



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Mélytanuló algoritmusok alkalmazása Bridge Weigh-in-Motion rendszerekben

TDK DOLGOZAT

Készítette
Szinyéri Bence
Csató Erik

Konzulens
dr. Kővári Bence

2023. november 2.

Tartalomjegyzék

Kivonat	i
Abstract	ii
1. Bevezetés	1
1.1. Motiváció	1
1.2. Elért eredmények	1
1.3. A dolgozat felépítése	2
2. Monitoring rendszer	3
3. Irodalom áttekintése	7
3.1. Szimuláció	7
3.2. BWIM-algoritmusok	9
3.2.1. Ablakdetektálás	10
3.2.2. Sebességbecslés	11
3.2.3. Tengelydetektálás	12
3.2.4. Tengelysúlybecslés	14
3.3. Adathalmaz	16
3.3.1. BME-Simulated-I	16
3.3.2. Vasbeton szerkezetű hidak esetén alkalmas adathalmaz	18
4. Saját megközelítés	20
4.1. Konvolúciós neurális hálózat alapú sebességbecslés	21
4.1.1. Motiváció	22
4.1.2. Architektúra	22
4.1.3. Tanítás	23
4.2. Konvolúciós neurális hálózat alapú tengelydetektálás	23
4.2.1. Motiváció	23
4.2.2. Architektúra	24
4.2.3. Tanítás	24
5. Elért eredmények	27
5.1. Sebességbecslés	27
5.1.1. Lokális maximum alapú sebességbecslés	27
5.1.2. Eredmények összehasonlítása	29
5.2. Tengelydetektálás	33
5.2.1. Lokális maximum alapú tengelydetektálás korlátai	33
5.2.2. Eredmények összehasonlítása	33
6. Kitekintés, jövőbeli tervek	36

6.1. BWIM-rendszer	36
6.2. Ablakdetektálás	37
6.3. Jövőbeli tervek	38
7. Konklúzió	39
Köszönetnyilvánítás	40
Irodalomjegyzék	41

Kivonat

A hidak életciklusa nyomon követésének egyik pillére a járműforgalom nyomon követése. A híd állapotának nyomon követése szükségessé teszi a járműforgalom monitorozását. A szakértők és az üzemeltetők a hídon áthaladó járművek tengelyterheléseinek és összsúlyának ismeretében olyan modelleket képesek létrehozni, amelyek a hídszerkezet fáradását előre tudják jelezni. Ezért az áthaladó forgalom nyomon követése fontos feladat az építőmérnökök számára. Az ilyen funkcionalitással rendelkező rendszereket Bridge Weigh-In-Motion (BWIM) rendszereknek nevezik.

Négy fő lépése van ezeknek az adatvezérelt BWIM rendszereknek. Az első lépés a járművek időablakainak kinyerése. A jármű ablaka azt az időintervallumot határozza meg, amikor egy jármű áthaladt a szenzorok keresztmetszetén. A második lépés általában a jármű sebességének kinyerése. A jármű sebességét a legtöbb rendszerben állandónak feltételezik, ha a járművek sebessége magas (high-speed weigh-in-motion, HS-WIM). A harmadik lépés azon időbélyegek azonosítása, amikor a jármű tengelyei a szenzorok felett áthaladtak. A jármű sebességének és tengelyeinek időbélyegeinek ismeretében kiszámíthatóak a tengelytávolságok. A tengelytávolságok és a sebesség ismeretében negyedik lépésként a tengelyterhelések megbecsülhetők. Ez a dolgozat a járműsebesség kinyerésére és a tengelyek detektálására összpontosít.

Egy újszerű, mélytanuláson alapuló sebességbecslőt mutatunk be a dolgozatban. A fejlesztés egyik előnye, hogy képes egy keresztmetszetben elhelyezett szenzorok jelei alapján is megbecsülni a sebességet, miközben a hiba alacsony marad. Ez a módszer egy mélytanulási alapú megoldás. A módszerünk és a szakirodalomban is használt lokális szélsőértékek detektálásán alapuló módszerek összehasonlítása is bemutatásra kerül. Megállapítható, hogy ez az újszerű, kevesebb nyúlásmérő bélyeget használó megközelítés alkalmazható a BWIM rendszerekben.

Egy újszerű, mélytanulási alapú tengelydetektáló algoritmust is bemutatunk a dolgozatban. A tengelydetektálás során a legnagyobb problémát az egymáshoz közeli tengelyek, úgynevezett tandem és tridem tengelycsoportok detektálása jelenti. A tandemek és tridemek a tehergépkocsikon megfigyelhető, szorosan egymás mellett elhelyezkedő tengelyek. A fő hozzájárulásunk az, hogy ez az újszerű módszer képes megtalálni a tandemek tengelyeit további szenzorok vagy kamerák használata nélkül, miközben a megoldás hordozható marad.

Az új módszereket egy nagy méretű szimulált szintetikus adathalmazon és valós címkézett adatokon is kiértékeljük, és az általunk fejlesztett rendszerről is egy átfogó képet adunk.

Abstract

One aspect of monitoring the life cycle of bridges is to monitor vehicular traffic. The demand of monitoring bridge performance has led to measuring traffic flow. Experts and operators can create models predicting fatigue damage by knowing the axle loads and total load of crossing vehicles. This is why monitoring traffic flow is an important task for civil engineers. Systems with this functionality are called Bridge Weigh-In-Motion (BWIM) systems.

These data-driven BWIM systems have four major steps when estimating axle loads. The first step is extracting vehicle time windows. A vehicle time window defines the time interval when a car was crossing the section of sensors. The second step is usually extracting vehicle speed. Vehicle speed is presumed to be constant in most high-speed weigh-in-motion (HS-WIM) systems. The third step is identifying time stamps when axles of the vehicle were crossing above the sensors. Knowing the vehicle speed and timestamps of crossing axles, axle spacings can be calculated. In the light of axle spacings and speed, axle loads can be estimated as a fourth step. The current paper focuses on vehicle speed extraction and axle detection.

A novel deep learning-based speed extractor is presented in this paper. One of the main contributions of our development is that it uses sensors in only one cross-section of a girder while the error is kept low. This method is a deep learning-based solution. Comparison of this novel method and previously introduced peak-based methods is also presented. It can be stated that this novel approach using less strain gauges can be applied in BWIM systems.

A novel deep learning-based axle detector is also presented in this paper. The biggest issue is to solve the problem of detecting axles of trucks with lighter loads in tandems and tridem. Tandems and tridem are closely ranged axles that can be observed on trucks. Our main contribution is that this novel method can find the axles of tandems without the usage of additional sensors or cameras.

New methods are evaluated on a large simulated synthetic dataset and on real annotated scenarios as well, and we give a brief summary of the developed BWIM system.

1. fejezet

Bevezetés

1.1. Motiváció

A hidak életciklusa és fáradása előrejelzésének egyik pillére a járműforgalom nyomon követése. A híd állapotának nyomon követése szükségessé teszi a járműforgalom monitorozását. A szakértők és az üzemeltetők a hídon áthaladó járművek tengelyterheléseinek és össz súlyának ismeretében olyan modelleket képesek létrehozni, amelyek a hídszerkezet fáradását előre tudják jelezni [1]. A híd fáradásának nyomon követése pedig lehetővé teszi, hogy a híd biztonságos maradhasson. Ezért az áthaladó forgalom nyomon követése fontos feladat az építőmérnökök számára. Az ilyen funkcionalitással rendelkező rendszereket Bridge Weigh-in-Motion (BWIM) rendszereknek nevezik.

Munkánk során egy ilyen tengelysúlymérő rendszer fejlesztésén dolgoztunk, amelynek gyakorlati alkalmazására a komáromi Monostori hídon áthaladó forgalom monitorozása során is lehetőség volt. A feladat megoldásához rendelkezésre álló adatsorok a híd alján található, a hidat hosszirányban tartó trapézbordákra szerelt nyúlásmérő szenzorokból származnak. A rendszer fő előnye, hogy nem invazív. Ez azt jelenti, hogy ellentétben egyes Weigh-in-Motion (WIM) [2] rendszerekkel, a szenzorok nem az úttest felületén, abba beépítve vannak, ezért a rendszer az úttest felbontása és a forgalom korlátozása nélkül is telepíthető már meglévő hidakra, ezáltal széleskörű felhasználást lehetővé téve. Ez komoly versenyelőnyt jelent. Másfelől nézve viszont ezek a rendszerek komplexebb számítások révén tudják csak az elvárt kimeneteket (tengelysúly, össz súly, tengelytávolságok) előállítani.

Ezeknek a BWIM-rendszereknek a fő célja a tengelysúlyok kiszámítása. Ehhez azonban szükség van bizonyos előfeldolgozó (preprocess) lépésekre a tengelyek pozícióinak és a járművek sebességének kiszámítására. Ezeket a jellemzőket általában egy-egy különálló modul határozza meg.

Az elmúlt években ezen a területen is elterjedtek deep learning alapú megoldások [3, 4, 5, 6]. A jelen dolgozatban olyan újszerű sebességbecslő és tengelydetektáló algoritmusokat mutatunk be, amik valamilyen szempontból felveszik a versenyt az eddigi irodalomban talált megoldásokkal.

1.2. Elért eredmények

Két újszerű konvolúciós neurális háló alapú sebességbecslő és tengelydetektáló modult sikerült készítenünk, amelyeket a BME-S1 adathalmaz mellett valós mérési adatokon is sikeresen alkalmaztunk.

A sebességbecslő modulunk kevesebb szenzor telepítése mellett is képes átlagosan 1.34 km/h hibával, az esetek 95%-ban pedig maximum 5 km/h hibával kiszámítani a sebességet egy szintetikus adathalmazon. Egy kis méretű valós mérési eredményeken alapuló

adathalmazon is 0.90 km/h átlagos hibával sikerült a sebességet becsülni, ami átlagosan 1,79% hibát jelent. A megoldás jó alternatívát adhat kevesebb szenzoron alapuló BWIM-rendszerek létrehozásához.

A tengelydetektáló neurális háló a sávonkénti összegjelek, és a sávok külső szélén lévő szenzorok összegjelei alapján végzi a tengelyek azonosítását. Ez a módszer alkalmazható vasbeton hidak esetében is. A szintetikus adathalmazon végzett kísérletek megmutatták, hogy 99.57%-os Recall és 99.08%-os F1-score értékek mellett sikerült azonosítanunk tengelyeket zajmentes adatokon, a zajos adatokon ezen értékek rendre 96.59% és 96.87% voltak. A felsorolt eredmények szemléltetik, hogy a neurális háló alapú megközelítések messze felülmúlják a lokális maximum alapú megoldásokban rejlő potenciált. A megoldásunkat a komáromi hídon mért valós adatsorokon is teszteltük.

