgásának kezeléséhez szükséges HO (Handover) és TAU (Tracking Area Update) üzeneteket. Ezen üzenetek alapján lehet meghatározni a felhasználók mozgását a mobilhálózatban, és különböző módszerekkel becslések készíthetők a várható hálózati igényekről. Dolgozatom célja annak vizsgálata, hogy a TAU és HO üzenetek elemzésével hogyan lehet a jelenleginél pontosabb előrejelzést adni a várható mobilhálózati igényekről és ezek alapján a jelenleginél optimálisabban méretezni a hálózati erőforrásokat.

# 2 Abstract

The resource demands in mobile networks change based on the user's mobility and device usage. Service providers are constantly improving the capabilities of the network by introducing new technologies, currently with virtualization of network functions, which help to optimize the network's resources better than before. Resource allocation is networks based on the experienced traffic data, which is optimal as long as the network needs of future traffic are similar as before. In general, it can be said that a large part of society lives according to their daily routine – for example, they commuting from the agglomeration in the morning and then return home in the afternoon – so the mobile network traffic they generate can be easily predicted. On the other hand, as a result of the increasingly widespread home office culture, it is not easy to predict the volume and type of network traffic for each day. The possibility of predicting the traffic of IoT (Internet of Things) endpoints also varies: the traffic of sensors generating periodic traffic, and IoT traffic tied to events or campaigns is difficult predict. If we are able to estimate, statistically, where and when the users will use what computing, memory and bandwidth demanding services, then we can efficiently distribute and allocate network resources for future needs.

By monitoring and processing the core network traffic, it is possible to determine the movement of users, from which the load of the TAs (Tracking Area) and trend-like changes in the average traffic can be estimated in advance. Several methods can be used to estimate the expected value of signal switching messages, but the behavior of users is often unpredictable - for example, in case of unexpected events - so traditional mathematical time series analyzes are not expected to give sufficiently accurate results. Therefore, by applying a machine learning method, I try to produce a more accurate estimate of the intensity of the expected signaling traffic in the network. The data processing neural network can also discover hidden patterns in the input data set. The same data will be considered during the estimation.

In mobile networks, the control and data traffic between the RAN (Radio Access Network) and CN (Core Network) travels on separate interfaces and connects other network entities. Currently, the public 5G mobile network operates in non-standalone mode, so I am modeling on the LTE S1AP protocol what we will experience later on the NGAP protocol of the 5G SA network. During the task, I collect samples

from the data stream by monitoring the S1AP interface of the public LTE mobile network, which connects the eNB and the MME (Mobility and Management Entity) in the core. Through this interface, the network manages the HO (Handover) and TAU (Tracking Area Update) messages required to manage the movement of users. Based on these messages, the movement of users in the mobile network can be determined and estimates of expected network demands can be made using different methods. The goal of my thesis is to examine, by analyzing TAU and HO messages, it is possible to provide a more accurate forecast than the current one about the expected mobile network needs and, based on these, to dimension the network resources more optimally than at present.

# 3 Bevezetés

A vezeték nélküli mobil távközlésben néhány kivétellel fontos szempont a felhasználók mobilitásának biztosítása, amely hatására a különböző igények időben változatosan jelennek meg a hálózat egyes szegmenseiben, lehetséges torlódásokat okozva egy-egy cellában. A hálózat erőforrás-elosztásának optimalizálásában nagy előrelépést jelent a virtualizáció és a felhő alapú technológiák. Egy virtualizált erőforrásokból felépített és szoftveresen vezérelt hálózatban (SDN - Software Defined Network) a korábban célhardverekkel megvalósított hálózati funkciókat szoftver szinten implementálják és általános, adatközpont topológiájú hardverközpontokban futtatják.

A hálózati virtualizációt adatközponti környezetben már korábban is alkalmazták, a mobil távközlésben viszont az 5. generációs hálózatok fejlesztésével került előtérbe. Az 5G SA (Standalone) maghálózatok már alapértelmezetten SBA (Service Based Architecture) architektúrára épülnek, melyben a hálózati funkciók virtualizálva vannak (NFV – Network Function Virtualization).

Ennek a mintájára meg lehet valósítani az LTE (Long Term Evolution) hálózatot is SBA-ként, hiszen a működés főbb irányelvei hasonlóak. A hálózat hatékony működésének szempontjából már az LTE is külön kezeli a felhasználói adatokat (UP – User Plane) a jelzésinformációktól (CP – Control Plane). Így a hálózat menedzsmentje és méretezése jobban tud igazodni az adott szolgáltatási terület igényeihez, valamint egy esetleges kapacitás bővítés is könnyebben elvégezhető. Ebben a dolgozatban a CP-re koncentrálva mutatok példát arra, hogyan lehet gépi tanulási (ML – Machine Learning) algoritmusok képességeit felhasználni a hálózati forgalom előrejelzésére.

Bár a virtualizáció nagyban elősegíti a rugalmas hálózatmenedzselést és egy esetleges bővítés is nagyon költséghatékony tud lenni, ez még nem elég, hiszen egy lefedettségi területen jelentkező igények időben nem állandóak. A forgalmi igényeket több tényező is befolyásolhatja. Az eMBB (enhanced Mobile Broadband) szolgáltatások esetén a legjelentősebb tényező az emberek mozgása, míg a V2X (Vehicle to everything) kommunikációban a járműveké. Nagyvárosok esetén például általánosan elmondható, hogy a város népessége (hétköznapokon) napközben jelentősen megnő, azáltal, hogy az agglomerációban és vidéken lakó munkavállalók a munkahelyükre beutaznak. De nem csak a városi mobilhálózat kapacitását kell erre felkészíteni, hanem a felhasználók által

utazás közben érintett cellákat is. Egy nagyvárosi forgalmas bevezetőúton ilyen esetben a munkaidő előtt és után sokszorosára nő a jelzésváltási forgalom, munkaidőben pedig az adatforgalom, ez időszakon kívül viszont nincs szükség akkora kapacitásra ezeken a helyeken. De ezt sem lehet minden napra általánosítani, hiszen a hétvégék, vagy az egyre több változó munkarendben dolgozó ember „összezavarhatják" a leegyszerűsített modelleket. Mivel az utóbbi években a sok munkahelyen megváltoztak a munkavégzés szabályai, így egyre nehezebben lehet az adott igényeket kiszámolni, hiszen sokan távmunkába dolgoznak és csak heti 2-3 napot töltenek a munkahelyen. A hálózati erőforrások elosztását sokkal optimálisabbá tehetjük, ha képesek vagyunk megbecsülni, hogy a felhasználók statisztikailag hol, mikor milyen számítási-, memória- és sávszélesség-igényű szolgáltatásokat fognak használni, és a hálózat erőforrásait már felkészülten, előre tudjuk allokálni a közeli jövőben jelentkező igényekhez.

Becslést többféle módszerrel és technikával elő lehet állítani, viszont a felhasználók „viselkedése" sokszor kiszámíthatatlan, így a hagyományos matematikai idősoros elemzések nem adhatnak elég pontos eredményt. Ezért döntöttem a gépi tanulás alkalmazása mellett erre a feladatra. A gépi tanuláson belül a DNN (Deep Neural Network) technológia segítségével mutatom meg a várható forgalmi igényekre vonatkozó becslési lehetőségeket.

A neurális hálózat képes rejtett mintázatokat is felfedezni az adatsorokon, rátanulni és figyelembe venni azokat a becslés során. Egy ilyen feladat során az egyik legfontosabb dolog a bemeneti adathalmaz, amin dolgozni fog a hálózat. Jelen helyzetben a felhasználók viselkedését legjobban reprezentáló adatok a mobilitás- és session-menedzsment adatok, melyekből kiderül a felhasználók mozgása és az általuk használt alkalmazások típusa; ebből lehet következtetni az igényeikre. A hálózathoz való hozzáférés és a megfelelő interfészen történő adatgyűjtés lehetősége mellett, a megfelelő adatfeldolgozás után előre lehet becsülni a TA-k (Tracking Area) leterheltségét és az átlagforgalom trendszerű változásait. A becslés előállításához fel lehet használni az előző napi, heti, havi vagy akár valós idejű adatokat. Ebben a dolgozatban offline adatfeldolgozással fogom tanítani a neurális hálózatot. A valós idejű adatfeldolgozás egy más jellegű problémamegoldást igényel, de nagy valószínűséggel abban az esetben is szükség lenne egy múltbéli adatbázisra, valamint sokkal nagyobb számítási kapacitásra, mert nagyon sok adat generálódik valós időben.

Virtualizált erőforrású hálózatok menedzsmentjéből számtalan publikáció született – ezek különféle algoritmusokat, funkciókat, módszereket mutatnak be és hasonlítanak össze, de kevés, ami konkrét megvalósítást tartalmaz és *élő hálózati* adatokon dolgozik.

Több irányadó publikáció született már hálózati forgalmak mély neurális hálóval történő osztályozásáról – ezek főbb ismérveit többek között Rezavei és Liu is összefoglalta 2019-ben [1]. A mobilhálózati forgalom további specialitásokat mutat, hiszen a sliceing megoldásokhoz különösen fontos a titkosított forgalom klasszifikációja. Ebből a célból már alkalmazzák a DNN technológiát mobilhálózati felhasználói adatokon is [2][3][4]. A jelzéshálózati forgalom alapján többek között hibaterjedés is becsülhető [5].

Az 5G és a későbbi mobilhálózati generációkban a jelzésforgalom becslése azonban különös jelentőséggel bír. Sokkal nagyobb forgalomra lehet számítani, mint a korábbi generációkban, és az előrejelzési feladat is fontos – mind a fenntarthatóság (green networking), mind a dinamikus erőforrás-allokáció céljai miatt. A mobilitási jelzésforgalom becsléséről azonban nagyon kevés publikáció elérhető még [7][7], valós adatokat vizsgáló munkák – amilyeneket a jelen dolgozat is bemutat – pedig különösen hiányosak. A mobilitás adatok, és azokra vonatkozó becslések nagyon jól kiegészítenek egy forgalmi igény modellt, hiszen így helyileg is, akár cella vagy TA szinten meg lehet határozni az igények helyét és volumenét.

Ebben a dolgozatban minél pontosabb becslést próbálok megadni a várható felhasználói mobilitási adatokról az *élő hálózatból* gyűjtött adatok vizsgálata alapján, mely segítséget jelent a virtualizált mobilhálózatok erőforrás-allokálási menedzsmentjében.

# 4 Műszaki háttér

Az új generációs mobilhálózatok bevezetésében a felhő alapú és virtualizációs technológiák kulcsfontosságúak. A fejlesztők célja a minél általánosabb hardverre épülő hálózatok megvalósítása, és ezáltal a költségek csökkentése. Ennek a megvalósításához új szabványokra is szükség volt, mely a különböző hálózati elemek közé interfészeket határoznak meg, így biztosítva a kompatibilitást akár különböző gyártók által forgalmazott, valamint a régebbi eszközök között. A teljes hálózat megvalósításához több technológia együttes alkalmazása szükséges, melyek különbözőek, de gyakorlatilag egymásra épülnek.

## 4.1 Hálózati virtualizáció

Ahhoz, hogy a dinamikus erőforrás-allokáció megvalósulhasson, célhardverek helyett virtualizált hálózatokra van szükség. A virtualizált hálózatok menedzsmentjét egy orkesztrátor végzi [8]. Feladata, hogy adott kapacitás-igényekhez allokáljon számítási kapacitást. A dolgozatban bemutatott neurális háló becslései adják a bemeneti adatokat az orkesztrátor számára, ami azok alapján megfelelő számítási kapacitásokat tud allokálni az egyes hálózati szegmensekbe.

*Software Defined Networking*

Definíció szerint a SDN [8] (Software Defined Network) egy olyan hálózati architektúra, ahol az adatsíkot (data plane – DP) alkotó hardverek vezérlését egy operációs rendszer végzi (control plane – CP), mely távolról hozzáférhető. Az SDN hálózatoknak fontos szempontjai vannak, amelyek hatékonnyá teszik, nem csak egy hálózati funkció szoftveres megvalósítását jelenti. Az egyik legfőbb, hogy a vezérlő- és adatsík külön van választva. A vezérlési funkciókat eltávolítják a hálózati eszközökről, melyek így egyszerű csomagtovábbító elemek lesznek a hálózatban. Az adattovábbítás folyamat alapú (flow control), mely során a hálózat az egy folyamathoz tartozó csomagokat együtt kezeli, megegyező szabályozással, ezeket egy táblában tartja számon. Az SDN-ben a folyamat egy adott forrás és cél közötti csomagok sorozatát jelenti. Ez lehetővé teszi a különböző típusú hálózati eszközök működésének egységesítését, beleértve a routereket, switch-eket és tűzfalakat. A folyamatprogramozás nagymértékű rugalmasságot eredményez a hálózatban, jelentős kapacitásbővülést lehet elérni, mely az implementált folyamattáblázattól függ. A vezérlési logika átkerül egy külső entitásba, az

SDN kontrollerbe. Az SDN kontroller egy szoftverplatform (operating system – OS), mely általános szerveren fut és biztosítja a hálózati csomagtovábbító elemek konfigurációját. A hálózat szoftveres alkalmazásokon keresztül programozható, melyek az OS-en futnak és együttműködnek az alapul szolgáló adatsík eszközeivel.