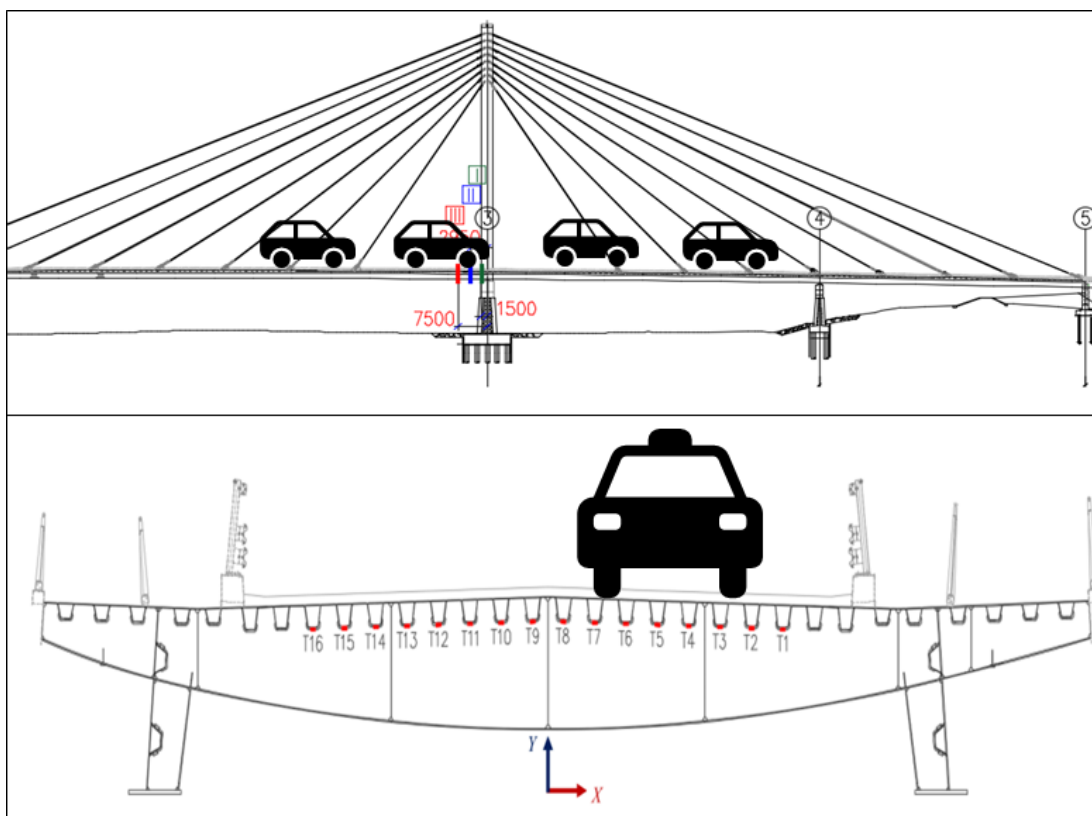
1.3. A dolgozat felépítése

A 2. fejezet részletesen bemutatja a komáromi Monostori hídon kialakított monitoring rendszert, és az általa szolgáltatott adatsorokat. A 3. fejezet az irodalomban fellelhető megoldásokat és adathalmazokat mutatja be. A 4. fejezet az általunk fejlesztett megoldásokat írja le. A fejezeten belül a 4.1 fejezet egy újszerű sebességbecslő algoritmust, míg a 4.2 fejezet egy újszerű tengelydetektáló algoritmust mutat be. Az 5. fejezet az elért eredményeket mutatja be, összehasonlítva az irodalomban található eredményekkel. A 6. fejezet egyfajta kitekintést ad, hogy a tengelysúlymérő rendszerbe hogyan illeszkednek a korábban bemutatott megoldások, valamint jövőbeli fejlesztési irányokat is megfogalmaz. A 7. fejezet összefoglalja az elért eredményeket, és megfogalmazza kutatásunk konklúzióit.

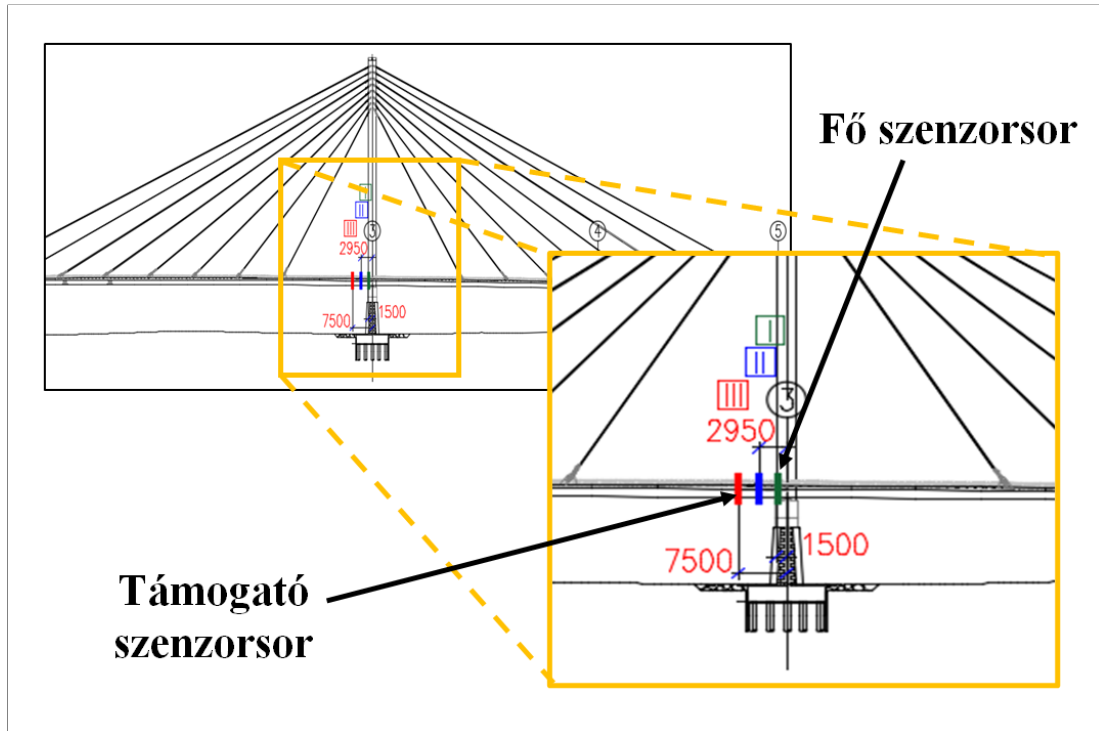
2. fejezet

Monitoring rendszer

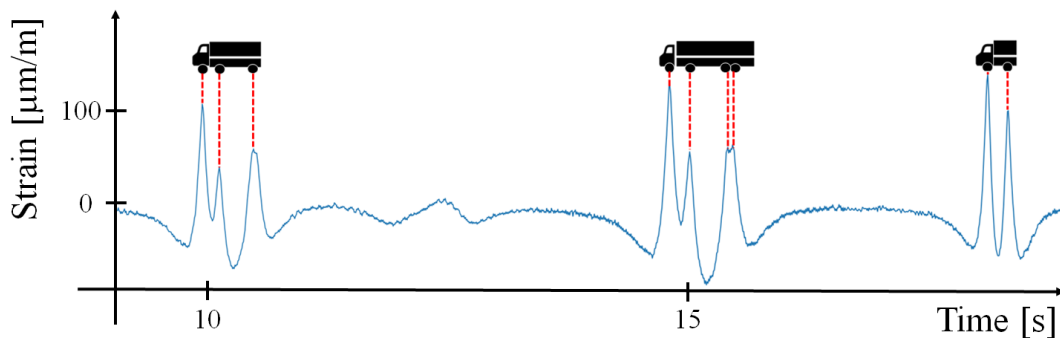
Kutatásaink alkalmazása valós körülmények között a komáromi Monostori hídon történik meg. A BME Építőmérnöki Kara Hidak és Szerkezetek Tanszékének kutatói alakítottak ki egy monitoring rendszert a hídon. Ez egy acélszerkezetű ortotróp hídpálya [7]. A híd a magyar-szlovák határon helyezkedik el. A híd egyes sávjainak az azonosítása szempontjából lényeges ez. A Magyarországról Szlovákiába haladó forgalmat lebonyolító sáv HU→SVK jelöléssel, míg a Szlovákiából Magyarországra haladó forgalomnak helyet adó sáv SVK→HU jelöléssel lesz hivatkozva a későbbiekben.



2.1. ábra. A híd szerkezete oldalnézetből (fent) és az egyik keresztmetszetben (lent). A jármű ikonok illusztrálják, hogy milyen irányban közlekednek a járművek. Látható, hogy az egyik keresztmetszetben a T1-T16 elnevezésű nyúlásmérő bélyegek hogyan helyezkednek el. A T8 és T9 nyúlásmérő bélyegek között található a sávelválasztó vonal pozíciója.

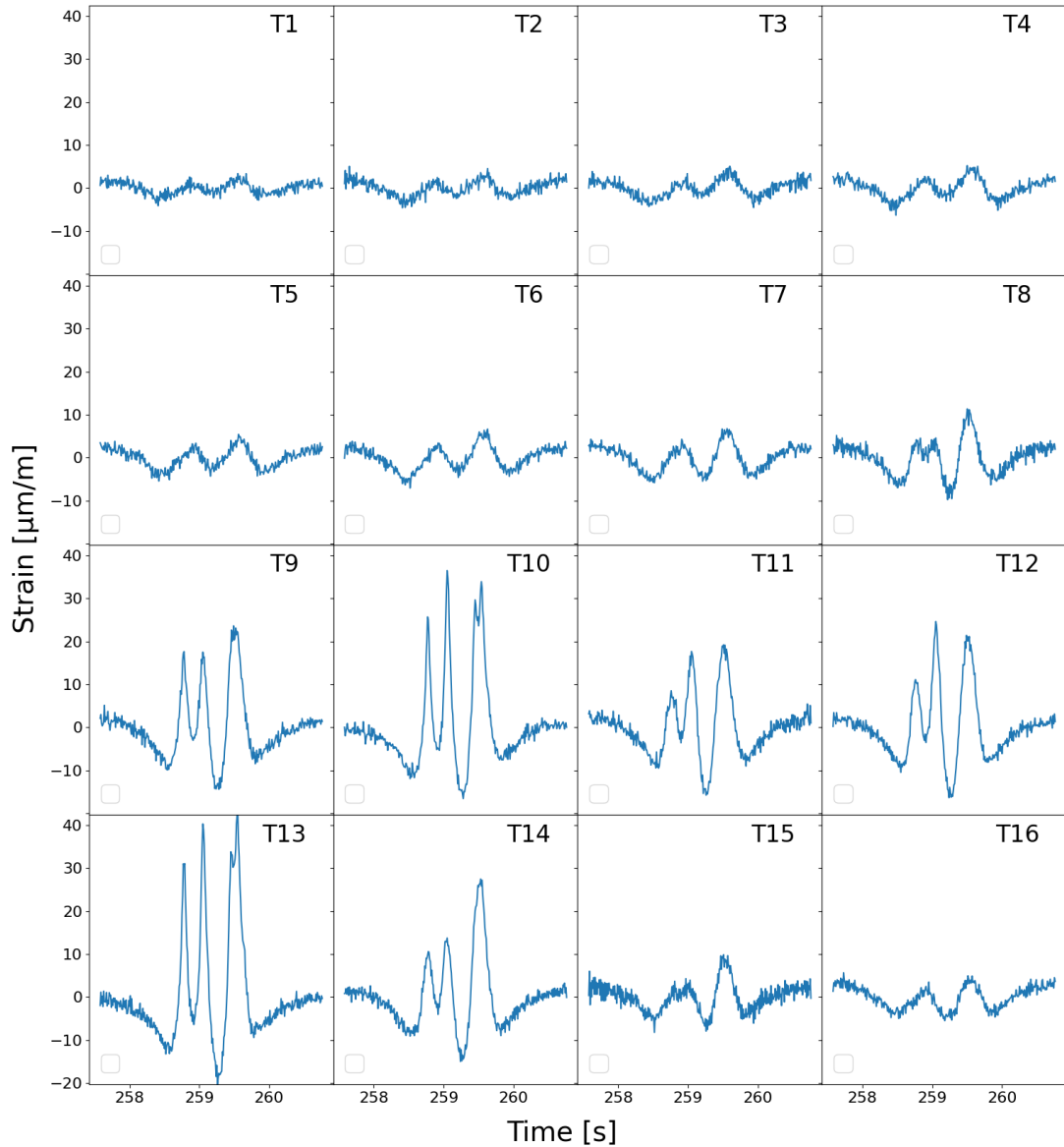


2.2. ábra. Az egymástól 6 méterre elhelyezkedő fő- és támogató szenzorsor keresztmetszetének elhelyezkedését mutatja be az ábra.



2.3. ábra. Egy sáv szenzorai által mért jelek összege olvasható le az ábráról. A járművek sziluettje alapján beazonosítható, ahogy egy-egy tengely áthaladt a hídon a fő szenzorsor keresztmetszetében. Látható, hogy a tengelyek áthaladása nagyobb megnyúlásváltozásokat okoz.

A monitoring rendszer a híd aljára szerelt nyúlásmérő szenzorok adatait fogadja. Ezek a nyúlásmérő szenzorok a hidat hosszirányban tartó trapézbordákra vannak felragasztva egy-egy keresztmetszet keresztmetszetében. A nyúlásmérő szenzorok két keresztmetszetben vannak felszerelve. A két keresztmetszet egymástól hat méterre helyezkedik el. Az egyik keresztmetszetben 16, a másik keresztmetszetben 10 nyúlásmérő bélyeg szolgáltat számunkra adatokat. Az előbbi keresztmetszetbe tartozó szenzorokat fő szenzorsornak nevezzük a továbbiakban, az ide tartozó szenzorok elnevezése T1-T16. Az utóbbi keresztmetszetbe tartozó szenzorokat másodlagos/támogató szenzorsornak nevezzük. A T201-T205 és T212-T216 elnevezésű szenzorok tartoznak ebbe a szenzorsorba. A 2.1 ábra bemutatja a



2.4. ábra. Egy pár másodperces részlete látható a T1-T16 nyúlás-mérő bélyegek által szolgáltatott adatoknak. Az egyes szenzorokhoz tartozó idő-nyúlás grafikonokból kiolvasható, hogy egy 4 tengelyes jármű haladt át a hídon. Látszik, hogy a kerekei a T10 és a T13 szenzor trapézbordáihoz közel haladtak el.

T1-T16 szenzorok elhelyezkedését. A szomszédos trapézbordákon elhelyezkedő bélyegek egymástól 60-60 cm távolságra helyezkednek el. A T201 szenzor a T1 szenzorral megegyező keresztirányú pozícióban helyezkedik el egy másik keresztmetszetben. A 2.2 ábra a fő és másodlagos/támogató szenzorsor elhelyezkedését mutatja be. Látható, hogy a fő szenzorsor keresztmetszete a pilontól 1,5 méterre helyezkedik el, a támogató szenzorsor keresztmetszete pedig 7,5 méterre, tehát egymástól 6 méterre van a két szenzorsor keresztmetszete.

A rendszer 100 Hz frekvenciával monitorozza a trapézborda nyúlását. A nyúlás (strain) mértékegysége a microstrain ($\mu\text{m/m}$). A 2.3 ábra bemutatja, hogy a járművek áthaladása mérhető nyúlásváltozást okoz a fő szenzorsor által mért jelben. A nyúlásadatok alapján a tengelyek szenzorsorok feletti áthaladási időpontja megállapítható. A 2.4 ábrán

látható példa bemutatja, hogyan néznek ki a fő szenzorsor által szolgáltatott jelek egy kb. 5 másodperces időablakban, miközben áthalad egy jármű a szenzorsor felett. A grafikonokból az is látható, hogy az egyik sávban áthaladó jármű a másik sávban is gerjeszt megnyúlást, de kisebb mértékűt. Látható, hogy keresztirányban a keletkező nyúlásváltozás mértéke csökken, ahogy egyre távolabb kerülünk a jármű kerekeitől.

Az ábrákon az is látható, hogy valamiféle "zaj" figyelhető meg a jelekben. Ez a zaj a járművek által keltett dinamikus hatások és a külső környezeti tényezők eredményeként áll elő. Külső környezeti hatások közé tartozik a szél vagy a hőmérséklet például. A hőmérsékletváltozás a híd nyugalmi megnyúlásának mozgóátlagát befolyásolja csak hosszabb távon. Mivel a jármű áthaladása a lényeges keresztmetszetben csak pár másodperc, így a hőmérséklet lényegi módon nem befolyásolja a mért nyúlásváltozásokat ennyi idő alatt. Ezt egy egyszerű mozgóátlag alapú eltolással lehet kezelni. A többi hatás kezelése már másfajta kezelést igényel, erről a továbbiakban még lesz szó.

3. fejezet

Irodalom áttekintése

Az egyes járművek tengelyterheléseinek kiszámításához előzetesen több jellemző kiszámítására is szükség van: első lépés általában a járművek időablakainak detektálása [8]. A járművek időablaka alatt azt az időintervallumot kell érteni, amikor egy-egy jármű áthaladt az egyes szenzorsorok felett. Az időablak meghatározásához hozzátartozik a sáv megállapítása is. A sebesség kinyerése [9] és a tengelyek detektálásának [10] lépése következik ezek után. A sebességről általában feltételezik, hogy konstans, mivel amíg a jármű áthalad a szenzorok szempontjából lényeges rövid (10-15 méter) szakaszon, addig a sebesség nem szokott lényegesen változni haladási sebesség esetén. Az ilyen rendszereket általánosságban high-speed weigh-in-motion (HS-WIM) rendszereknek nevezik [11]. A tengelyek detektálása esetében pedig az a feladat, hogy azokat az időbélyegeket kell megtalálni, amikor egy-egy tengely áthaladt közvetlenül az egyik szenzorsor felett. Az így kapott információk alapján már lehetséges a tengelyek terheléseinek megállapítása [12]. Az egyes jellemzőket általában egy-egy dedikált algoritmus számítja ki, azonban vannak olyan módszerek is, amelyek akár több jellemzőt is egy modulban számítanak ki [6, 13, 14]. Emellett léteznek módszerek, amik más jellemzőket is kinyernek, mint a járművek pontos keresztirányú pozíciója [14, 15, 16], hogy ezzel is támogassák a pontosabb tengelysúlybecslést, azonban a legtöbb tengelyterhelés-becslő algoritmus számára ezek nem szükségesek. A fejezetben bemutatásra kerül az irodalomban fellelhető megoldások áttekintése. A dolgozat fókuszában lévő sebességbecsléssel és tengelydetektálással részletesen is foglalkozik a fejezet.

3.1. Szimuláció

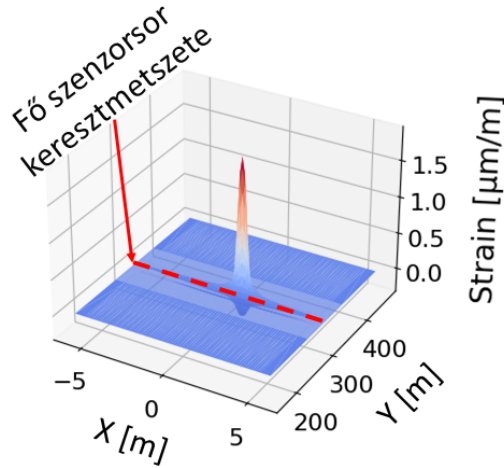
Amennyiben ismerjük egy jármű pontos paramétereit, a jármű tengelyeinek terheléseit, kerekeinek felfekvéseinek területét, tengelyeinek szélességét, sebességét, keresztirányú pozícióját és tengelyeinek távolságait, akkor a jármű áthaladása által a hídban gerjesztett nyúlás, és így a nyúlásmérő szenzorok által mérhető értékek szimulálhatóak. Mivel a legtöbb módszer épít erre a szimulációs algoritmusra, ezért ebben az alfejezetben ez bemutatásra kerül.

Egy adott időpillanatban az s szenzor által mérhető megnyúlás (strain_s) kiszámítása lehetséges a járművek pillanatnyi pozíciójának és tengelyterheléseinek ismeretében. A szimuláció megértésében segít a következő képlet:

$$\text{strain}_s = \sum_{(A,W) \in T} \frac{W}{\iint_A 1 \, dx \, dy} \iint_A I_s(x, y) \, dx \, dy \quad (3.1)$$

A képletben az I_s függvénynek kiemelt jelentősége van. Az I_s függvény adja meg az s szenzorhoz tartozó határfelületet. Az s szenzorhoz tartozó határfelület megadja, hogy a

T7 szenzorhoz tartozó hatásfelület



3.1. ábra. A T7 szenzorhoz tartozó hatásfelület. A hatásfelület jelen esetben a keresztirányú (X) és hosszirányú (Y) pozíció függvényében megmutatja, hogy 1 kN erő hatására mekkora nyúlásértékek lesznek mérhetőek az adott szenzorban által mért jelben.

hídon egy pontban egységnyi súlyú test mekkora megnyúlásváltozást okoz a szenzor által mérhető értékekben. A hatásfelület így a keresztirányú pozíció (x) és a hosszirányú pozíció (y) függvénye. A hatásfelületet mutatja be a 3.1 ábra.