*Network Function Virtualization*

A hagyományos mobilhálózatokban az egyes hálózati funkciókat célhardverekkel valósítják meg, melyek bővítése és fejlesztése magas költségekkel jár a szolgáltatók számára. A hálózati funkciók virtualizálásával ezek a költségek csökkenthetők, valamint a folyamatok gyorsíthatók. Az NFV **Hiba! A hivatkozási forrás nem található.** (Network Function Virtualization) lehetővé teszi a szolgáltatók számára a hálózati funkciók szoftveres megvalósítását, melyek általános szervereken futnak virtualizációs technológiákat használva ahelyett, hogy valamilyen célhardveren futnának. Az egyre gyorsabban növekedő és változó felhasználói igények kiszolgálásához egyre nagyobb szükség van az NFV és SDN integrálására. Ezzel együtt a hálózat menedzselése is rugalmasabb lesz, lehetőség nyílik az automatizálásra is. Mivel a rendelkezésre álló erőforrások fizikai elhelyezkedésüktől függetlenül használatóak, és az egyes hálózati funkciók (Virtual Network Function – VNF) között tetszőlegesen csoportosíthatóak, így az adott felhasználói igényekhez lehet alakítani a hálózati erőforrásokat. Az NFV koncepciója az SDN-től származik, de nem teljesen egyforma a két technológia. Az NFV a hálózati alkalmazásokat a dedikált hardverről virtuális konténerekbe helyezi, míg az SDN egy magasabb szintű hálózatmenedzsmentet végez.

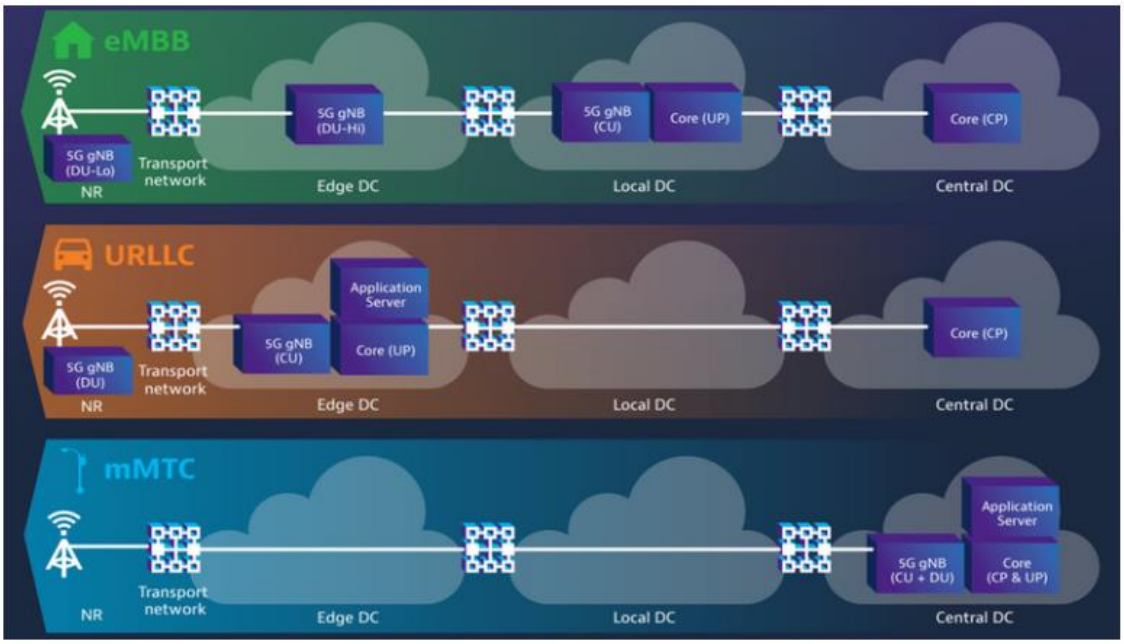
*Mobile Edge Computing*

A hálózatba kapcsolt okoseszközök típusai egyre különbözőbbek, és a legtöbbnél nem követik a mobiltelefonoknál látható trendet, hogy az alkalmazások igényeihez mérten egyre nagyobb számítási- és akkumulátor kapacitással szerelik fel azokat. Például az okos város koncepciókban a legtöbb okos eszköz nem rendelkezik a helyi döntések elvégzéséhez szükséges számítási kapacitással. A nyers adatokat elküldi a hálózatnak, mely továbbítja valamilyen szerver felé, ahol elvégződnek a számítások, majd az eredmények visszaküldésre kerülnek a készülékhez. Az ilyen szerverek (Application Server – AS) nagy számítási kapacitással rendelkeznek, de a QoS-t (Quality of Service) még így is nehéz betartani késleltetésre érzékeny alkalmazási típusoknál, hiszen az AS

nem biztos, hogy közel van a felhasználóhoz, így a hálózati késleltetés összemérhető, vagy akár nagyobb lesz, mint a feldolgozási idő. Erre jelenthet megoldást a Mobile Edge Computing (MEC) [11], melyben a cellás hálózatok peremén helyezik el a számítási kapacitást biztosító szervereket, így minimalizálva a felhasználói eszközöktől való távolságot és az ebből adódó késleltetést. Ezen kívül egy ilyen implementáció lényegesen csökkentené a mögöttes hálózat terhelését is, hiszen a számítási és tárolási kapacitás-igények gyakorlatilak lokálisan biztosítva lennének.

*Network Slicing*

A network slicing **Hiba! A hivatkozási forrás nem található.** az, amikor egy fizikai hálózatot több virtuális hálózatra osztanak, melyek különböző alkalmazásokra vannak optimalizálva. Ezek a virtuális hálózatok (slice-ok) egy közös hardveren jönnek létre és egymással párhuzamosan futnak, teljesen elszeparálva. A felhasználó szempontjából olyan, mintha egy dedikált privát hálózata lenne. A slice egy önálló hálózat saját virtuális erőforrásokkal, topológiával, folyamatokkal és bedefiniált szabályokkal. A slice-ok lehetnek egyformák, ha privát hálózatra van igénye például egy ipari felhasználónak, vagy különbözőek. Az egyes felhasználói típusoknak teljesen más erőforrás-igényeik vannak. NB-IoT (Narrowband IoT) szolgáltatáshoz például mobilitási funkciókat nem kell megvalósítani, mert az eszközöknek statikus helyük van és csak jelzés forgalmat generálnak, amin keresztül viszik át a felhasználói forgalmat is, viszont nagy számú kapcsolatot kell tudnia kezelni a hálózatnak. Ezzel szemben eMBB (enhanced Mobile Broadband) slice-nak nagy maghálózati kapacitás szükséges. Mobilitást támogató, valamint nagy adatátviteli sávszélességet és alacsony késleltetést biztosító hálózati funkciókkal kell rendelkeznie. A 2.1. ábrán egy lehetséges megvalósítás architektúrája látható, melyben együtt kerülnek alkalmazásra a fent említett technológiák.




**2.1. ábra: End-to-end network slicing architektúra.** Hiba! A hivatkozási forrás nem található.

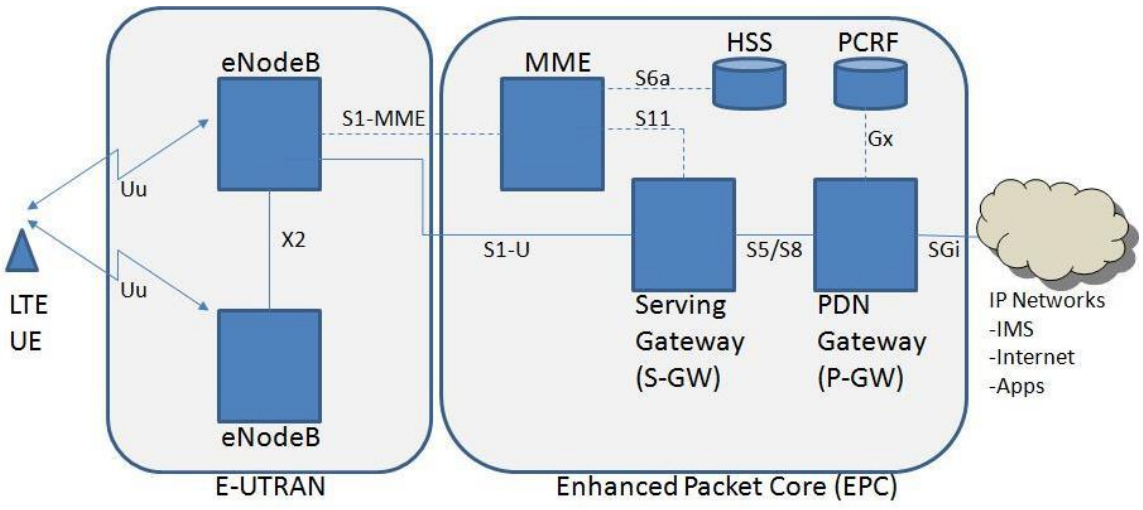
Bár a slice-ok logikailag elkülönülnek egymástól, de az erőforrásokat meg kell osztaniuk. A különböző slice-okat end-to-end kell megvalósítani, hogy megfelelhessenek az adott alkalmazástípus QoS követelményeinek. A maghálózati slicing megvalósításához elengedhetetlen az SDN és NFV technológiák alkalmazása. Az SDN elválasztja a vezérlő- és felhasználói adatforgalmat így azok külön kezelhetőek a hálózatban. A vezérlő sík központosított telepítése támogatja a menedzsment funkciókat, míg az adatsík eloszthatóvá, így például az alacsony késleltetésű slice-on haladó adatok becsatornázhatóak egy MEC szegmensbe. AZ NFV által pedig minden CP és DP hálózati funkció telepíthető, példányosítható, és a példányok száma dinamikusan változtatható.

## 4.2 Mobilhálózati adatgyűjtés

A mobilhálózatokat alkotó különböző hálózati szegmenseket szabványosított interfészek kötik össze, melyeken szintén szabvány szerint meghatározott protokollokon keresztül folyik az üzenetváltás. Egy hiba, vagy vizsgálat során meg kell határozni, hogy melyik hálózati szegmensek felelősek az adott funkció megvalósításáért, és az azok közötti interfészre kapcsolódva kell a forgalmat lementeni, és célszoftverrel dekódolni a későbbi részletes elemzéshez. A 2.1-es alfejezetben említett virtualizációs technológiák az 5G SA (Standalone) architektúrában kerülnek általános alkalmazásba. A jelenleg kereskedelmi forgalomban lévő 5G hálózatok NSA (Non-Standalone) architektúrára épülnek, melyek egy LTE EPC-re (Evolved Package Core) kapcsolt NR (New Radio)

hálózat, tehát a core hálózat még nem „új”. Teljes 5G SA hálózat még nagyon korlátozottan elérhető. Emiatt ebben a dokumentumban LTE technológián, az 5G NSA core-t is megvalósító „Evolved Packet Core” (EPC) hálózati adatain fogom vizsgálni a hálózati jelzésváltás forgalom becslését. Ez nem jelent hátrányt, mivel az LTE és 5G hálózatoknak sok közös tulajdonsága van, erre jó példa a fizikai réteg koegzisztenciája, vagy épp maga az 5G NSA-t támogató, LTE-vel fizikailag közös EPC. Ahogyan már korábban is említettem, a dolgozat szempontjából a legfontosabb tulajdonságuk, hogy mind az LTE, mind az 5G hálózatokban külön van választva és kezelve a vezérlő- és adatsík.

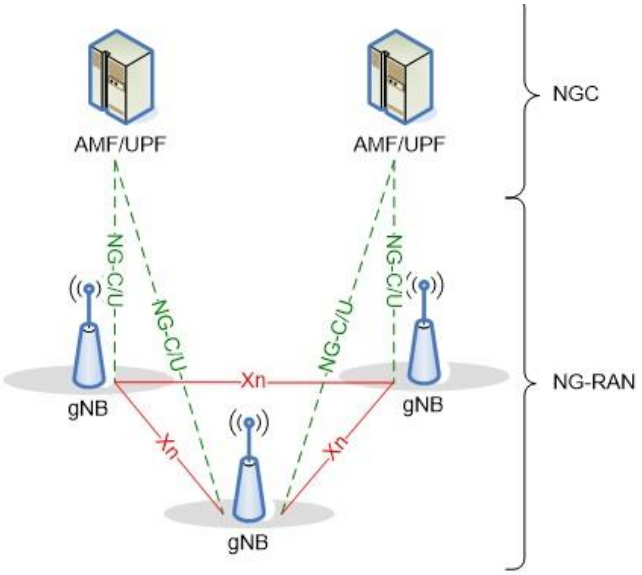
LTE-ben a rádióhálózat (RAN – Radio Network) és a core hálózat (CN – Core Network) vezérlősíkja között az S1-MME interfész teremt kapcsolatot, melyen S1AP protokollt alkalmaznak. Egészen pontosan az S1-MME az eNB-t (evolved NodeB) és az MME-t (Mobility and Management Entity) köti össze, ahogy az a 2.2. ábrán látható. Az LTE core hálózatban az MME kezeli a rádiós hozzáférés biztonságával és mobilitásával kapcsolatos feladatokat, valamint felelős a felhasználói készülékek nyomon követéséért (TAU – Tracking Area Update), és az készenléti állapotban (idle mode) lévő készülékek értesítéséért (paging). Ezen kívül még előfizetői információkat tartalmazó adatbázist (HSS – Home Subscriber Server) kezel.




**2.2. ábra: LTE hálózati architektúra.** Hiba! A hivatkozási forrás nem található.

Az 5G SA hálózatokban az AMF (Access and Mobility Management Funcion) hasonló az LTE MME-hez, ez a NAS (Non-Access Stratum) végpontja, feladata a hálózat integritásának védelme, hozzáférés engedélyezése, hitelesítés, mobilitás és kapcsolatok

kezelése. A 2.3. ábrán is látható, hogy a rádióhálózat és az AMF között az NG-C interfész teremt kapcsolatot.