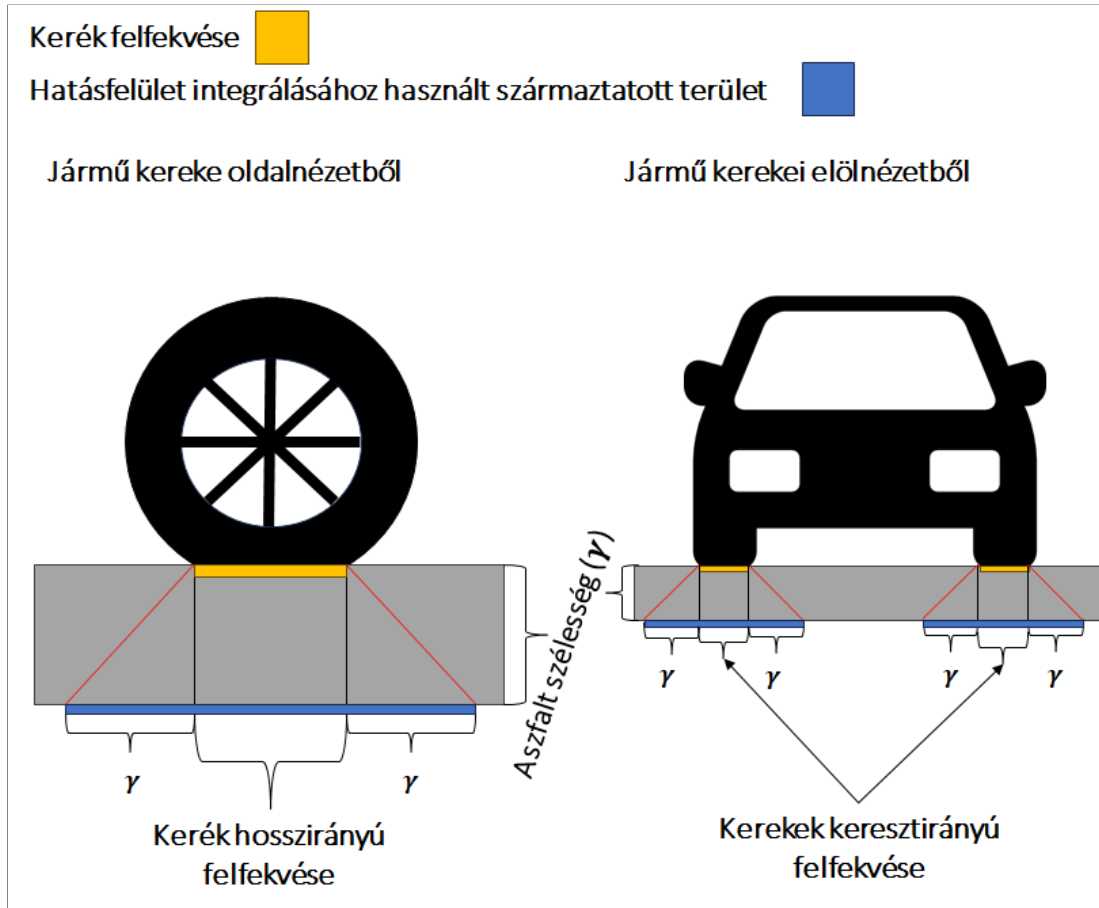
A képletben a $T = \{(A_1, W_1), (A_2, W_2), \dots, (A_n, W_n)\}$ halmaz a hídon lévő kerekek felfekvéseinek paramétereit írja le egy adott időpillanatban. Az A_i paraméter az i sorszámú kerék felfekvésehez tartozó területet mint pontok halmazát írja le, ahol a hatásfelületet ki kell integrálni. A W_i paraméter az ugyanezen sorszámú kerékre nehezedő súlyt írja le, tehát ez az az erő, amivel a kerék nyomja a híd felületét. A képletből is látható, hogy az erő nagyságával lineárisan nő a mérhető megnyúlásváltozás mértéke. A T halmaz minden időpillanatban a jármű paramétereinek és sebességének ismeretében kiszámítható. A szimulált járműveink esetében feltételeztük, hogy a felfekvések téglalap alakú területek. Az A_i terület kiszámításakor az aszfalt vastagságát is figyelembe kell venni. Ez azt jelenti, hogyha az i sorszámú kerék pozíciója az (x, y) koordinátákkal írható le, a kerék szélességét t_w , a hosszirányú felfekvését pedig t_l jelöli, akkor az A_i felület a következő módon adódik, ahol a képletben a γ paraméter az aszfalt vastagságát (asphalt tickness) írja le:

$$A_i = [x - (\frac{t_w}{2} + \gamma), x + (\frac{t_w}{2} + \gamma)] \times [y - (\frac{t_l}{2} + \gamma), y + (\frac{t_l}{2} + \gamma)] \quad (3.2)$$

A 3.2 ábra jól illusztrálja a A_i felület kiszámítását. Lényegében az erő tovaterjedését kell figyelembevenni.

A szimuláció részletes leírása a komáromi Monostori híd esetében, és a hatásfelület előállításához szükséges végeselemes módszer (VEM, Finite Element Method, FEM) alapú számítások az [5] cikkben olvashatóak.

Egyes módszerek a hatásfelület koncepciója helyett a hatásvonal koncepcióját használják [12]. A hatásvonal koncepciójának alapja, hogy az egyes szenzorok által mért jelek összege (összegjel) egy jármű esetén a keresztirányú pozíciótól (x) csak alacsony mértékben függ. Így létrehozható a hatásvonal, ami egy jó közelítést ad az előbb említett összegjelre, és ami kizárólag a hosszirányú (y) pozíciótól függ, a keresztirányú pozíciónak



3.2. ábra. Az aszfalt vastagságának figyelembevételével adódó felület (kék) kiszámítása az eredeti, aszfalt felületén mérhető (sárga) felfekvés paramétereiből.

már nem függvénye. A hatásvonal kiszámítása általában egyszerűbb, valós mérési adatok alapján, a végeselem-módszer alapú szimuláció nélkül is kiszámítható. A hatásfelülettel azonban pontosabb tengelysúlymérő rendszerek létrehozása tűzhető ki célul, mivel így pontosabb lokális információval rendelkezünk. Ez nyilván magával vonzza azt is, hogy az előfeldolgozó lépések során a járművek pontosabb paramétereit kell kinyerni [17]. Az első BWIM-rendszerek [12] a hatásvonal koncepcióját használták.

3.2. BWIM-algoritmusok

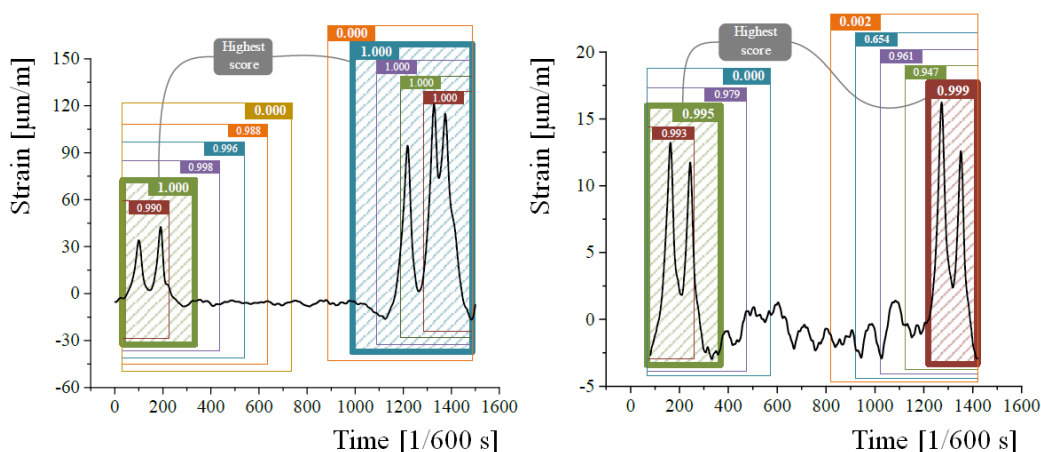
A BWIM rendszerekben használt szenzorok típusai változatosságot mutatnak. Nyúlásmérő (strain) [18], gyorsulásmérő (acceleration) [4], vibrációmérő (vibration) [14], optikai alapú (fiber-optic) [19] és nyírás mérő (shear strain) [20] szenzorokat is használnak. Ezek mellett még meg kell említeni, hogy egyes rendszerek kameraképek [17] adatait használják. A szenzoros háttér erősen befolyásolja az egyes részfeladatok megoldásához szükséges algoritmusok komplexitását. Bizonyos részfeladatok megoldása céljából telepített szenzorokkal az algoritmusok bonyolultsága jelentősen redukálható, azonban a szenzorok számának növekedésével nyilván a rendszer költsége is növekszik. A korábban bemutatott monitoring rendszernek megfelelően ebben a fejezetben a nyúlásmérő bélyegek adatain alapuló megoldásokkal fókuszban tárgyaljuk az irodalmat, de az egyéb megoldásokra is kitérünk. Ebben az alfejezetben az algoritmusokat a részfeladatok szerint csoportosítva taglaljuk.

3.2.1. Ablakdetektálás

Ahhoz, hogy további számítások legyenek elvégezhetők a szenzorok által kinyert adatokból, azonosítani kell, hogy egy adott jármű mikor haladt el a senzorsorok felett. Erre azért van szükség, mert a senzorsor feletti áthaladás időablakában tapasztalhatóak a legnagyobb megnyúlásváltozások a híd szerkezetében, és ezen részletei a mért jelnek a legfontosabbak.

Jelenleg olyan megoldások terjedtek el a korábban említett feladat megoldására, mint a kameraalapú járműfelismerés [17], nyúlásmérő szenzorok [8], illetve gyorsulásmérő szenzorok [4] használata. A forgalomfigyelő kamerák segítségével komplexebb forgalmi szituációk (mint például sávok közötti áthaladás egy előzésnél) is kezelhetővé tehetők [17]. Emellett a kameraképek segítségével akár a járművek rendszámát is le lehet olvasni, amennyiben szükséges. Azonban pusztán a kameraképből a tengelyek terhelésére következtetni nem lehet, ezért a kamerát általában csak kiegészítő eszközként használják más szenzorok mérései (esetünkben a nyúlásmérő) mellett. Léteznek olyan módszerek, amik esetében a kamerák csupán a tanító adatsor felvételéig maradnak a hídon, amíg a BWIM rendszer nem működik az elvárásoknak megfelelően [3, 14]. Ezeket a megoldásokat a későbbiekben a 3.3 fejezetben részletesen tárgyaljuk.

Megoldást jelenthet a feladatra még az úgynevezett NMS (Non-Maximum Suppression) megközelítés használata [8]. Az NMS algoritmus gyakran alkalmazott az objektumdetektálás [21] területén, ebben az esetben a hídon elhelyezett nyúlásmérő szenzorok jeleiből gépjármű ablakok kinyerésére. Az algoritmus először bizonyos konvolúciós filterek (cascade filtering) és lokális szélsőérték-kereső algoritmusok alkalmazásával naiv módon kinyer a jelből járműablakokat. Ezek a jármű ablakok azonban még tartalmazhatnak olyan eseteket, amik valójában egymáshoz közel haladó járművek (close-range following vehicles, CRFVs), csak az algoritmus egyszerűsége okán ezeket egy járműnek vette. A [8] cikkben neurális hálózatokat és az NMS algoritmust alkalmazzák az egymáshoz közel haladó járművek szétválasztására. A folyamat a 3.3 ábrán van szemléltetve. A korábban egy járműként meghatározott ablak időintervallumában lettek keretek kiszámítva. Egy-egy keret egy-egy járműablak detekcióját mutatja bizonyos konfidenciával, aminek kiszámítása egy neurális háló segítségével történik. A keretek ezek után úgy lettek kiválasztva, hogy konfidenciájuk kombinációja maximális legyen, és ne fedjék át egymást. A kiválasztott keretek lettek a már szétválasztott járművek ablakai, a többi keret eltávolításra került.

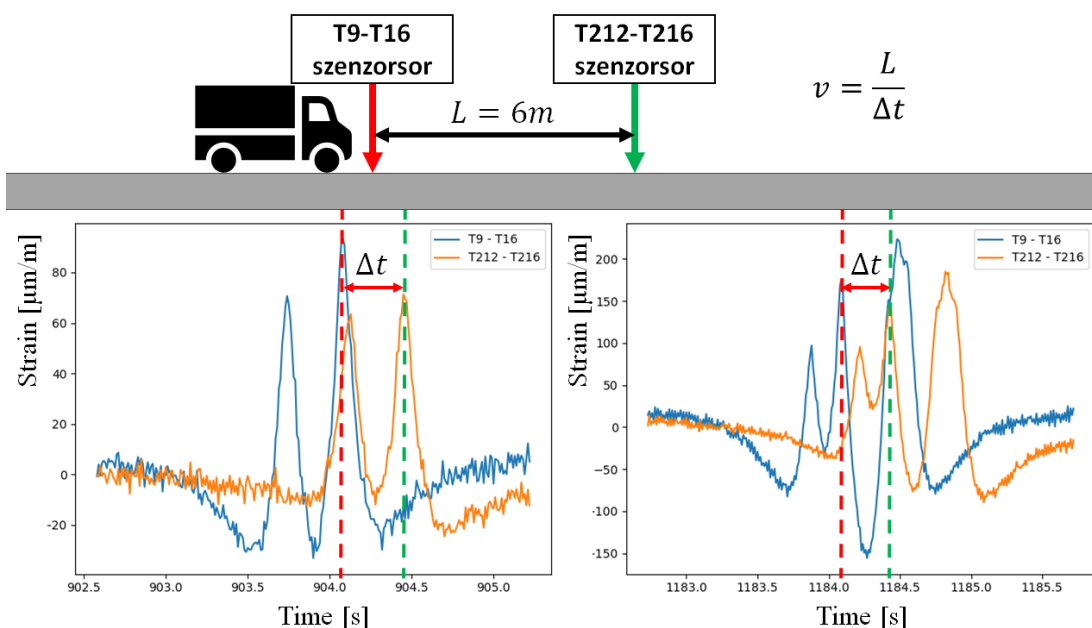


3.3. ábra. A grafikonokon egy-egy CRFV eset látható a [8] cikkből. Az egyes járműdetekciók konfidenciája a keretek sarkában látható. A legmagasabb konfidenciájú, át nem fedő keretek lettek kiválasztva a járművek ablakainak, ahogyan az látható.

3.2.2. Sebességbecslés

A különböző sebességbecslő módszerek áttekintése történik meg ebben az alfejezetben, fókuszban a nyúlásmérő szenzorokat alkalmazó rendszerekben is implementálható módszerekkel.

A legtöbb sebességkinyerő módszer használja a másodlagos szenzorsor által szolgáltatott adatokat is [22, 23, 24, 25]. A másodlagos szenzorsor telepítésének általában csak ennél a lépésnél van szerepe. A két szenzorsorban mérhető jelek általában hasonló alakot vesznek fel. Ez teszi lehetővé a két szenzorsor feletti áthaladás közötti időkülönbség kiszámítását. Ez az időkülönbség megadja, hogy a két szenzorsor közötti 6 méteres távolságot mennyi idő alatt tette meg a jármű. Innen erre a szakaszra egy átlagsebesség könnyedén kiszámítható a távolság és az időkülönbség hányadosaként. Mivel a sebességet a HS-WIM rendszerekben közel állandónak tekinthetjük, az átlagsebesség ezen a szakaszon a valós sebességet jól közelíti. A 3.4 ábra illusztrálja mindezt.

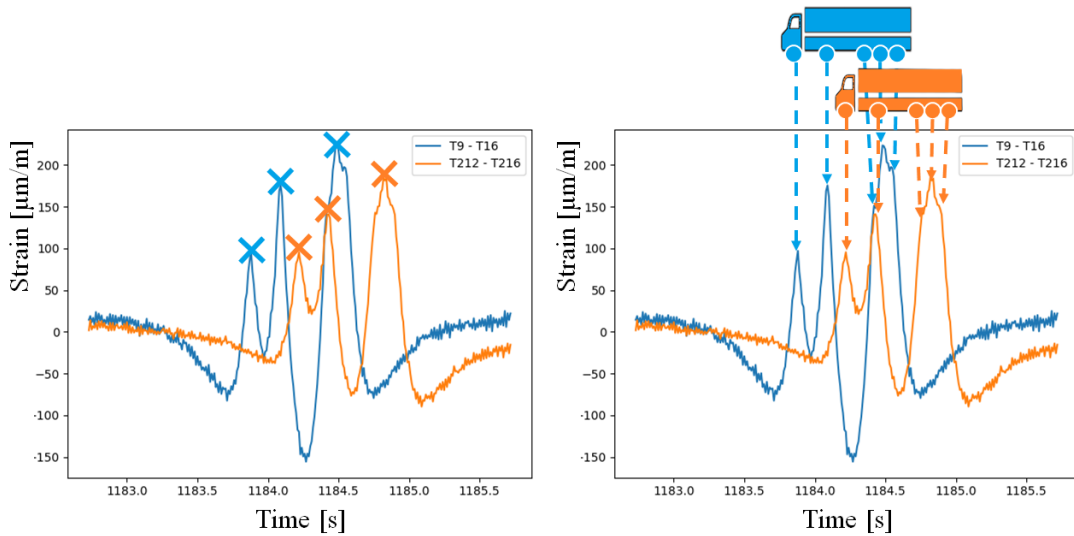


3.4. ábra. Az ábra a két keresztmetszetben mutatja egy jármű áthaladásakor keletkező összegjeleket. Látható, hogy a két keresztmetszetben gerjesztett jel, mennyire hasonló. A Δt időeltolás ismeretében és az $L = 6m$ keresztmetszetek közti távolság ismeretében a sebesség (v) már könnyedén kiszámítható.

Egyes módszerek vizsgálják azt, hogy a jármű sebessége változhat, amíg áthalad a szenzorok szempontjából lényeges szakaszon, így a jármű sebességét nem egy konstans értéként kezelik [26].

A két szenzorsor adatainak összehasonlítását a legtöbb módszer a jelekben megfigyelhető lokális szélsőértékek pozíciói alapján végzi el. Erre ad illusztrációt a 3.5 ábra. Mivel a lokális maximumok problémája összefügg a tengelypozíciók időbélyegeinek meghatározásával, ezért gyakran a sebességhatározás lényegében azt jelenti, hogy detektáljuk, hogy a két szenzorsorban a jármű tengelyei mikor haladtak át, és a tengelyenkénti időeltolásokból egy átlagolt értéket veszünk az időeltolásaként. A tengelydetektálás problémáját a későbbiekben részletesen is kifejtjük.

Kawakatsu és társai [3, 14] kutatásában neurális háló segítségével sikerült megmutatni, hogy a sebesség információ akár egy szenzorsor alapján is kinyerhető lehet, a mért



3.5. ábra. Az ábrákon az látható, hogy a lokális maximum (bal) és a tengely alapú (jobb) oldali esetben mely pontokat kell összeegyeztetni a két keresztmetszetben, hogy a Δt időkülönbséget ki tudjuk számolni.

hiba azonban egészen magasnak bizonyult az esetek jelentős részében. A sebességet 3,23 km/h átlagos hibával képes ez a rendszer becsülni. A hibák szórása 6,26 km/h volt. A járművek sebessége 78 km/h mediánú és 7,75 km/h szórású eloszlást mutatott. A megoldás egy konvolúciós neurális hálóra épül, ami bemenetként a nyúlásmérő szenzorok jelét kapta meg. A hálót a Mean Squared Error veszteségfüggvénnyel tanították. A tanító adathalmaz valós felcímkézett adatsor volt, amit egy négy sávós hídon rögzítettek. Ez a megoldás a sebességbecslés mellett a tengelydetektálást, a keresztirányú pozíció becslését és tengelysúlybecslés feladatát is képes megoldani. A 4.1 fejezetben bemutatott általunk fejlesztett sebességbecslő erre a megoldásra épül, ezt fejleszti tovább.

3.2.3. Tengelydetektálás

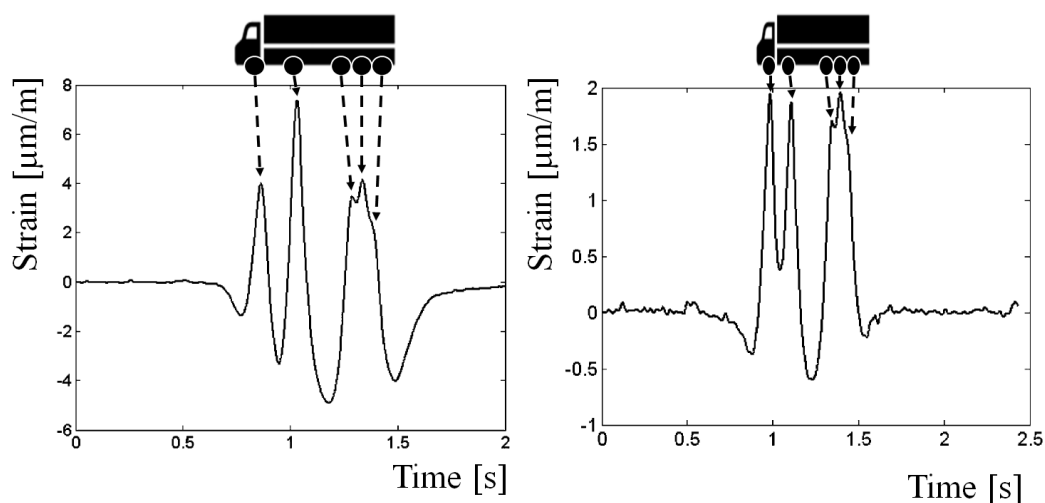
A tengelydetektálás vonatkozhat az áthaladó jármű tengelyeinek számának megállapítására vagy a tengelyek fő szenzorsor feletti áthaladásának időbélyegeinek meghatározására is. Természetesen az utóbb említett folyamat komplexebb, de a tengelysúlyok kiszámításához erre van szükség, valamint ennek a feladatnak a megoldásából a tengelyszám is származtatható.

A tengelydetektálás szempontjából nem mindegy a híd szerkezete. A hidak szerkezetük alapján több csoportba is sorolhatóak. A közúti hidak szempontjából lényegesek az ortotróp acélpályával és a vasbeton szerkezettel rendelkező hidak. A két híd típus esetében a tengelydetektálás feladata más komplexitású. Az ortotróp acélpályás hidak esetében a különböző típusú szenzorok által mért jelekben lokális szélsőértékek formájában jelennek meg általában a hídon áthaladó járművek tengelyei. A vasbeton szerkezetű hidak esetében ez nem ennyire triviális feladat. Vannak olyan komplex szituációk, ahol kettő (tandem) vagy esetenként három (tridem) tengely nagyon közel helyezkedik el egymáshoz, ahogy a 3.6 ábrán is látható. Vasbeton szerkezetű hidak és kizárólag nyúlásmérő szenzorok használata esetén ezek megtalálása komplex feladattá válhat. A 3.7 ábra megmutatja, hogy az ilyen esetekben a tengelyek áthaladása a szenzorsor felett nem implicálja a jelben mérhető

lokális maximumokat, tehát összetettebb algoritmusokra van szükség, hogy ezen helyzetek is kezelhetők legyenek a tengelydetektálásnál.