**2.3. ábra: NR és 5GC közötti interfészek.** Hiba! A hivatkozási forrás nem található.

Ahogyan LTE-ben, úgy 5G-ben is vannak jól elkülönített hálózati funkciók, amelyek kapcsolódnak a vezérlő sík központi eleméhez és kiegészítik a működését, de ezek részletes bemutatása túlmutat a dolgozat témáján. Ezek miatt a hasonlóságok miatt vezethetnek eredményre a 4G hálózatokon történő fejlesztések az 5G hálózatok optimalizálása során. Jelen dolgozat témájának konkrét esetében a jelzésváltás üzenetek mennyiségére próbálok meg minél pontosabb becslést adni LTE EPC hálózatból gyűjtött múltbeli adatok alapján, mely az 5G NSA-t is kiszolgálja. Ez később kis munkával alkalmazható lesz 5G SA hálózatokhoz is, hiszen maga az elv hasonló, csak a gyűjtött adatokat kell máshogy strukturálni, hogy működjön a megírt kóddal. 5G hálózatok esetén még fontosabb is lehet a jelzésváltási forgalom volumenének a minél pontosabb előrejelzése, mert a nagyobb információtartalom miatt (pl. UE Capability) tipikusan nagyobb az egyes üzenetek terjedelme, mint LTE esetén.

A feladat során főként a hálózati forgalom Handover és Tracking Area Update üzeneteit fogom felhasználni. A Tracking Area a cellák egy csoportját jelöli. Erre azért van szükség, mert az *idle* állapotban lévő, akár mozgó felhasználói eszközök pozíciójának nem szükséges a cellaszintű ismerete, továbbá a tartózkodásuk ilyen pontos nyilvántartása nagy mennyiségű jelzésüzenettel járna, amely egyrészt a hálózatot terhelné feleslegesen, másrészt a felhasználói eszközök akkumulátorát is gyorsabban merítené.

Ehelyett Tracking Area-kat használnak, és csak azok váltásakor frissül a készülék pozíciója. Ha a hálózat fel akarja építeni a mobilkészülékkel a kapcsolatot, akkor az adott Tracking Area minden cellájába paging üzenetet küld a Paging csatornán, melyet a mobil megválaszol, és onnantól válik ismertté a cellaszintű helyzete. Ezen kívül van periodikus TAU is, amely bizonyos időközönként frissíti a mobileszköz pozícióját függetlenül attól, hogy történt TA váltás vagy sem. 5G esetén is a UE (User Equipment) akkor hajt végre RA (Registration Area) frissítést, ha elhagyja az aktuális RA-t és újba lép át, és itt nem is Tracking Area Update-nek nevezik, mint 4G-ben. Az 5G rendszerben a UE mikor elhagy egy cellát, és amelyikbe belép, az már egy másik RA-hoz tartozik, Registration Request üzenetet kell küldenie, melynek egy tartalmazott paramétere – Registration Type = Mobility Update – fogja jelezni, hogy másik cellacsoport cellájába lépett át. Idle módban lévő UE pedig Periodic Registration üzenetet küld az AMF által meghatározott időközönként.

## 4.3 Predikciós módszerek

Mobilhálózati forgalom előrejelzésére több jól használható módszer is van, de eredményességük nagy mértékben függ a feladat típusától. A jelen feladat céljára alkalmazható módszerek alapvetően két csoportra oszthatóak: idősoranalízis és gépi tanulás (ML – Machine Learning).

### 4.3.1 Idősoranalízis

Az idősoranalízis technikák magukban foglalják az auto regressziót (AR – Auto Regression), a mozgóátlagot (MA – Moving Average) és az autoregresszív mozgóátlagot (ARMA – Autoregressive Moving Average). Ezeket mind használják forgalom előrejelzésre, de általánosságban elmondható, hogy az MA modell nem a legjobb eredményeket adja idősoranalízis során, többnyire csak arra használják, hogy a zajt eltávolítsák az idősoros adatból. Az AR modell teljesítménye nagymértékben függ a megfigyelés időtartamától, nagyobb időablakokkal jobb eredményt lehet elérni. Az ARMA, ami az AR és MA kombinációja, széleskörűen használt mobilhálózati előrejelzések megvalósításához. Az idősoros analízis egy másik megközelítése a Markov modell, amely egy sorozat adatokat reprezentáló sztochasztikus modell. Egy Markov modellben egy idősoros adathalmaz minden adata egyedi állapotként van modellezve és az egyik állapotból a másikba való átmenet valószínűségét a múltbéli adatkészletből tanulja meg. Tehát a Markov modell nem csak hálózati előrejelzések előállításához

használható, hanem modellt is lehet vele alkotni. Ugyanakkor több kutatásból és cikkből is kiderül [15], hogy az idősoros elemzési modellek nem olyan pontosak összetett hálózati forgalmak esetén, mint a hálózati mintázatok elemzése. Ezért a mobilforgalom előrejelzés technikai trendje megváltozott, idősoranalízis helyett inkább gépi tanulási módszereket alkalmaznak erre a célra.

### 4.3.2 Machine Learning

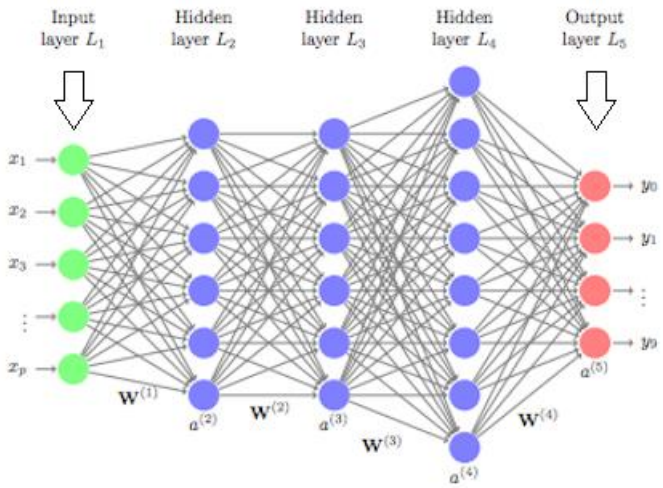
A mobilhálózatot működése és funkciói szempontjából három fő rétegre lehet bontani: fizikai, hálózati és alkalmazási réteg. Ezen a három szinten van lehetőség ML módszereket implementálni a mobilhálózatokba, ebben a dolgozatban a hálózati rétegbe tartozó funkcionalitással foglalkozom. Az ML technológiák közül a Deep Learning (DL) lehet a legalkalmasabb erre a feladatra, hiszen alkalmazásával nagyon jó Learning Rate (LR) érhető el valós minták alapján, valamint dinamikusan változtatható a feldolgozási lánc és a döntési hálózat. Mind osztályozási, mind regressziós problémák megoldására jól alkalmazható.

Az ML modelleknél, legyen is az bármelyik, kulcsfontosságú a tanító adathalmaz minősége és terjedelme, ami alapján betanítjuk a modellt. Ha nem tartalmaz megfelelő számú mintát az egyes osztályokhoz például, akkor a kimenet nem lesz az elvártaknak megfelelő. A modellek felépítése és működése nagyon sokféle lehet, a tanítás például felügyelt vagy nem felügyelt. Felügyelt tanítás esetén a tanító adathalmaz és egyes elemei címkézve vannak. Ez a típus jól használható döntési fák vagy KNN (K Nearest Neighbour) problémák megoldására. Döntési fákat (DT – Decision Tree) is lehet vele építeni. Előnyei, hogy valós számítási problémák megoldására jól használható, hiszen megfelelően címkézett adathalmazok alapján pontos becslést tud adni, a döntési kritérium utólagos változtatásával utólagos optimalizálásra is van lehetőség. Limitációi, hogy ismeretlen információ nem kérhető le az adathalmazból, mert nem tud önállóan új klasztereket és osztályokat felfedezni, továbbá egy bizonyos adatmennyiség felett csökken a pontossága a modellnek.

A nem felügyelt tanítás során a modell címkézetlen adathalmazokat használ a betanuláshoz, így rátanulhat rejtett adatmintázatokra és struktúrákra. Klaszterezésre alkalmas, ezen belül például K-means- és hierarchikus klaszterezés algoritmusok valósíthatók meg. Előnyei, hogy kevésbé komplex, mint a felügyelt tanítással tanított modell, nem kell felcímkézni az adatokat, és valós időben is elég jól tud működni.

Hátránya, hogy a címkézés hiánya miatt kevésbé pontos, és számítási szempontból is sokkal bonyolultabb, mint más ML technikák.

Egy DL modell (ld. 2.4. ábra) több rétegben tartalmaz neuronokat – amik matematikai műveletet végeznek el a bemenetükön megjelenő adaton, valamint súlyozva is vannak – a bemeneti mintákból további mintákat állít elő, melyek minden réteg után egyre komplexebbek lesznek. Előnyei, hogy több modell előállításához elegendő a már meglévő neurális háló rétegszámát megnövelni. Hátránya, hogy nagyon sok memória és számítási kapacitás szükséges hozzá, az optimalizálás komplex. Ezen felül nagy adathalmazra van szükség a tanításhoz, ami erősen befolyásolja a pontosságot. Az adatfeldolgozás során a láncszabály szerint a hálózat minden előrelépés (egy következő rejtett rétegbe) után módosítja a modellparamétereket, azaz a súlyokat (w) és a bias (a) értéket.




**2.4. ábra: Fully-connected DNN három rejtett réteggel. [16]**

Speciálisan idősoros adatok elemzésére és becslésére több gépi tanulásos módszert is kidolgoztak. Ilyenek például a felügyelt tanulás jellegűek: a Random Forest (RF), Support Vector Machine (SVM) és Deep Neural Network (DNN, Mély Neurális Háló) jól használhatóak mobilhálózati forgalom előrejelzésére [17].

# 5 Adatgyűjtés

## 5.1 Maghálózati monitorozás

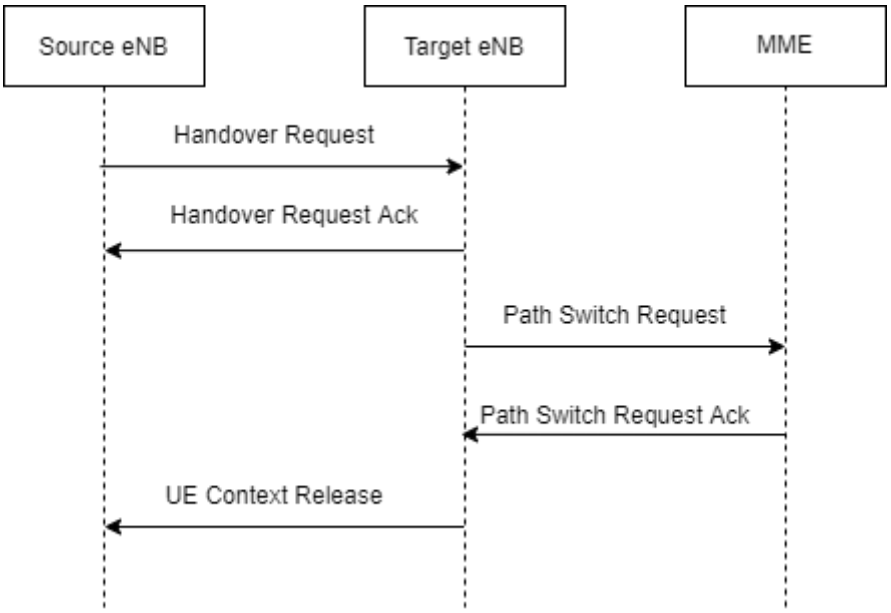
Az adatok az SGA hálózat-monitorozó rendszer [18] segítségével kerültek legyűjtésre egy LTE mobilhálózat S1-C interfészéről. Az SGA rendszer képes a jelzés-üzenetek bit-by-bit elkapására, lementésére, sorbarendezésére és dekódolására. Természetesen már itt is szűrve vannak az üzenetek, nem minden kerül lementésre, csak a Handover és TAU folyamatokhoz kapcsolódó kérés (Request) és válasz (Response) üzenetek, a felhasználói és egyéb érzékeny információk titkosításával. A szoftver a gyűjtött üzeneteket dekódolja és szöveg formátumban egy .txt kiterjesztésű fájlba menti. Az üzenetek azonosítói alapján kérés-válasz párokat lehet kialakítani, melyekből leválogatom a sikeres handover és TAU üzeneteket.

### 5.1.1 S1AP

Ebben az alfejezetben a dolgozatban felhasznált, S1AP interfészen haladó üzenetek és tartalmuk kerül bemutatásra. Nem célja a teljes, rádiós interfészen haladó forgalmak részletezése. Az jelzésváltási üzenetek kérés-válasz időbélyege alapján számolható az interfész válaszideje.

*Path Switch Request*

A 3.1. ábrán is látható handover esetén az eNB felveszi a cél cella azonosítóját a folyamathoz, ha a cél és a forrás cellát ugyan az az MME szolgálja ki. A Handover Request üzenet tartalmazza a UE információkat, melyek azonosítják az S1AP interfészen. A cél eNB kéri az S1-U GTP csatorna átkonfigurálását, mely üzenet alapján az MME az eNB s1AP ID-val azonosítja a UE-t. Ez az üzenet szintén tartalmazza az új cella és a TA azonosítóját. Az MME kérésére az SGW módosítja a csatorna útvonalát a cél eNB-hez, frissíti a vivőt (bearer) és küld egy választ az útvonal változtatás sikerességéről az MME-n keresztül. Fontos különbség, hogy Handover csak connected módban történhet, TAU viszont idle vagy connected módban is.