3.6. ábra. Az ábrán egy nyerges vontató látható. A rajta elhelyezkedő tridem tengelycsoport pirossal bekeretezve látható.



3.7. ábra. Az grafikonok a [10] cikkből származnak, rajtuk az látható, hogy vasbeton szerkezetű hidakon nem minden tengely áthaladása jár együtt lokális maximumokkal.

Megoldást jelenthet a bemeneti jel második deriváltjának [26] vizsgálata. Ez a módszer azonban érzékeny a már korábban is említett zajra [4], illetve érzékeny lehet a másik sávban elhaladó jármű által gerjesztett nyúlásra. A zajszűrő megoldások kidolgozása ilyen esetben szükséges.

További tengelyszámbecslésre használt algoritmus a Virtual Axle Method (virtuális tengely módszer) [13, 27]. Ez a megoldás egyszerre oldja meg a tengelysúlyok becslését és a tengelyek detektálását. Az algoritmusnak bemeneti adatként szüksége van a jármű sebességére, így a tengelydetektálás ebben az esetben függ a sebességbecslőtől. A módszer úgy működik, hogy feltételezzük, hogy a járműnek számos, hosszirányban egyenletesen

elosztott virtuális tengelye van, amelyek lefedik a jármű teljes hosszának intervallumát. A módszerben a virtuális tengely arra utal, hogy nyilván vannak virtuális tengelyek, amikre az algoritmus nulla közeli tömeget ad eredményül, és ezek hosszirányában a valós járművön nyilván nincsenek tengelyek. A megközelítés a Moses-algoritmuson alapul, ami a dolgozatban később kerül kifejtésre, ott még erre az algoritmusra visszatérünk.

Gyakran alkalmazzák a Continuous Wavelet Transform (CWT) eszközön alapuló megközelítést, mely hatékony megoldást nyújt az akusztikus és vizuális jelek elemzésére. Korábbi tanulmányok már igazolták hatékonyságát tengelydetektálásban [4, 10]. A [4] cikkben a wavelet-transzformált jelek további elemzésre kerülnek egy neurális hálózat segítségével, amely binárisan osztályozza a bemeneti jeleket, hogy egy adott időpillanatban (ami valójában egy mintavételnyi nagyon rövid időszakasz) haladt-e át tengely a fő keresztmetszetben. Azonban a wavelet alapú eljárások hátránya lehet, hogy a paraméterek nagy számát kell hidanként újra konfigurálni, ami nehezítheti a tesztelést. A különböző hidakra különböző wavelet kiválasztása működhet például, aminek megtalálása hosszadalmas folyamat lehet.

A jelek frekvencia-idő tartományának analízise során leggyakrabban a Short Time Fourier Transform (STFT), illetve a CWT módszert választják [4]. Az STFT rögzített ablakkal rendelkezik, így bizonyos frekvenciájú tartományokban már kevésbé működőképes. Ezzel szemben a CWT alkalmazkodó ablakmérettel számol, így lehetővé teszi a jobb idő-, és a jobb frekvenciatartománybeli lokalizációt. Ez a rugalmasság gyakran alkalmassá teszi a CWT-t komplex jelek elemzésére.

A [3] cikkben neurális hálózatot használtak a tengelyek detekciója céljából. A tengelyeket 0,354 méter átlagos abszolút hibával (MAE) találja meg, míg az négyzetes hibák átlagának gyöke (RMSE) 0,693 méter volt. Ez a megoldás az előző fejezetben említett neurális hálózat alapú sebességbecslő megoldással együtt jelent meg.

Az általunk fejlesztett tengelydetektáló neurális hálózatot a 4.2 fejezetben mutatjuk be. A megoldásunk elsősorban a [4] cikk megoldását fejleszti tovább.

3.2.4. Tengelysúlybecslés

Az első tengelysúlybecslésre használt algoritmus, ami kizárólag nyúlásmérő szenzorok adatait használta, az úgynevezett Moses-algoritmus volt. A Moses-algoritmus közel 40 éves algoritmus, azóta több olyan algoritmust is fejlesztettek, ami erre az algoritmusra épít. Emellett az elmúlt években a mélytanuló algoritmusok elterjedésével több újszerű neurális hálót alkalmazó módszert is kifejlesztettek.

Bár a tengelysúlybecslés részfeladata ennek a dolgozatnak nem áll közvetlen fókuszában, a Moses-algoritmust részletesebben is bemutatjuk, hogy érthetővé váljon a 3.2.3 alfejezetben már említett virtuális tengely módszer (Virtual Axle Method), ami épít a Moses-algoritmusra, és egyszerre végzi el a tengelyek detektálását és tengelysúlybecslését.

A Moses-algoritmus [12] a következő módon működik: feltételezzük, hogy sikerült kinyerni a sebesség jellemzőt és a tengelypozíciókat. Ezek után a célunk, hogy megtaláljuk azokat a tengelysúlyokat, amikkel a szimuláció során kiszámítható összegjel a lehető legjobban megközelíti a valós körülmények között mért összegjelet valamilyen hibafüggvény szerint. Alább a problémát formalizáltan is megadjuk:

$$\min_P ||IP - M||_2^2, P \in \mathbb{R}^n \quad (3.3)$$

A 3.3 kifejezés optimalizálási problémaként írja le a feladatot. Az $P \in \mathbb{R}^n$ vektor egy n tengellyel rendelkező jármű tengelysúlyait reprezentálja. A P_i érték az i . tengely terhelését mutatja. Az $M \in \mathbb{R}^T$ vektor a valós mért jelet írja le. Az M_t érték a t időpontban a jármű

sávjában mérhető jelek összegével egyezik meg. Az $I \in \mathbb{R}^{T \times n}$ mátrix a hatásfelület vagy hatásvonal koncepcióját alkalmazva nyerhető ki:

$$I = \begin{bmatrix} I_{1,1} & I_{1,2} & I_{1,3} & \dots & I_{1,n} \\ I_{2,1} & I_{2,2} & I_{2,3} & \dots & I_{2,n} \\ \dots & & & & \\ I_{T,1} & I_{T,2} & I_{T,3} & \dots & I_{T,n} \end{bmatrix}, \text{ where } I_{t,i} = IL((t - A_i) \times v) \quad (3.4)$$

A 3.4 képletben az IL a hatásvonalat jelöli. Az $I_{t,i}$ érték megadja, hogy az i . tengely mekkora megnyúlást okozna t időpillanatban, amennyiben egységnyi súlyú lenne. Az egyes tengelyek áthaladási idejét A_i adja meg. A képlet szerint a fő szenzorsor a nulla hosszirányú pozícióban helyezkedik el. Ez a képlet lényegében a már korábban bemutatott szimulációs eljárásnak egy egyszerűsített leírása. A probléma során a cél, hogy a szimulált és a mért jel L_2 normáját minimalizáljuk, azaz megtaláljuk azt a P vektort, amire a fenti kifejezés minimális. Ez a feladat a lineáris algebra területén ismert pszeudoinverz eszközzel megoldható.

A korábban már említett Virtual Axle Method nevű eljárás lényege a következő: ez az eljárás csak a sebesség ismeretét feltételezi előzetesen. Ez a módszer a tengelyek pozícióját és azok tengelysúlyait egy algoritmusban számítja ki. A probléma a következő módon módosul:

$$\min_P \|IP - M\|_2^2, \quad P \in \mathbb{R}^n, \quad P \geq 0 \quad (3.5)$$

A minimalizálandó kifejezés ugyanaz, azonban a tengelyek súlyára egy kényszert kell előírunk, méghozzá azt, hogy a tengelyek súlyának nemnegatív számnak kell lennie. A közel nulla tömegű tengely azt jelenti, hogy abban a pozícióban nincs tengely. Az algoritmus ezek után vázlatosan a következő módon működik: kiszámítja a 3.5 probléma megoldását. Az így kapott P vektor elemzésre kerül. A nulla vagy nulla-közel értékű indexeknek megfelelő időpontokban nem haladhatott át tengely a szenzorsor felett. A fennmaradt tengelyek sem biztos még, hogy valóban tengelyek. A megmaradt tengelypozíciók különböző lehetséges kombinációira kiszámítja a Moses-algoritmussal a tengelysúlyokat. Azt az esetet veszi az algoritmus a helyes tengelyelrendezésnek, amelyikkel a lehető legjobban képes minimalizálni a szimulált és a valós jel már korábban felírt különbségét. A módszer alkalmazhatósága zaj esetén kérdésessé válhat [4].

Az egyes Moses-algoritmus továbbfejlesztései még figyelembe vesznek esetleg más paramétereket, hogy jobb közelítést kapjunk. Ilyen paraméter lehet a keresztirányú pozíció [15, 16]. Az eddigiek során a feladatot egy jármű esetre írtuk le. Az irodalom foglalkozik azokkal az esetekkel is, hogy hogyan lehet olyan esetben megoldani a problémát, hogyha több sávban is haladnak járművek [22]. Ebben az esetben a pontosság általában romlik.

A mélytanulás eszköztárát nem használó statikus algoritmusok [16, 28, 29, 30, 31] áttekintését és összehasonlítását Carraro és társai [18] végezték el.

Léteznek neurális hálózat alapú tengelysúlybecslő megoldások a BWIM-rendszerekben [3, 6, 32]. Mivel ennek a dolgozatnak nem fókusza a tengelyterhelések becslésének megállapítása, ezért ezeket a megoldásokat jelen dolgozatban nem részletezzük. Ezeknek a rendszereknek amiatt lehet értelme, mert a pontatlan sebességbecslés és tengelydetektálás a tanítás során figyelembe vehető úgy, hogy zajos sebesség- és tengelypozíció-információval tanítunk. Korábban sikerült megmutatni, hogy az ilyen zajos tanításon alapuló neurális hálóval jobb eredmények érhetők el, mint a Moses-algoritmussal [33], mivel a Moses-algoritmus érzékenyebb ezekre a hibákra.

3.3. Adathalmaz

A mesterséges intelligencia algoritmusok esetén egy tanító adathalmazra lehet szükség. A különböző hidak esetében pedig dedikált adathalmazra lehet szükség. A címkézést különböző kutatócsoportok különböző módszerekkel végzik el. A feladat során nyilván az adatok felcímkézése lehet nehézkes, mivel nyers jelet a monitoring rendszer nagy mennyiségben hozzáférhetővé tesz. Ebben a fejezetben először különböző adathalmaz címkézési megoldásokat mutatunk be, majd utána az általunk használt adathalmazt mutatjuk be.

Még mielőtt ezek a címkézési eljárások bemutatásra kerülnek, látnunk kell, hogy egyes módszerek nem igényelnek nagy méretű címkézett adathalmazt. A Moses-algoritmus, és azok továbbfejlesztett változatai, valamint a lokális maximum alapú tengelydetektáló és sebességbecslő módszerek ilyenek. Ezen módszerek esetében is szükség lehet azonban ismert paraméterekkel rendelkező járművel való helyszíni kalibrációs mérésre. A kalibrációs mérésekre azért van szükség, mert ezekkel lehet validálni a végeselem-módszerrel generált hatásfelületet, vagy ezekből a kalibrációs mérési adatokból is lehetséges a hatásvonal kiszámítása [29, 34]. A validációhoz kevesebb mérés (pár tíz) is elegendő lehet.

Egyes adathalmazok kizárólag kameraképek alapján készültek [3, 8, 14]. Ezek az adathalmazok azonban csak a sebesség és tengelypozíció címkéket teszik hozzáférhetővé. Az adathalmazok annotációja ebben az esetben többféle módon is történhet. Történhet manuális módon [8], ebben az esetben egy pár ezer járműből álló adathalmaz felcímkézése is energiaigényes lehet. Más esetekben [14] az adathalmaz felcímkézése történhet képi algoritmusok, például Faster R-CNN [21] alkalmazásával. Az adatsorok származhatnak már meglévő kameraképet szolgáltató forgalom-monitorozó (traffic surveillance system, TSS) rendszerekből, vagy egy ideiglenesen, a hídon elhelyezett kamerarendszerekből is. A tanító adathalmaz létrehozása után, a betanított nyúlásmérő szenzorok adataira épülő modul már a forgalomfigyelő kamerák támogatása nélkül is képes elvégezni a feladatát, kizárólag a nyúlásmérő szenzorok adataira támaszkodva. Az ideiglenesen felszerelt kamerarendszer új BWIM-rendszer telepítésekor újra felhasználható lehet.

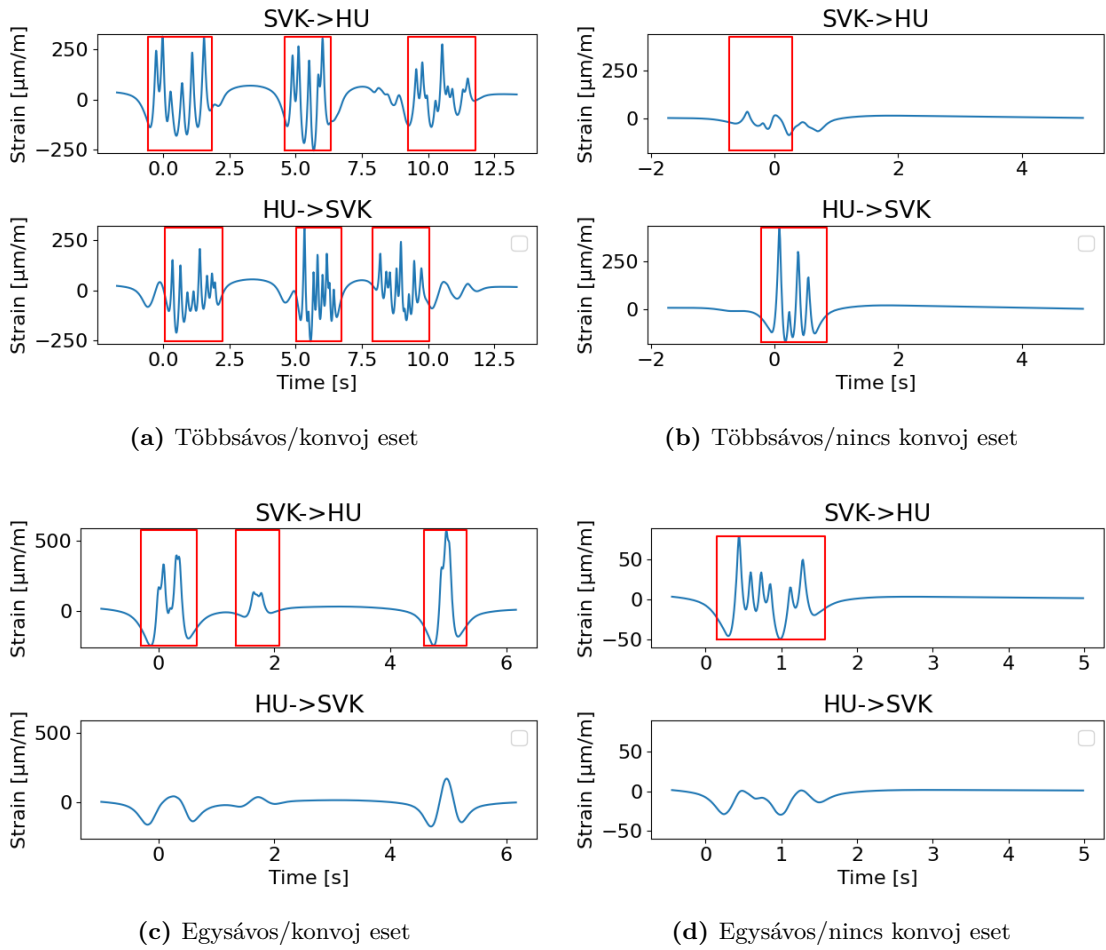
Egy másik lehetséges módszer extra szenzorok felhelyezése a hídra vagy más helyre, amit az adathalmaz elkészülte után leszerelnek. Egy vasúti híd esetében például átmenetileg a sínekre felhelyezett nyúlásmérő szenzorok segíthetnek abban, hogy a tengelypozíciók címkézése könnyedén végbemehessen [4]. Ezek az adathalmazok általában tengelysúly információkat még mindig nem tartalmaznak.

Egy harmadik lehetséges módszer szintetikus adathalmaz létrehozása. A szintetikus adathalmazok a korábban leírt szimulációs algoritmussal hozhatóak létre. Először járműparamétereket kell generálni, majd ezekkel egy-egy eset szimulálható. Ilyen adathalmaz több is hozzáférhető [5, 18], azonban ezek nem mindegyike alkalmas mélytanulás alapú rendszerek tanítására. A szintetikus adathalmazok esetében fontos kiemelni, hogy a valós mért jelekben tapasztalható zajt nem tartalmazza. Ez a zaj a dinamikus hatások és a külső hatások (szél, hőmérséklet) eredőjeként jelenik meg. Ezek szimulációja nehézkes, és sok esetben nem is lehetséges, ezért ezt a hatást különböző szórású Gauss-zajjal szokták modellezni [5, 6].

A dolgozatban a BME-Simulated-I adathalmazt használtuk, ezért ennek a részletes áttekintése olvasható a továbbiakban.

3.3.1. BME-Simulated-I

A BME-Simulated-I (BME-S1) adathalmaz [5] lehetővé teszi egyszerű statikus és mélytanulás alapú algoritmusok készítését, tanítását és összehasonlítását. Az adathalmaz publikusan is elérhető [35]. Mivel dolgozatunkban ezt használjuk a későbbiekben, ezért ebben az alfejezetben részletes bemutatásra kerül.



3.8. ábra. BME-S1 adathalmaz példák. A két sávban (SVK→HU, HU→SVK) lévő szenzorok által mérhető szimulált jelek összege van ábrázolva. Az egyes bekeretezett téglalapok egy-egy jármű áthaladását jelölik.

Ez egy szintetikus adathalmaz. Tanításra 96 012 jármű adatait tartalmazza, míg tesztelésre 12 808 jármű adatait. Ezek a járművek különböző scenáriókban haladnak át a hídon. Egy-egy scenárió egy pár vagy pár tíz másodperces időtartamot ölel fel. Az egyes scenáriók bizonyos szempontok szerint csoportokba oszthatók.

Egyik szempont, ami alapján a scenáriók csoportokba sorolhatóak, hogy a járművek mindkét sávban közlekednek-e egy-egy esetben. Ez alapján két csoport különböztethető meg: amikor mindkét sávban közlekedik jármű a sáv haladási irányának megfelelően (kétsávós/többsávós, Both-Lane, BL), és azok az esetek, amikor csak az egyik sávban közlekednek járművek (egysávós, One-Lane, OL).

Egy másik szempont, ami alapján a scenáriók csoportosíthatóak, hogy a járműveken megfigyelhető-e a korábban már tárgyalt tandem- és tridem-tengelyek. Ez alapján megkülönböztethetünk tandem (T) és tandem nélküli (Non-Tandem, NT) scenáriókat. Míg az előbbieket csoportjában a járművek tartalmaznak közeli tengelyeket, az utóbbi adathalmaz nem tartalmaz ilyen tengelycsoportokat.

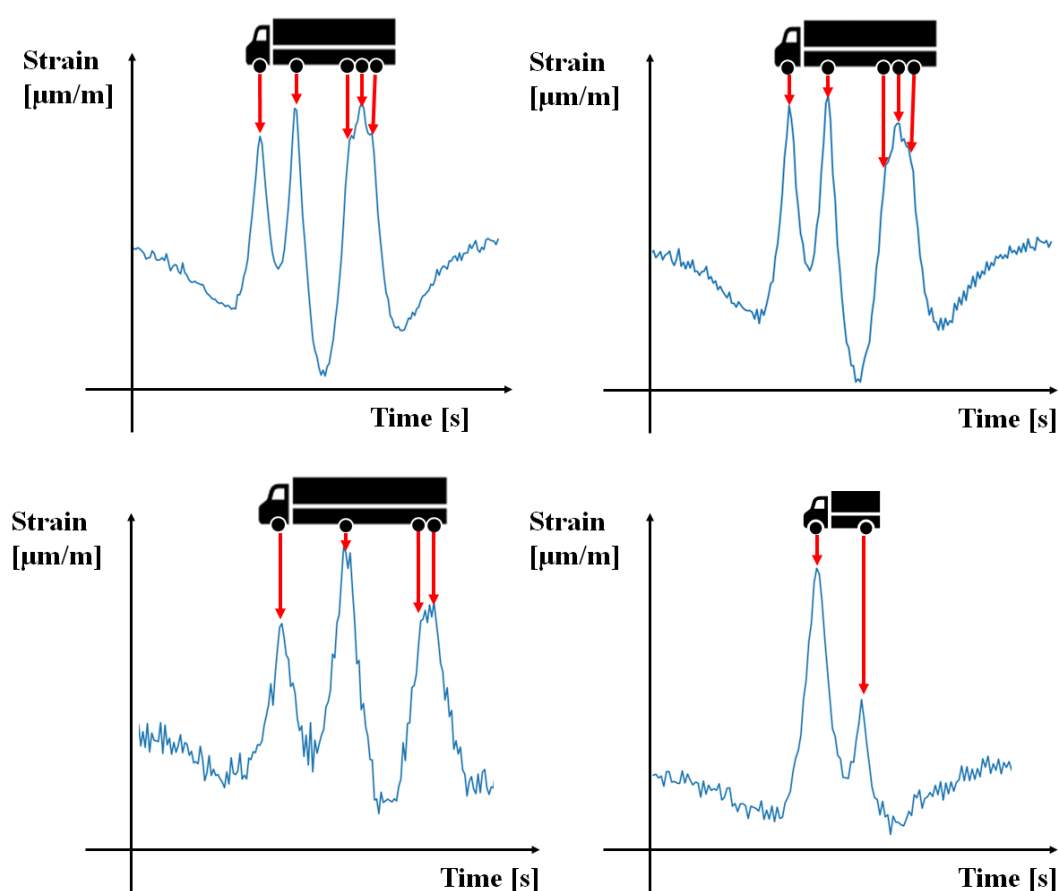
A harmadik szempont, ami alapján a scenáriók csoportosíthatóak, hogy a járművek egy-egy sávban haladnak-e konvojban. Konvoj alatt egymáshoz közel haladó járműveket (closed-range following vehicles, CRFV) értünk. Ez alapján van konvoj (Convoy, C) és

konvoj nélküli (Non-Convoy, NC) adathalmaz, az előbbi tartalmaz konvojokat, az utóbbi nem.