**3.1. ábra: Handover procedúra folyamatábrája.**

*Tracking Area Update*

Ha egy UE a mozgása során átlép egy új TA-ba, mindenképpen frissítenie kell a hálózati nyilvántartást az új TA azonosítójával. Ehhez hasonlóan a hálózat is megkérheti a UE-t, hogy frissítse a helyét periodikusan, mely segít a hálózatnak gyorsan megtalálni a UE-t mikor adatot kell neki küldenie. Ekkor egy broadcast csatornán kiküldött paging üzenettel keresi meg a legutóbb bejegyzett TA-ban. Ezen kívül a UE periodikus TAU üzeneteket is küld az MME-nek idle módban még akkor is, ha nem lép át új TA-ba. A TAU folyamatok is kérés-válasz üzenetpárokból épülnek fel, melyeket szintén össze lehet párosítani az egyedi azonosítók alapján.

## 5.2 Adat strukturálás

Az üzeneteket egy Python nyelven íródott script string karakterenként végigszkenneli és JSON fájlformátumba konvertálja. A JSON fájlból egy másik script kikeresi a szükséges információkat – például időbélyeg, üzenet típus, különböző cella és hálózati azonosítók, valamint az üzenet végpontjai – és ezeket egy .csv kiterjesztésű fájlba vesszővel és whitespace karakterekkel elválasztva táblázatos formátumba rendezi.

## 5.3 Az adatsorok bemutatása

A .csv fájlból az adatokat egy Python Pandas dataframe-be olvastam be úgy, hogy az egyes sorok azonosítója az adott üzenet időbélyegje lett. A teljes adathalmaz címkéi, valamint a dataframe felépítése a következő:

```
DatetimeIndex: 6450104 entries, 2022-05-09 08:00:00.298733 to 2022-
05-10 08:03:19.074527
Data columns (total 20 columns):
 #   Column             Dtype
---  ------             -----
 0   connection_id      int64
 1   IMSI               float64
 2   TMSI               object
 3   procedure_code     object
 4   DTAP_EMM_msg_type  object
 5   source_ip          object
 6   dest_ip            object
 7   end_date           object
 8   duration [s]       float64
 9   ECI                object
 10  TAC                object
 11  old_MME_S1AP_ID    object
 12  new_MME_S1AP_ID    object
 13  eNB_S1AP_ID        object
 14  old_EBI            object
 15  old_GTP_endpoint   object
 16  new_EBI            object
 17  new_GTP_endpoint   object
 18  result             object
 19  cause              object
     dtypes: float64(2), int64(1), object(17)
```

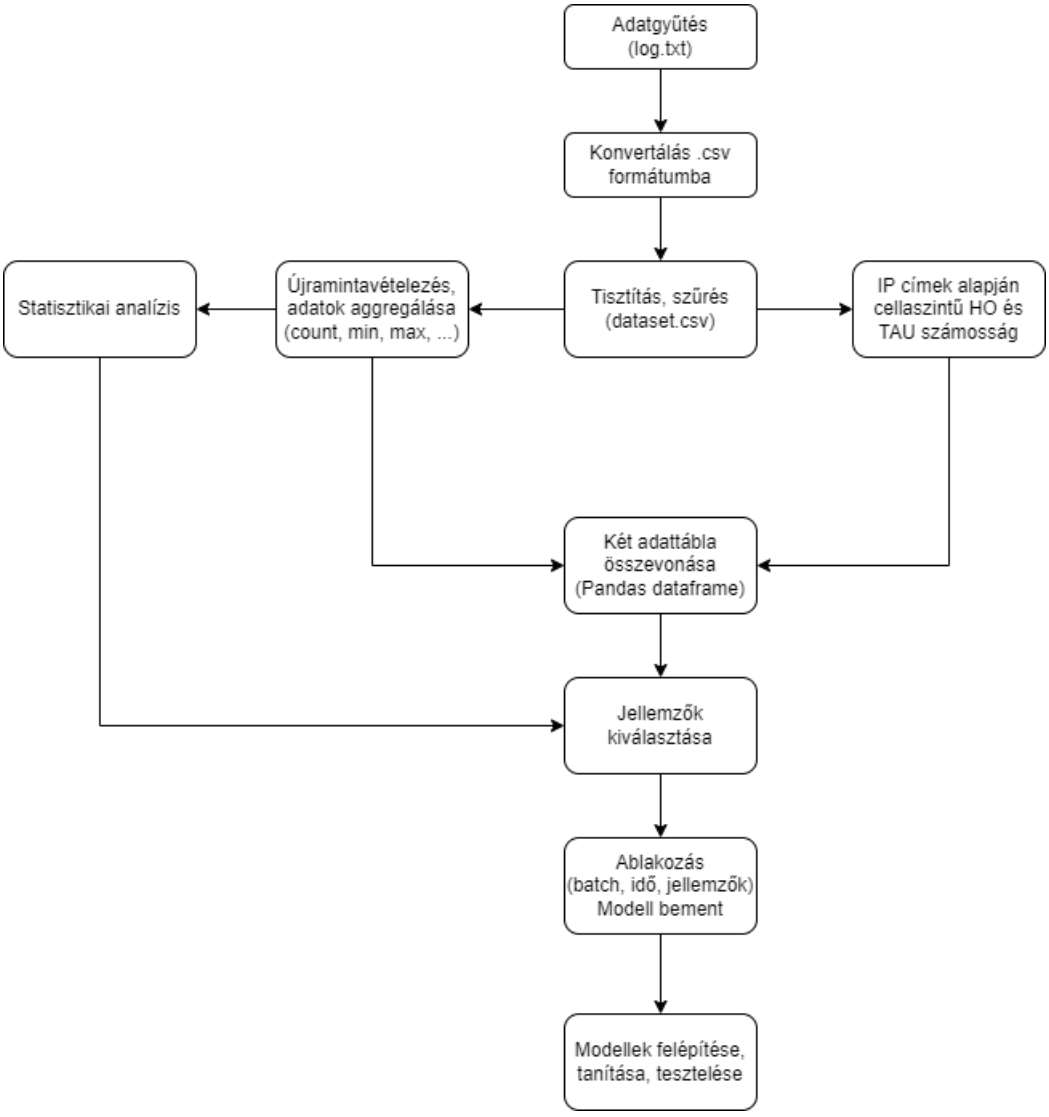
Láthatóak az egyes adatok típusai, a sok object típus azért van, mert azokban az értékekben valamilyen speciális karakter található, ami általában pont (pl. IP címnél) vagy csillag (azonosítók kitakarása). Így nehéz velük dolgozni, de amelyik adat szükséges, annak könnyen lehet egyedi ID-t generálni és az alapján azonosítani.

# 6 Tervezés

A feladat végrehajtásának lépései az alábbiak:

1. Adatgyűjtés
2. Adatok strukturálása, szűrése, egyszerűsítése
3. A tisztított adathalmaz statisztikai elemzése
4. Jellemzők (feature-ök) meghatározása
5. Egyszerűbb, majd komplexebb modellek felépítése
6. Neurális hálózat megvalósítása

A feladat végrehajtásának egyes lépéseit a 4.1. ábrán látható folyamatábra mutatja be az adatgyűjtéstől egészen a modellek tanításáig.




**4.1. ábra: A munkamenet folyamatábrája.**

## 6.1 Adatok elemzése, jellemzők kiválasztása

### 6.1.1 Adatok tisztítása, szűrése és leválogatása

Várható, hogy csúcsidőszakokban a jelzésforgalom nagymértékű megnövekedése nagyobb válaszidőt eredményez a CP-n. Mivel a TAU és HO (Handover) üzeneteket, vagyis a mobilitást az MME kezeli, így először csak az időbélyegekkel és az üzenetek mennyiségével foglalkoztam és elemeztem. Ehhez a Pandas DataFrame-ből egy másik változóba másoltam a *„duration [s]”* adatokat tartalmazó oszlopot, mely másodpercben tartalmazza az egyes TAU és HO üzenetek időtartamát (lefutási idejét). Az időbélyegeket nem kellett külön átmásolnom, hiszen azokat indexként állítottam be az adatsorokhoz. Ez után egy rövid statisztikát csináltam a *duration* értékekről kétféle konfidenciaszinten, mely alább látható a 4.1. táblázatban:

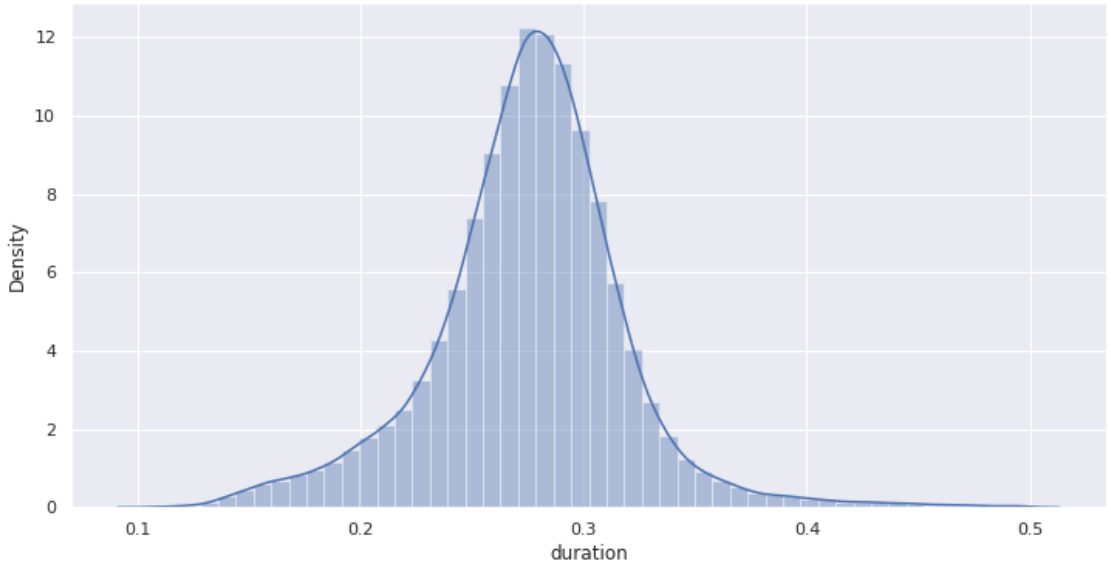
| | | | |
|---|---|---|---|
| count | 6.450104e+06 | count | 6.450104e+06 |
| mean | 2.843606e-01 | mean | 2.843606e-01 |
| std | 4.527810e-01 | std | 4.527810e-01 |
| min | 1.000000e-01 | min | 1.000000e-01 |
| 25% | 1.330000e-01 | 1% | 1.017200e-01 |
| 50% | 1.650500e-01 | 50% | 1.650500e-01 |
| 75% | 3.365000e-01 | 99% | 9.840000e-01 |
| max | 2.826543e+02 | max | 2.826543e+02 |

**4.1. táblázat: Az mobilitási procedúrák hosszának statisztikája.**

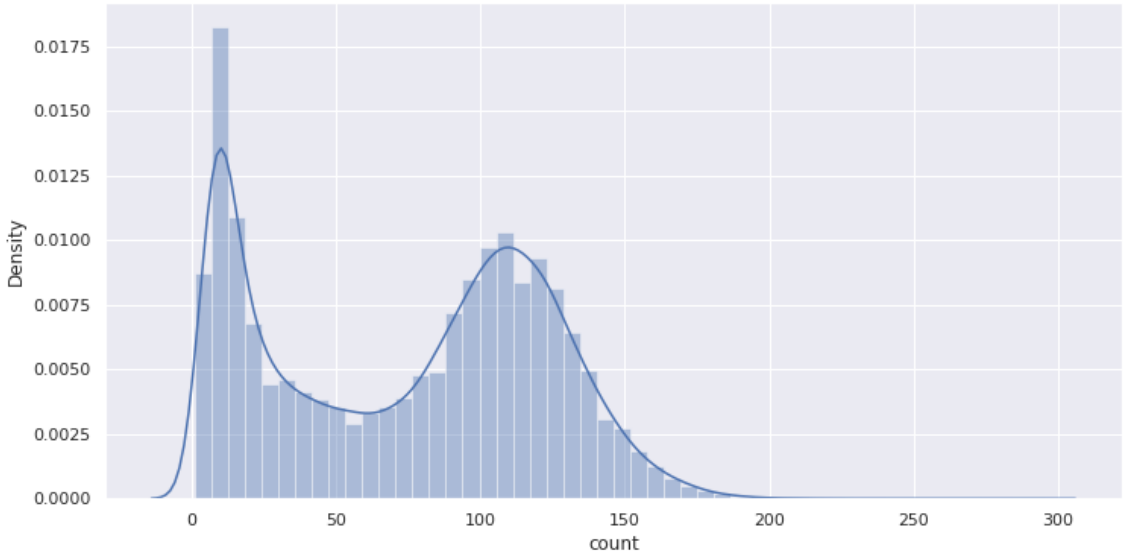
Látható, hogy „felfelé” elég nagy az adatok szórása, a nagyon kiugró adatokból viszont kevés van. Ezek az értékek annyira nagyok a többihez képest, hogy negatívan befolyásolhatják az elemzést, ezért ezeket ki kell venni az adatstruktúrából. Ez nem azt jelenti, hogy ezekkel nem kellene foglalkozni, csupán csak külön kell őket értékelni és esetleg valamiféle hibára gyanakodni. A statisztikából az is látszik, hogy arányaiban ezekből az adatokból nincsen sok. Mivel itt nagyon sok adatról van szó – több millió sor – ezért nem célszerű egyesével ábrázolni mindegyiket, mert nem lesz átlátható. A kiugró értékek mellet az adatok szórása is elég nagy. Az adathalmazból minden sort kitöröltem, ahol az S1AP interfész válaszideje nagyobb volt, mint 0.5 másodperc. Ez kevés adatsort jelent, az adatbázis méretével nem összemérhető a kitörölt sorok száma, de így nem torzítják el a statisztikát.

### 6.1.2 Statisztikai elemzés

A statisztikai elemzés és ábrázolás első lépéseként megnéztem a két vizsgált paraméter eloszlását, amire egy Gaussi kernel becslés görbéjét illesztettem rá. A Gaussi KDE (Gaussian Kernel Density Estimation) egy paraméter nélküli módszer egy adott valószínűségi változó sűrűségfüggvényének becslésére. A KDE egy alap adatsimítási probléma, ahol véges számú adatminta alapján vonunk le következtetéseket egy adatsokaságra. A 4.2. ábrán a *duration* hisztogrammja, míg a 4.3. ábrán az üzenetek másodpercenkénti darabszámának hisztogrammja látható. A megszűrt és kiátlagolt *duration* értékek normális eloszlást követnek.

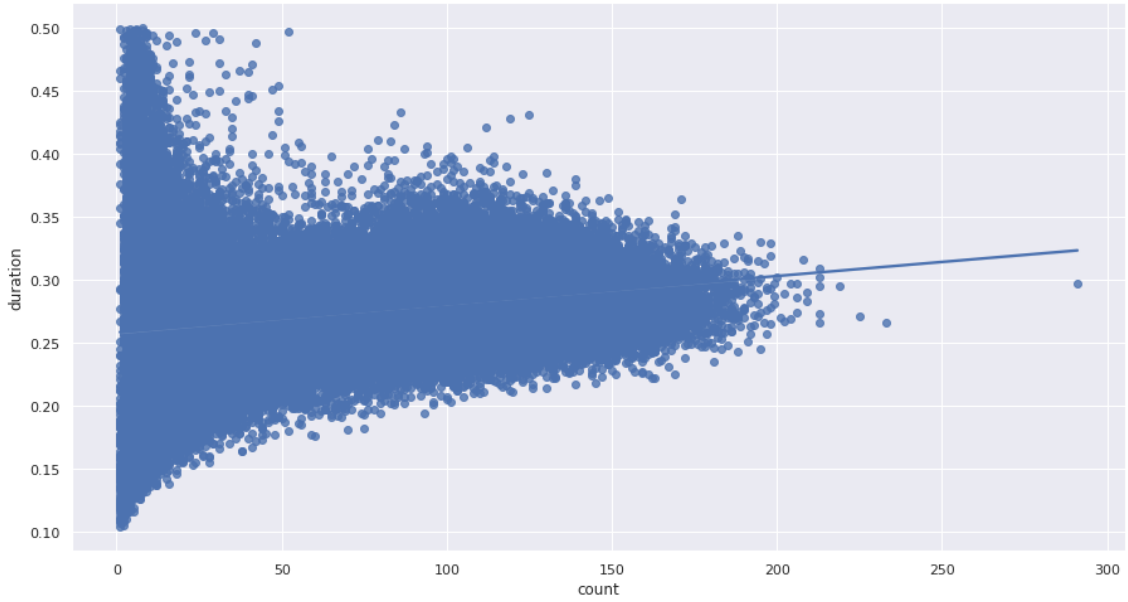



**4.2. ábra: A duration értékek hisztogrammja és Gaussian KDE görbéje.**




**4.3. ábra: Az üzenetek másodpercenkénti darabszámának eloszlása és Gaussian KDE görbéje.**

Annak a vizsgálatához, hogy növekszik-e számottevően a mobilhálózat S1AP interfészének válaszideje a nagyobb terhelés hatására, ahhoz a két adathalmaz egymáshoz való viszonyáról kell információt szerezni. Erre a célra a regresszió a megfelelő, ami az megmutatja, hogy milyen kapcsolat van a két adathalmaz között, ha van egyáltalán. A 4.4. ábrán látható a regressziós ábra, amely megmutatja milyen kapcsolat van a jelzésüzenetek száma és a válaszidő között.

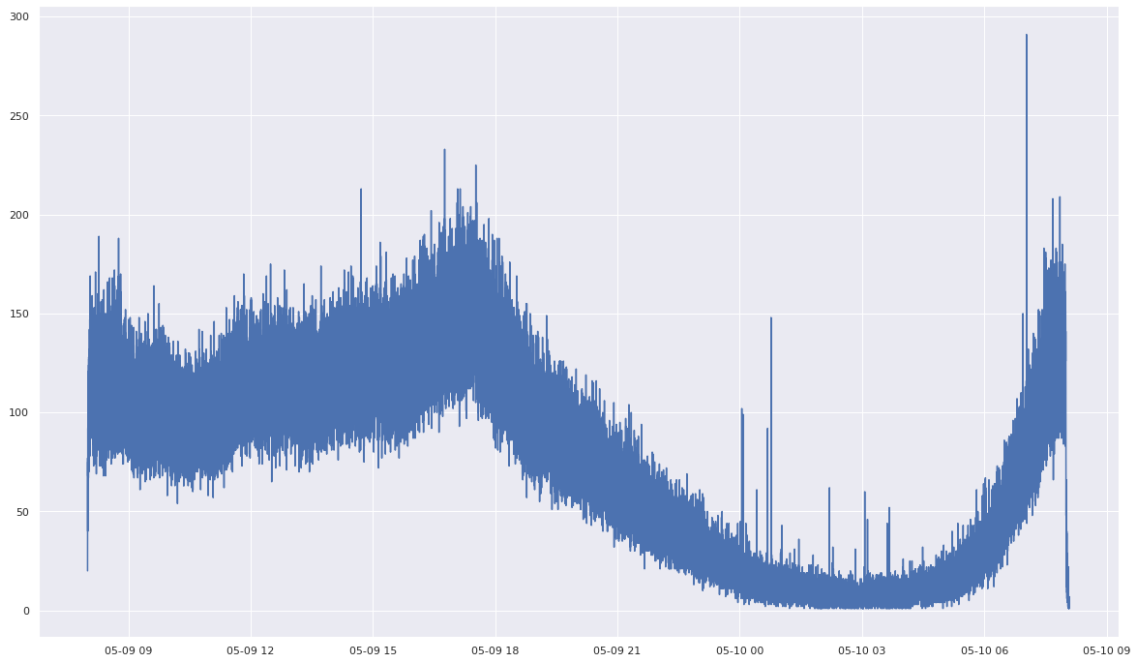



**4.4. ábra: Lineáris regresszió: a hálózati terhelés és válaszidő kapcsolata.**

A kirajzolódó pontfelhőnek nagyon nagy a szórása. Egyenest illesztve rá, a két változó között gyenge lineáris kapcsolat feltételezhető, de ez mégsem valószínű. A forgalmi volumen változásától nem függ nagy mértékben az válaszidő.

### 6.1.3 Újabb szűrőfeltételek kidolgozása

Láthatóan csak ezek az információk nem lesznek elegendőek, hogy forgalmi mintázatot lehessen alkotni az adatokból, szükség lesz a TA cella szintű felbontásra és adatelemzésre. Az egy napos adatok alapján meg lehet határozni olyan forgalmasabb időintervallumokat, amiket érdemes lehet vizsgálni. A 4.5. ábrán látható, hogy egy nap alatt hogyan változott a CP terhelése.




**4.5. ábra: Egy LA egész napos forgalmi terhelésének változása. A vizszintes tengelyen az idő, a függőleges tengelyen az üzenetek darabszámának normált mennyisége látható.**

Látható, hogy főleg a reggeli órákban és délután növekszik meg nagymértékben a jelzésváltási forgalom, értelemszerűen akkor, amikor az emberek munkába mennek-jönnek. Ezek lehetnek azok az időintervallumok, amiket érdemes elemezni. Különböző területen elhelyezkedő TA-k esetén ez az ábra teljesen máshogy nézhet ki, például egy Budapesti vagy egy agglomerációban lévő TA között biztosan nagy lenne a különbség az ingázó emberek miatt.

A fenti elemzés alapján a kiválasztott jellemzők a jelzésváltási üzenetek száma, TA azonosítók, cella azonosítók, és az előfizetői azonosítók egy külső adatbázisként történő felhasználása, hogy az egyes terminálok mozgását össze lehessen „kötni” a cellák között, és így valós felhasználói trendeket meghatározni.

## 6.2 Algoritmusok a viselkedésminták azonosítására

Viselkedésminták azonosítására, és azok alapján történő becslés előállítására többféle algoritmus alkalmas. Az egyes algoritmusok használata és a modellek felépítése előtt viszont meg kell határozni, hogyan fog történni a modell hatékonyságának és sikerességének a mérése, megállapítása. Mi az a pont, ahol már elfogadható lesz a ML modell, és indokolttá válik az alkalmazása? Ennek a megállapítására olyan jól működő

alapmodelleket is létre kell hozni, amikkel a bonyolultabb modellek eredményei összehasonlíthatóak lesznek. Ebben a feladatban két referenciamodell – baseline és lineáris – és egy ML modell – konvolúciós (CNN – Convolutional Neural Network) – létrehozása mellett döntöttem.

Az alapvető feladat az, hogy a múltbeli forgalmi értékek alapján a modellek egy becslést adjanak a jövőbeli forgalom volumenéről. A forgalmi adatokat tartalmazó adatbázis sorai az előfeldolgozás során negyedórás intervallumonként lettek aggregálva, mely egy viszonylag nagy időintervallum. Ebből következik, hogy a volumen változása nem lesz annyira gyors és drasztikus, mintha kisebb – például perces – időátfogással lenne előállítva a bementi adattábla. Így kiinduló referenciamodellnek jó lehet egy baseline modell, ami becslésként mindig az idősorban eggyel korábbi értéket adja. Ez a modell gyakorlatilag egy időintervallummal „lemaradva” követi a forgalmi változásokat a hálózatban. Értelemszerűen ez nem lesz egy pontos becslés, de ha a ML modellek ennél is rosszabbul teljesítenek, akkor azt jelenti, hogy nagy hiba van bennük. A lineáris modell a legegyszerűbb tanítható modell, amit idősoranalízisre lehet használni. Szimplán egy lineáris transzformációt végez el a bemeneti adatokon és a kimenet csak ettől függ. Ha nagyon kicsi időintervallumra – milliszekundumra – a szeretnénk előre jelezni, akkor a lineáris regresszió vagy extrapoláció szinte kivétel nélkül jó becsléseket fog adni, ezeknél jobbat nem nagyon lehet elérni. Nagyobb időintervallumokra viszont nem feltétlen a legpontosabb, de jó referenciapontot biztosít a bonyolultabb modellek sikerességének értékeléséhez.

A konvolúciós neurális hálókat strukturált adathalmazok feldolgozására alkalmazzák, melyek állhatnak szövegből, képekből vagy számszerű adatokból. A konvolúciós hálózatok architektúrája egy többrétegű feed-forward neurális hálózat, melyben több rejtett réteg épül egymásra sorban. Ez a szekvenciális felépítés teszi lehetővé a konvolúciós neurális hálózatok számára, hogy „megtanulják” az adatsorok hierarchikus jellemzőit. A rejtett rétegek tipikusan konvolúciós rétegek, melyeket tipikusan aktivációs és esetleg pooling rétegek követnek.

## 6.3 A gépi tanuló keretrendszer

Egy gépi tanuló keretrendszer több, egymástól lényegesen eltérő funkciójú rétegből áll össze, melyek mindegyike egy előre meghatározott transzformációt végez el az adathalmazon. CNN esetén ezek az elsődleges rétegek a következők:

1. Konvolúciós réteg
2. Pooling réteg
3. Fully connected réteg

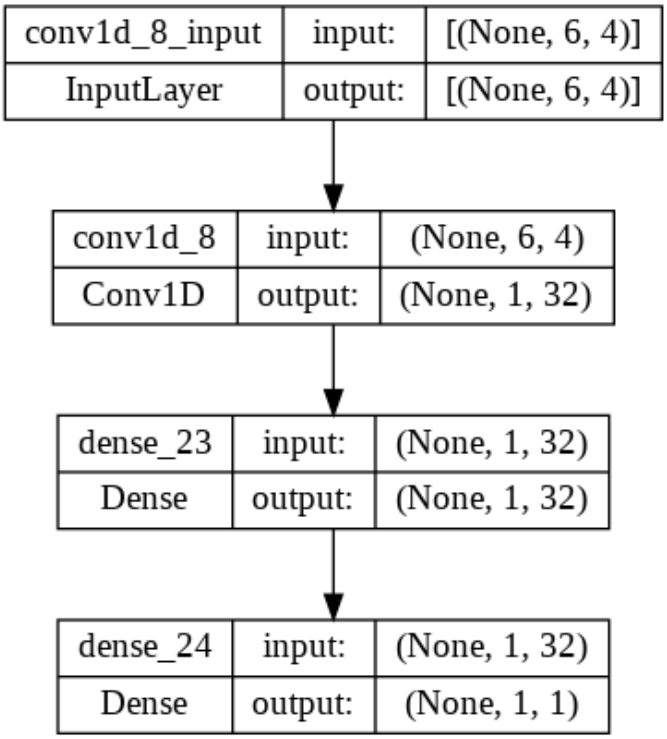
A konvolúciós réteg egy szorzást hajt végre a bemeneti adatokon, amellyel kiemeli az adatsor fontos jellemzőit. Mivel a bemeneten egy idősoros egydimenziós adathalmaz lesz, így 1D konvolúciót kell használni. A pooling rétegben a konvolúció utáni eredménymátrix vonódik össze és egy új mátrix képződik. Az összevonás történhet a mátrix maximum vagy átlag értékei alapján is, aminek a mértéke egy állítható paraméter. A fully connected réteg az a része a modellnek, amely valójában „tanulni" fog ahogy átfuttatja az adatokat a neurális hálózaton. Idősoros előrejelzésnél 1D-s tömb a bemeneti adathalmaz, ezt kell több dimenziós mátrixszá alakítani. A célmátrix dimenziója attól függ, hogy hány neuron van a fully connected réteg bemenetén, valamint hány korábbi adat alapján szeretnénk a következő értéket megbecsülni.

Maga a programkód a TensorFlow nyílt forrású platformra lett alapozva, ami gépi tanulásra fejlesztett eszközöket, könyvtárakat és közösségi forrásokat foglal magában. Működése során az adatokat tenzorok formájában kezeli, mely egy matematikai objektum, amik a programozási analógiában többdimenziós tömböknek felelnek meg. A TensorFlow segítségével lehet neurális hálózati modelleket építeni és tanítani. A TensorFlow-ra épül a Keras, ami egy magasabb szintű API, segítségével könnyebben lehet kezelni a TensorFlow funkcióit.

### 6.3.1 A kiinduló konvolúciós neurális hálózat bemutatása

A kiinduló konvolúciós neurális háló egy kevés rétegből álló modell, ahol a bemeneti réteget egy konvolúciós és két dense réteg követi. A modell felépítése a 4.6. ábrán látható. A konvolúciós réteg a megadott ablakméretekkel „letapogatják" a bemenetre kapott adathalmazt és azokból újabb mintákat állítanak elő. A rejtett rétegek neuronjai a bemenetükön keresztül megtanulják a lehetséges absztrakt reprezentációkat, amelyek jellemzően csökkentik az adathalmaz dimenzióját. Ez nagyon hasznos, mivel a gépi tanulás végrehajtásához szükséges számítási mennyiség általánosan közelítőleg exponenciálisan növekedne a bemeneti adathalmaz dimenziójának egységes növekedéséhez képest. A hálózat azt feltételezi, hogy ezek az absztrakt reprezentációk egymástól függetlenek, a bemeneti jellemzők viszont nem. Jelen példában 6-os ablakmérettel, egy címkével és 1-es csúsztatással dolgoztam. Aktivációs függvénynek

ReLu-t választottam, így elkerülhető a gradiens elhajlás, mely például a sigmoid függvény alkalmazása mellett előfordulhat. Veszteségfüggvényként átlagos négyzetes hibát (MSE – Mean Squared Error) használok, optimalizáló algoritmusként pedig Adam algoritmust.

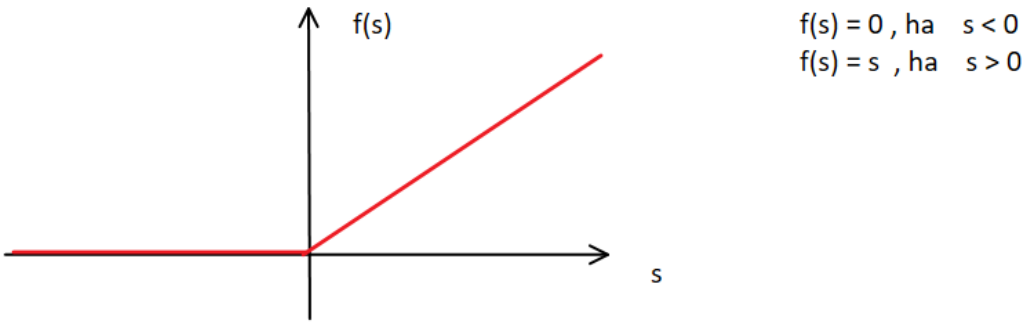



**4.6. ábra: A kiinduló konvolúciós modell felépítése.**

### 6.3.2 Rectified Linear függvény

Egy betanított CNN olyan rejtett rétegekkel rendelkezik, amelyek neuronjai a bemeneti jellemzők egy lehetséges, egymástól független absztrakt reprezentációinak felelnek meg. Amikor egy új, ismeretlen bementi érték érkezik a neuronháló még „nem tudja”, hogy az általa ismert absztrakt reprezentációk közül melyik lesz releváns az adott bement szempontjából. A rejtett réteg minden neuronjára, ami egy adott betanult absztrakt reprezentációt jelent, két eset lehetséges: az adott neuron releváns vagy sem. Ha egy neuron nem releváns az még nem jelenti azt, hogy ennek következtében más absztrakt reprezentációk is kevésbé valószínűek. A 4.7. ábrán látható Rectified Linear függvény pont ezt a funkciót valósítja meg, és azt a neuront, ami ezt használja ReLU-nak (Rectified Linear Unit) nevezik. Ennek a függvénynek két fő előnye van a szigmoid függvényekkel szemben: kiszámítása egyszerű, mert a bement és a 0 érték összehasonlítását kell csak elvégeznie, valamint 0 és 1 deriváltja is van attól függően, hogy a bement negatív vagy

sem. Ez utóbbi különösen fontos a backpropagation-höz tanítás során. Egy neuron gradiensének a kiszámolása nem jár nagy költséggel.




**4.7. ábra: Rectified Linear függvény.**

A nemlineáris aktivációs függvények – mint például a szigmoid – nem rendelkeznek ezzel a tulajdonsággal. Tehát a ReLU használata segít megakadályozni a neurális hálózat működéséhez szükséges számítási igények exponenciális növekedését. Ha a CNN-t növeljük, akkor a számítási költségek az egyes ReLU-k hozzáadásával „csak" lineárisan nőnek. Továbbá megakadályozza, hogy egy neuron gradiense a nullához közelítsen a bementi magas értékek hatására. Míg a szigmoid függvények deriváltjai pozitív végtelenhez közeledve 0-hoz tartanak, a ReLU mindig 1-en marad. ez lehetővé teszi a backpropagation-t és a tanulás folytatását még az aktivációs függvény magas bemeneti értékeinek esetén is.

### 6.3.3 Adam optimalizáló algoritmus

Az Adam algoritmus a hálózati súlyok iteratív frissítésére szolgál a tanító adatok alapján, számításilag hatékony és kis memóriaigényű. A sztochasztikus gradiens süllyedés egyetlen tanulási sebességet tart fenn minden súlyfrissítésnél az egész tanulás során. Ezzel szemben az Adam minden hálózati súlyhoz külön-külön igazítja a tanulási sebességet. A módszer a gradiensek első és második pillanatának becslései alapján kiszámítja az egyéni adaptív tanulási sebességeket a különböző paraméterekhez, az átlag mellett felhasználja a varianciát is. Az algoritmus kiszámolja a gradiens és a négyzetes gradiens exponenciális átlagát, melyek csillapítási sebességét két paramétere – beta1 és beta2 – szabályozzák.

# 7 Implementáció

A feldolgozott adatbázis adatok egy TA 4 napi – 2 hétfő és 2 szombat – CP forgalmát tartalmazzák. A megtisztított és formázott adatbázist tovább bontottam, hogy részletesebb képet adjon a felhasználók mozgásáról, egészen cellaszinten. Az S1AP interfészen az MME és az eNB között haladó mobility üzenetek egy része a UE, és a kiszolgáló cella azonosítóját is tartalmazza, így meg lehet határozni a felhasználók mozgását a cellaváltások alapján. Egy Request üzenetben a forrás IP cím mindig beazonosítja a kiszolgáló cellát, a cél IP viszont az MME egyik interfészének címe. A cél cellát úgy a legkönnyebb meghatározni, ha megnézzük a felhasználó következő Path Switch Request vagy TAU üzenetének forrás IP címét. Ez a forrás IP cím az előző sikeres cellaváltás cél cellájának eNB IP címe. Így előáll egy adatbázis, mely a felhasználók cellák közötti mozgását tartalmazza időrendben. Ezeket az adatokat bizonyos időablakok szerint aggregálva egy átfogó képet kapunk a cellák, valamint a TA-kénti CP terheltségről. Az 5.1. ábrán látható egy adattábla részlet, mely két cellapárt tartalmaz, melyek szomszédosak. A handoverek száma 15 percenként van benne összegezve, minden 15 percben új sort vesz fel az algoritmus a táblába.

| | Source_IP_1-Dest_IP_1 | Source_IP_2-Dest_IP_2 |
|---|---|---|
| 0 | 111 | 124 |
| 1 | 131 | 189 |
| 2 | 103 | 141 |
| 3 | 98 | 115 |
| 4 | 84 | 108 |
| 5 | 85 | 119 |
| 6 | 63 | 81 |

**5.1. ábra: Szomszédos cellák közötti handoverek száma.**

Hogy jól kezelhetőek legyenek az adatok és a modell tulajdonságai is tesztelhetőek, ellenőrizhetőek legyenek, az adatbázist néhány olyan céllára szűrtem, amelyek egy hosszabb forgalmas, ám sűrűbben lakott területet fednek le. Így lesz olyan felhasználói csoport, amely végighalad a főúton, és sok handovert generál az egymással szomszédos cellákon sorban, és lesznek, akik kevésbé „szabályos” mozgást követnek.

Az így előállított, cellapárokra bontott adattábla utolsó oszlopa után további oszlopokat szúrtam be, melyek a kérés-válasz üzenetpárok időtartamára vonatkozó információkat tartalmaznak. Itt nem alkalmaztam cellapáronkénti bontást, mert a vizsgált

TA-kat egy MME szolgálja ki, így az interfész válaszidejére vonatkozó információk az összes cella forgalmának tudatában értékelhetőek. Fontos, hogy ezek az időtartamok, csak az S1AP interfész válaszidői, a RAN hálózat késleltetése ehhez nem adódik hozzá. Ezek az oszlopok a következők:

- *count*: a cellapárok értékei soronként összegezve (ez a Path Switch és TAU üzenetek számának a 15 perces összege)
- *mean*: az üzenetpárok időtartamának 15 perces átlaga
- *min*: az üzenetpárok időtartamának 15 perces minimuma
- *max*: az üzenetpárok időtartamának 15 perces maximuma
- *std*: az üzenetpárok időtartamának 15 percre vett szórása

## 7.1 Adatok előkészítése a modellezéshez

### 7.1.1 Általános megfontolások

Mivel az összes adat egy táblába van, ezeket szét kell választani, hogy legyen tanító, validációs és teszt adathalmaz, ami alapján a modell pontosságát és egyéb jellemzőit meg lehet határozni. Idősoros adatoknál sajnos nem célszerű keverten kiemelni a validációra és tesztelésre szánt adatokat, mert akkor előfordulhatna, hogy nem egymást követő minták kerülnének az ablakokba. Ha a feldolgozási időablakokba közvetlen egymás utáni adatok kerülnek az biztosítja, hogy a modell betanítása után gyűjtött adatok alapján a validációs és teszteredmények kiértékelése valósághűbb lesz. Az adatokat 70-20-10 %-ban osztom fel tanító-, validációs-, és teszt halmazokra.

A neurális hálózat tanítása előtt nagyon fontos a jellemzők méretezése, ehhez normálom az adatokat, mely során minden jellemzőből kivonom az átlagot és elosztom a szórás értékével. Csak a tanító adathalmaz átlagát és szórását használom fel normálásra, így a modell nem fér hozzá a tanulás során a teszt adathalmazból származtatott értékekhez.

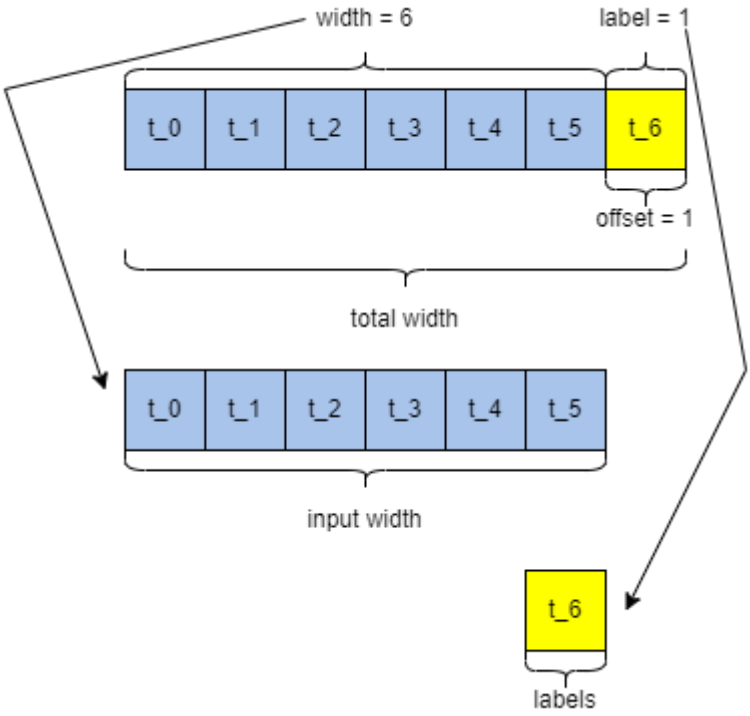
### 7.1.2 Ablakozás

Az ablakozás azért szükséges, mert ezzel lehet felparaméterezni, hogy a modell hány múltbeli értéket használjon fel a következő érték megbecsüléséhez, milyen címkéket használjon, és milyen lépésközöket alkalmazzon. Az ablakozást megvalósító

függvénynek négy bemeneti paramétere van, melyek egész számok, és az időbélyegek számára vonatkoznak:

- *width*: a bemeneti ablak szélessége
- *label*: a címke szélessége
- *offset*: a bementi ablak és címke közötti távolság
- *shift*: ablak csúsztatásának mértéke

Az ablakozással az 5.2. ábrán látható módon jellemző-címke párokra oszlik szét az adathalmaz, amelyekből már hatékonyan lehet a tanításhoz szükséges batch-eket kialakítani. Ezeket az egymást követő bemeneti ablakokat szétválasztottam egy *input* és egy *label* listába. Ezekből a listákból több dimenziós TensorFlow tömböket alakítok ki, melyek legkülső indexe a batch-eket, középső indexe az időbélyegeket, és legbelső indexe a jellemzőket azonosítja.

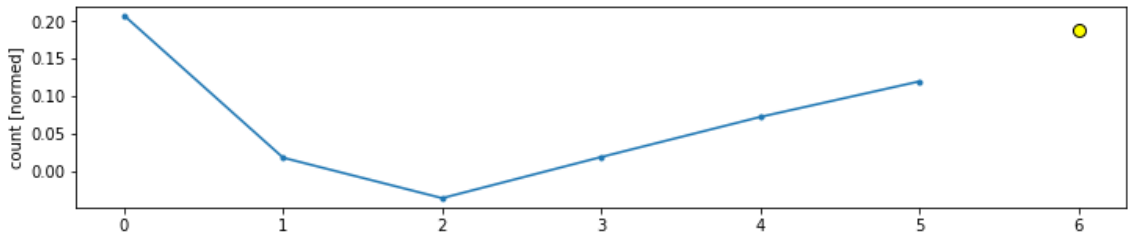



**5.2. ábra: Ablakozás és címkézés.**

Az ablakparaméterek beállítása után a *.shape()* beépített függvény segítségével lekérdezhető az adattáblák alakja, ami jelen esetben a következő:

```
All shapes are: (batch, time, features)
Window shape: (3, 6, 4)
Inputs shape: (3, 5, 4)
Labels shape: (3, 1, 1)
```

Ez azt jelenti, hogy a kód 3 darab 6 időlépéses batch-et csinált, minden lépésben 8 jellemzővel. Ezeket szétbontotta 3 darab 5 időlépéses batch-re, lépésenként 8 jellemzővel, valamint 3 darab 1 időlépéses batch-re, lépésenként egy jellemzővel. Az egy lépéses, egy jellemzős batch-ek a címkék, az értékük pedig a Path Switch és TAU üzenetek darabszáma, amelyre becslést várunk a modelltől. Az 5.3. ábrán látható grafikusan egy batch tartalma. Az adattábla többi oszlopa kirajzolva hasonló ábrákat eredményezne, azzal a különbséggel, hogy azok nem tartalmaznának címkéket.




**5.3. ábra: Egy batch tartalma. A vízszintes tengely a lépések számát, a függőleges tengely pedig az üzenetek számának normált értékét mutatja.**

Végül a batch-ekből egy tensorflow adatkészletet csináltam. Ehhez a két sorból két adattáblát kell csinálni, melyekből aztán a Keras *timeseries_dataset_from_array()* függvénye input-címke párokból álló TensorFlow adatkészletet fog csinálni. Az adatkészleten végzett iteráció eredményeképpen az alábbi adatstruktúra áll elő:

```
Inputs shape (batch, time, features): (32, 5, 4)
Labels shape (batch, time, features): (32, 1, 1)
```

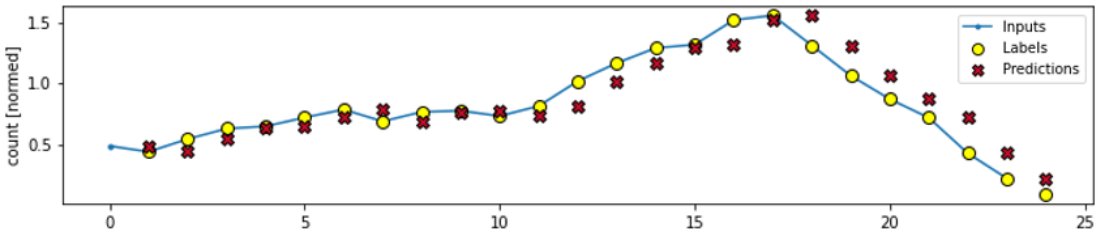
## 7.2 Predikciós modellek és vizsgálati eredmények

A következő modellek, melyeket készítettem, mind egylépéses modellek, mely azt jelenti, hogy a bementére adott adatok alapján készít egy becslést egy időlépéssel a jövőbe. Az alkalmazott modellek működése és felépítése jelentősen különbözik egymástól, de a fontosabb paramétereik – például a meneti adatok dimenziója – megfeleltethetőek egymásnak. Ez azt jelenti, hogy az ablakozást és az ablak paramétereket minden modellnél fel lehet használni. A fentiekben részletezett szűrt és

strukturált TensorFlow adatkészletből kiindulva először az ablakok paramétereit határoztam meg, melyek az alábbiak: *width = 6; label = 1; offset = 1; shift = 1*. Minden modellnél 24 időlépésben (24*15 perc = 6 óra) ábrázolom a működését. Minden ábrán kék színnel van jelölve a bemeneti adatfolyam, sárga pöttyel a címkék, és piros x-szel a modell által adott becslések.

### 7.2.1 Baseline

Idősor analízis során a baseline modell egy jó összehasonlítási alap a későbbi bonyolult modelleknek. Ha egy modell a baseline alapmodellnél gyengébb teljesítményt ér el, akkor azt el kell hagyni, nem lesz jó. A jelenlegi feladathoz egy egyszerű baseline modell is elég, mert az adatok 15 perces időintervallumokban vannak aggregálva, így nem jellemzi őket nagyon gyors, hirtelen ingadozás. Elegendő egy olyan baseline modell, ami a következő időlépés (t+1) becslőjének az adott (t) időlépés értékét adja. Ez a gyakorlatban annak felel meg, mintha a címkéket egy időlépéssel előretoltuk volna. Az 5.4. ábrán látható a baseline modell működése.



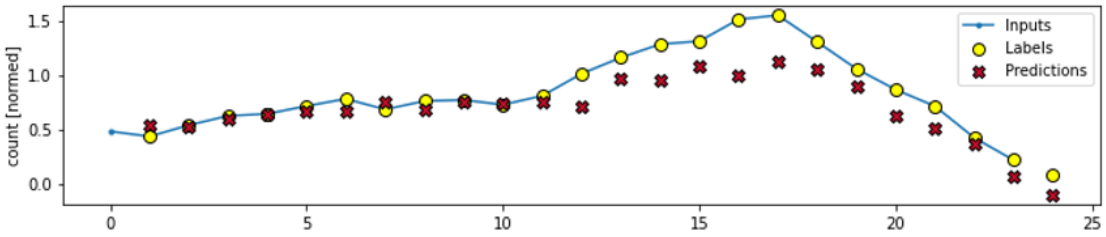

**5.4. ábra: Baseline modell becslései. A vízszintes tengely a lépések számát, a függőleges tengely pedig az üzenetek számának normált értékét mutatja.**

A baseline modell trendjében megfelelően becsli a bekövetkező értékeket, bár az eltérések a nagyobb változások környékén megnőnek, és ezek előfordulása után (pl. 1 18. lépéstől) több lépés után sem látható, hogy jelentősen csökkenne a becslés hibája.

### 7.2.2 Lineáris

A legegyszerűbb tanítható modell, ami alkalmas idősoros adatok becslésére, egy lineáris transzformációt elvégző modell, melynek a kimenete ettől az egy transzformációtól fog függeni. Ebből adódik egy nagy előnye, hogy viszonylag egyszerűen értelmezhető. A gyakorlatban egy Keras.Dense réteg, aktivációs funkció nélkül egy lineáris modell. Ez a réteg a <batch, time, inputs> adatokból csak az utolsó

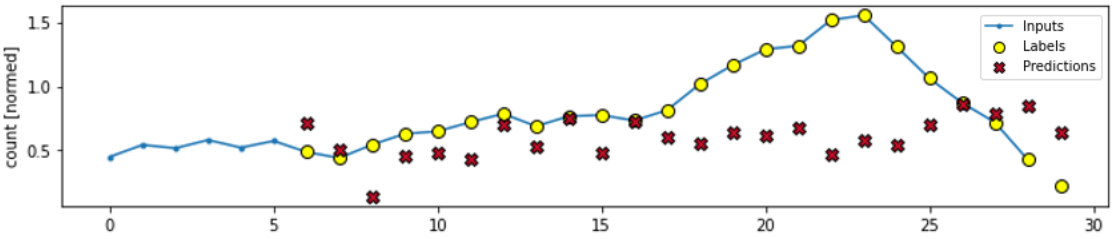
elemet fogja módosítani egymástól függetlenül minden batchben és időlépésben. Az 5.5. ábrán látható a lineáris modell egy jellemző lefutása. A piros x-szel jelölt becsléseket figyelve észrevehető, hogy a becslések több esetben is jobbak, mintha csak a bemeneti értéket adná vissza, de bizonyos esetekben van, hogy gyengébb egy-egy érték. A modell véletlenszerűen inicializálja a bemenetére érkező adathalmazt, ezért néha előfordulhat, hogy nem az általunk fontos jellemzőre helyezi a legnagyobb súlyokat és gyenge teljesítményt mutat.




**5.5. ábra: Lineáris modell becslései.**

### 7.2.3 Az egyszerű konvolúciós modell

Az előző két modell minden időlépést egymástól függetlenül kezelve állította elő az azt követő időlépés becslőjét. Ezzel szemben a konvolúciós modell konvolúciós rétege több időlépést használ fel bemenetként minden becslés előállításához. A kimenet rövidebb lesz, mint a bemenet, így az összehasonlítható ábrázoláshoz több időlépést kell kirajzolni. Ezért az 5.6. ábrán a 6-os bemeneti ablakmérethez megfelelően 6-tal több időlépés van felvéve, így összehasonlítható a kiinduló konvolúciós modell a baseline és lineáris modellel. Már az ábráról is látszik, hogy ez az egyszerű konvolúciós modell milyen pontatlan becsléseket ad. Ezért változtatni kell a modellen.

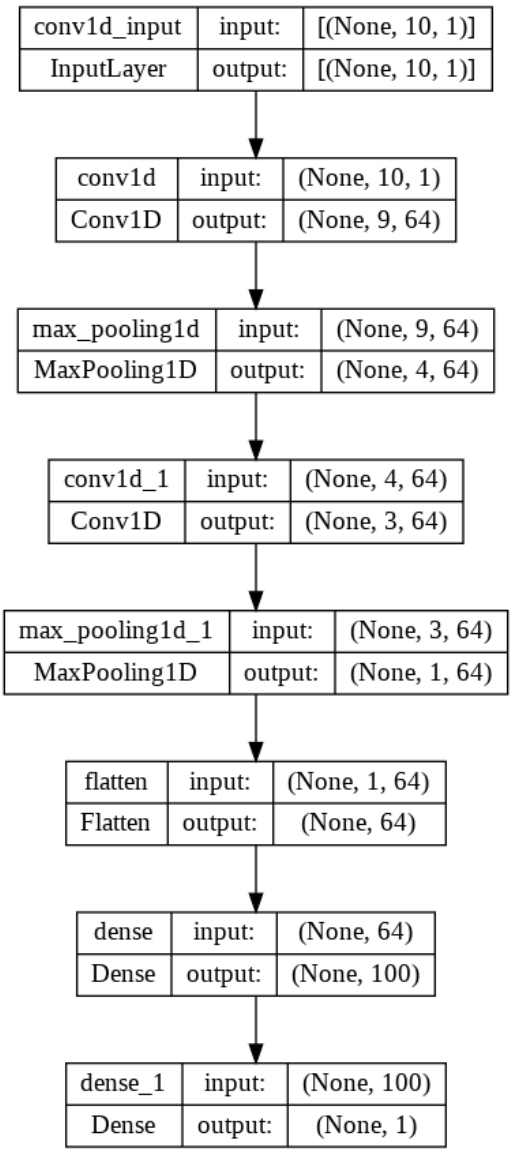



**5.6. ábra: Az egyszerű CNN modell becslései.**

### 7.2.4 Az összetettebb konvolúciós modell

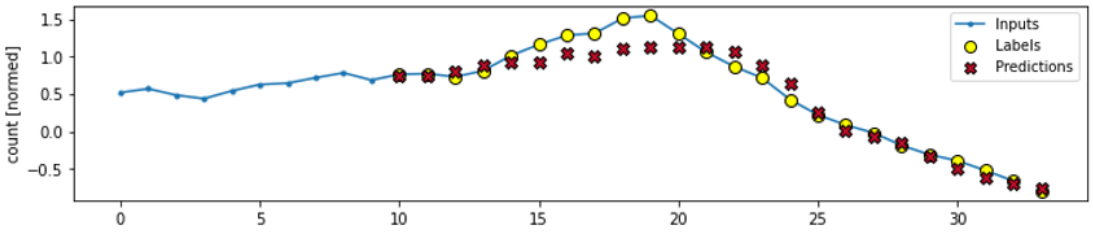
A változtatás történhet plusz rétegek hozzáadásával vagy a modellparaméterek állításával. Mivel ez egy kevés rétegből álló kiinduló modell, ezért további rétegek

hozzáadásával kezdtem. Beillesztettem még egy kovolúciós réteget, majd mindkét konvolúciós réteg után 1-1 MaxPooling réteget. A MaxPooling réteg alulmintavételezi a bemenetére érkező adathalmazt, úgy, hogy kiemeli a maximális értékeket minden poolból. A pool ablak mérete a függvény argumentumában állítható, az ablak folyamatosan tolódik a bemenetén érkező adatokon. Ez után 32-ről 64-re növeltem a szűrők számát a konvolúciós rétegekben, ennek hatására a bemeneti adathalmazból több minta készül. A két dense réteg megmaradt, de a rejtett réteg neuronjainak számát 100-ra növeltem. Ezen felül a bemeneti ablakméretet 10-re változtattam, így a több bemeneti minta alapján pontosabb becslést tudott előállítani a modell. A modell felépítése az 5.7. ábrán látható.




**5.7. ábra: Konvolúciós modell felépítése.**

A tanítás után, az előzőekhez hasonló ábrára kirajzoltam a modell bemenetére értkező értékeket és a becsléseket. A különbség annyi, hogy a megnövelt ablakméret miatt a baseline modellhez képest itt tízzel több időlépést ábrázoltam, hogy ugyan annyi becslés érték látszódjon. Észrevehető, hogy ez az összetettebb konvolúciós modell sokkal jobb, mint az alapmodell, és vélhetően a lineáris és baseline modellnél is jobban teljesít. Ha egy gyorsabb változás miatt kileng az egyik irányba, kevés időlépés alatt korrigál és megfeleően követi a trendet. A bemeneti és becsült értékek az 5.8. ábrán láthatóak.

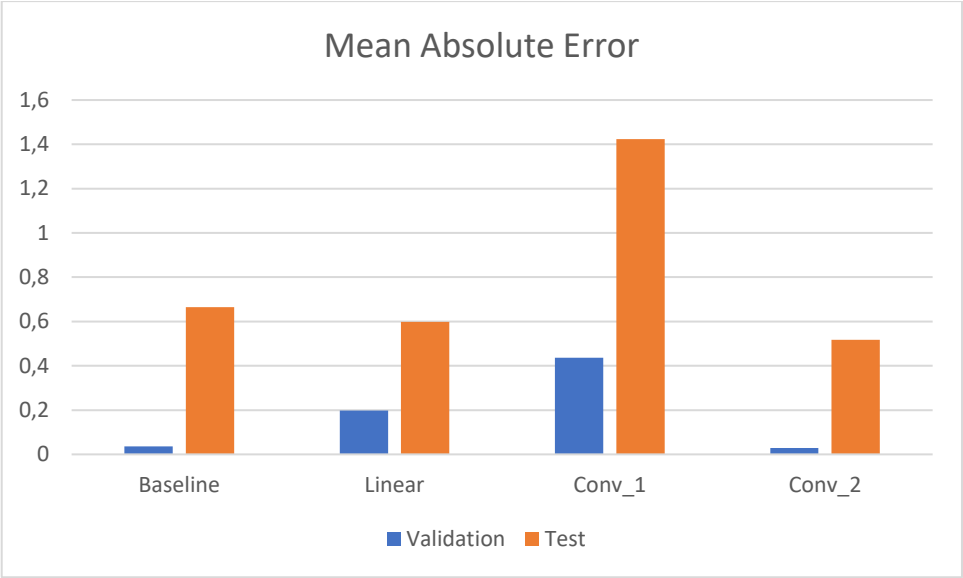



**5.8. ábra: Az összetettebb konvolúciós modell becslései.**

Ez egy nagyon fontos és látványos eredmény. Azt mutatja, hogy az összetettebb, többrétegű konvolúciós neurális hálózattal sokkal pontosabb becslés érhető el, mint bármely más, vizsgált modellel. Ezt a tulajdonságát még magas lépésszámoknál is megtartja; a nagyobb hibákból is képes visszatérni precíz becsléci állapotba.

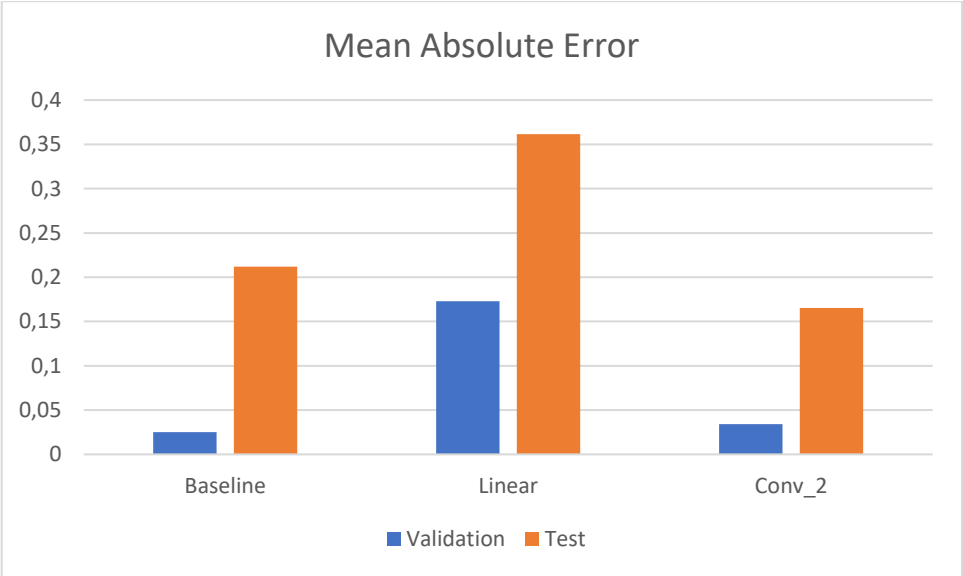
## 7.3 A predikciós modellek pontosságának összehasonlítása

A vizsgált modellek abszolút hibaértékét oszlopdiagramon ábrázoltam. Az 5.9. ábrán jól láthatóak a modellek teljesítményei közötti különbségek. Az első konvolúciós hálózat teljesítménye nagyon gyenge a többi modellhez képes, de az csak egy kiinduló modell volt. A bonyolultabb, pontosított konvolúciós modell viszont felülteljesítette a baseline és a lineáris modellt is. Ez azért is különösen figyelemreméltó eredmény, mert a lassabban változó idősoros adathalmazon (mint amilyenek a vizsgált jelzéshálózati üzenetmennyiségek), a lineáris modellek nagyon jól tudnak teljesíteni.




**5.9. ábra: A modellek teljesítménye 15 perces aggregálással.**

Az összetett konvolúciós modell azzal lehet jobb, hogy a MaxPooling rétegeknek köszönhetően jobban le tudja követni a nagyobb mértékű eltéréseket is. Ha csak 1 percenként aggregálom az adatokat, sokkal nagyobb eltérések lesznek a szomszédos időlépések értékei között. Ennek eredményeképpen még nagyobb különbség mutatkozik az alap és összetettebb modellek között. Az 1 perces adattáblára lefutatott predikciós vizsgálatok középhibáinak eredményei az 5.10. ábrán láthatóak.




**5.10. ábra: A modellek teljesítménye 1 perces aggregálással.**

A kapott eredmények alapján kijelenthető, hogy a felhasználók mobilitás- és session-management jelzés-forgalma összetett konvolúciós hálózatokkal pontosabban becsülhető különböző időtávokra, mint más, a gyakorlatban felmerülő modellekkel.

# 8 Összefoglalás

Ebben a dolgozatban a mobil maghálózat erőforrás-allokációjának tökéletesítését segítő forgalmi becslő módszereket mutatok be. Ezek a módszerek mobilitás- és session-management jelzésforgalom minél pontosabb becslését célozzák, sok perces időablakokra. A bemutatott eredmények fontos jellemzője, hogy az alkalmazott módszerek valós, élő hálózaton gyűjtött, üzeneteket (és nem folyam-statisztikákat) dolgoznak fel. Fontos eredmény, hogy az itt bemutatott, egyedi, összetett konvolúciós hálózat pontosabb predikciót szolgáltat, mint a baseline vagy a lineáris becslés, és sokkal jobbat, mint egy egyszerű konvolúciós hálózat.

Munkám fontos része volt, hogy a mobil maghálózatból gyűjtött adatokból egy használható adatbázist készítsek, ami a mobilhálózati működésre jellemző összefüggéseket is tartalmaz. Ennek az adattáblának a felhasználásával tanítottam és optimalizáltam egy gépi tanuló modellt, aminek a teljesítménye jobb lett, mint az idősor analízisre általában használt lineáris modellé. Az általam készített konvolóciós neurális hálózati modellel lehetőség van a hálózati jelzésforgalom előre-becslésére akár gyorsan változó terhelés esetén is.

A bemutatott eredmények egyik tanulsága, hogy gyakorlati, élő adatokon is bizonyítja: az ML technikák alkalmazása telekommunikációs hálózatokban sok előnnyel járhat, a predikciós módszerek hatékonyan működnek. Ennek folyománya, hogy az erőforrásmenedzsment optimalizálásához és a cellák minél ideálisabb Tracking Area-kra osztásához lényeges információk nyerhetők ki a hálózatból és dolgozhatóak fel mély neuron hálózatokkal. Konkrét példaként egy konvolúciós neuronhálót tanítottam, hogy minél pontosabban előrebecsülje a jelzésváltási forgalom várható értékét egy adott területen. Ezt alapul véve a kiszolgáló virtuális gépek (NFV) erőforrás-allokációja dinamikusan változtatható.

A mostani, általam kialakított adattábla cella szinten és összesítve is tartalmazta a HO és TAU üzeneteket, ami alapján a szomszédos cellák közötti mozgásokat össze lehet kapcsolni. Ahhoz, hogy cellák közötti átmeneteket is tudjunk becsülni, a bemutatott összetett konvolúciós hálózat nagyon jó kiindulást jelent – de a vizsgálatokhoz több hónapnyi jelzésforgalom feldolgozására van szükség. Így nem csak a napi, de a heti

szezonalitások (munkanapok mobilitása a hétvégékkel és ünnepnapokkal szemben) is taníthatók.

További előrelépést jelenthet, ha az érzékeny adatok titkosítása helyett anonimizációt használunk. Így az előfizetői azonosítók metaadatként való tárolásával a mozgásokat előfizetőkhöz lehetne kapcsolni. Ezzel megtartható az adatok közötti korreláció. Ekkor egy osztályozási feladatot ellátó ML modellel lehetségessé válik a felhasználó profilozására, és a most elkészített modell jelzéshálózat becslése szerint helyileg meg lehet határozni a várható CP és UP hálózati igényeket.

# 9 Köszönetnyilvánítás

Köszönetemet szeretném kifejezni témavezetőmnek, Dr. Varga Pálnak útmutatásaiért és javaslataiért. Hálás vagyok ipari konzulensem, Tánczos László szakértői tanácsaiért, melyek nagy segítséget nyújtottak számomra a munkám során.

# Irodalomjegyzék


[1] S. Rezaei, X. Liu, Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, *57*(5), 2019., 76-81.

[2] Á. Gabilondo, Z. Fernández, R. Viola, Á. Martín, M. Zorrilla, P. Angueira, J. Montalbán. Traffic Classification for Network Slicing in Mobile Networks. *Electronics*. 2022; 11(7):1097. https://doi.org/10.3390/electronics11071097

[3] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning. In *2018 Network traffic measurement and analysis conference (TMA)*, IEEE, 2018.

[4] X. Wang, S. Chen and J. Su, "App-Net: A Hybrid Neural Network for Encrypted Mobile Traffic Classification," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 424-429,

[5] B. Gyires-Tóth, P. Varga, T. Tóthfalusi. Utilizing Deep Learning for Mobile Telecommunications Network Management. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) 2019.*, pp. 575-580. IEEE.

[6] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin. Improving traffic forecasting for 5G core network scalability: A machine learning approach. *IEEE Network*, *32*(6), 2018. pp. 42-49.

[7] J. Jeong *et al*., "Mobility Prediction for 5G Core Networks," in *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 56-61, March 2021, doi: 10.1109/MCOMSTD.001.2000046.

[8] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, T. Ahmed. Virtual network functions orchestration in wireless networks. In *2015 11th International conference on network and service management (CNSM)*, 2015, pp. 108-116. IEEE.

[9] OIF/ONF Whitepaper 2017.02.10, „SDN Transport API Interoperability Demonstration”

[10] Network Function Virtualisation White Paper, October 15-17, 2013 at the “SDN and OpenFlow World Congress”, Frankfurt-Germany

[11] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young. Mobile edge computing—A key technology towards 5G. *ETSI white paper*, *11*(11), 2015. pp.1-16.

[12] 5G Network Slicing Whitepaper, Viavi, 2019

[13] LTE Definited through LTE Network Diagrams (rcrwireless.com), 2022.05.19

[14] 5G | ShareTechnote, 2022.05.19

[15] M. Joshi and T: H: Hadi, ”A rewiev of network Traffic analysis and prediction techniques”, 2015

[16] The Shortest Introduction To Deep Learning You Will Find On The Web | by Christoph Ostertag | Analytics Vidhya | Medium, 2022.05.19

[17] C. Zhang and P.Patras, ”Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks”, 2017

[18] D. Kozma, G. Soos, P. Varga. Supporting LTE network and service management through session data record analysis. *Infocommunications Journal*, *71*(2), 2016. pp. 11-16.