Ez az adathalmaz úgy lett elkészítve, hogy támogassa az inkrementális fejlesztést, tehát a legegyszerűbb esetektől (egysávos/nincs konvoj/nincs tandem, OL-NC-NT) a legbonyolultabb esetekig (többsávos/konvoj/tandem, BL-C-T) tartalmaz példákat nagy számban. Kisebb számban (kb. 10 jármű) annotált valós mérési eredményeket is tartalmaz.

A 3.8 ábra mutat példákat különböző csoportokba tartozó scenáriókra. A példák jól bemutatják azt is, hogy az áthaladó járművek milyen megnyúlásokat gerjesztenek a másik sávhoz tartozó szenzorok által mérhető jelekben. Emiatt mondhatjuk, hogy a többsávos scenáriók általánosságban komplexebb esetek.

3.3.2. Vasbeton szerkezetű hidak esetén alkalmas adathalmaz



3.9. ábra. Az ábrák egy-egy jármű sávjában mérhető összegjeleket mutatja. A járművek sematikus ábrája, és a tengelyekhez rendelt nyilak mutatják, hogy mely időpillanatokban haladt át a tengely a fő keresztmetszet felett.

A dolgozatban olyan megoldásokat kívánunk bemutatni, amelyek hordozhatóak, így a közúti hidak nagy hányadában alkalmas lehet a feladatok ellátására. Ezért az algoritmusoknak az ortotróp acélpályás és a vasbeton szerkezetű hidak esetében is jól kell működniük. A legtöbb részfeladat esetében a két híd típus között nincs nagy különbség. Ez alól a tengelydetektálás jelent kivételt. Korábban már bemutattuk, hogy ott az jelenti a problémát, hogy a tengelyek áthaladása a hídon nem jár együtt lokális maximumok

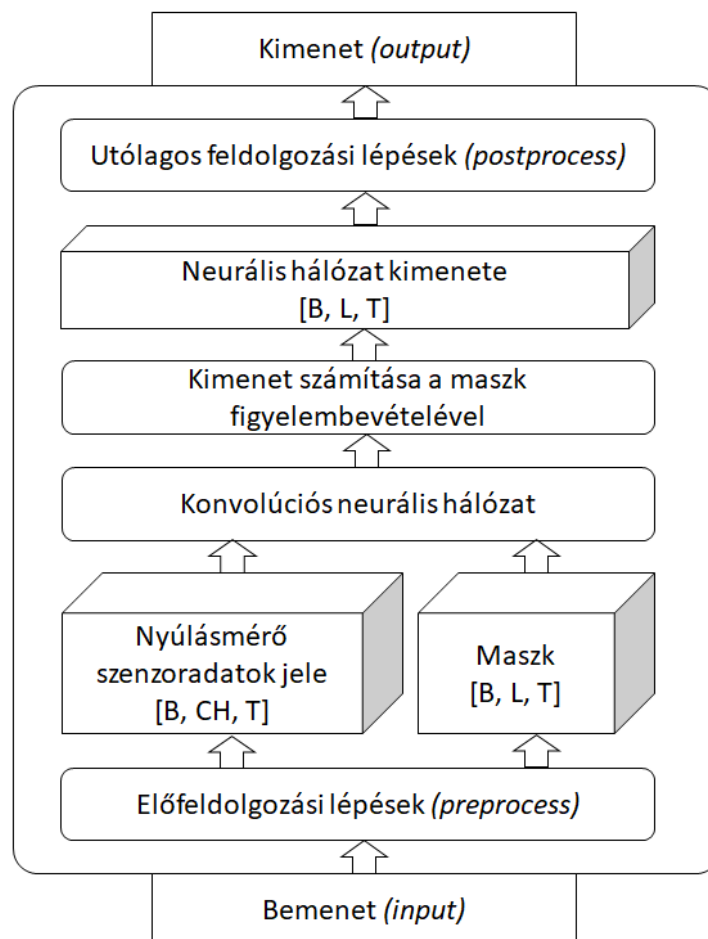
generálásával a jelekben. Ennek több oka is lehet: egyfelől a vasbeton szerkezetű hidak esetében a merevség oldalirányban nagyobb lehet, mint ortotróp acélpálya esetén. Emellett nem tipikus a sűrű kiosztású (3-4 méterenkénti) kereszttartók alkalmazása, mely a statikai viselkedést erőteljesen befolyásolja, valamint arányaiban sokkal merevebb szerkezetekről van szó, mint acélpálya esetén. Ez utóbbi azt eredményezi, hogy a zaj/jel arány megnövekszik. A [10] cikk esetében egy olyan hídon okozott problémát a tengelyek detektálása, ami esetén egy erősen vasalt vastag vasbeton lemez adja a felszerkezetet, ez okozza a tengelydetektálás nehézségét.

Mivel jelenleg nem rendelkezünk vasbeton szerkezetű hídmodellel, szimulációval és így adathalmazzal sem, ezért a következő megoldást választottuk a vasbeton szerkezet szimulációjára: a sávokban lévő szenzorok összegjelét, vagy több egymás mellett lévő szenzor összegjelét vizsgáltuk, ahelyett, hogy azokat egyesével vizsgáltuk volna. A 3.9 ábra mutatja be, hogy ilyen módon egy ortotróp acélszerkezetű hídon is elő tudjuk idézni a problémát. A 3.9 ábra és a 3.7 ábra jeleit összehasonlítva látható a hasonlóság, és hogy így a probléma ugyanúgy fogalmazható meg. A későbbiekben ilyen összegjeleken alapuló adathalmazokon is dolgoztunk, aminek a célja a vasbeton szerkezetű hidak esetében is hordozható megoldások kivitelezése volt.

4. fejezet

Saját megközelítés

Ebben a fejezetben kerül bemutatásra az általunk fejlesztett sebességbecslő- és tengelydektáló modul. Mindkettő egy-egy neurális hálózatra épülő megoldás.



4.1. ábra

Mivel a két modul vázlatosan azonos sablonra épül, ezért először ezt a sablont mutatjuk be, majd a két megoldást részletezzük a sablon specifikálásával. A 4.1 ábra mutatja be a sablont. Mindkét modul bemenetként a jármű bizonyos jellemzőit kapja meg. Ezek bizonyos előfeldolgozási (preprocess) lépéseken mennek keresztül, hogy megkapjuk a neurális hálózat bemeneteit, ami két tenzor. Ezek után a neurális hálózat feldolgozza az adatokat.

A neurális hálózat kimenete a különböző modulok esetében különböző számítások révén történik meg. Ezek után még utólagos feldolgozási lépések következnek, amik a neurális hálózat kimenetét sebességgé vagy tengelypozíciókká konvertálja.

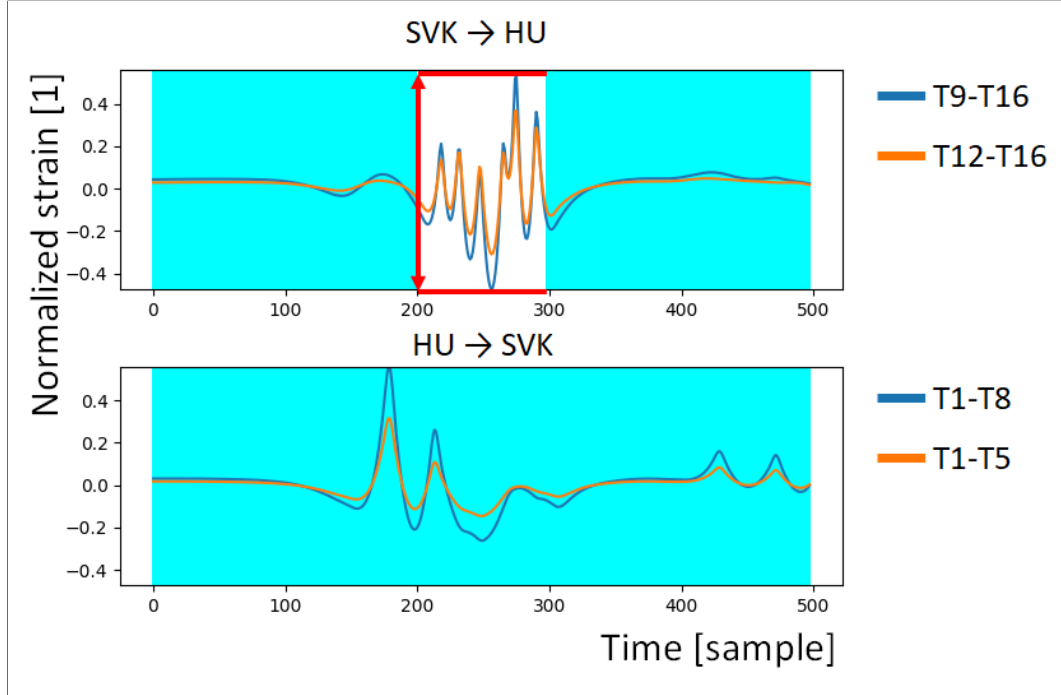
A bemeneti tenzorok mindkét modul esetében a következő módon jönnek létre: a modul bemeneteként megkapja a jármű ablakát ($[w_{begin}, w_{end}]$), és azt az információt, hogy a jármű melyik sávban közlekedik. Ezek után a nyúlásmérő szenzorok adatait kell tenzorralakítani. A tenzor mérete $B \times CH \times T$ méretű tenzor lesz, ahol B a batch mérete, CH a csatornaszám és T a jel hossza, tehát a mintavételek száma. Látszik, hogy ez egy 1D konvolúciós neurális hálózat. Minden csatorna esetében meg kell adni, hogy az mely nyúlásmérő bélyegek által szolgáltatott jelek összegét tartalmazza. Az így vett összegjeleket ezek után a következő módon transzformáljuk még: vesszük az összegjel időablakon belüli értékeinek átlagát, és az egész jelet ezzel az átlaggal eltoljuk, hogy az így képzett jel időablakon belüli átlaga nulla legyen. Erre a lépésre a valós jelek esetében azért lehet szükség, mert a hőmérsékletváltozások hatására a jel nyugalmi állapotban mérhető megnyúlása mozog. Ezek után még arra van szükség, hogy a szenzoradatok jeleit normalizáljuk. A normalizációhoz meg kell adni, hogy mely szenzorok összegjele alapján történjen meg. Ezek után vesszük a normalizáló összegjel időablakon belüli terjedelmét, és a korábban képzett csatornák jelét ezzel leosztjuk. Ez a normalizáló összegjel általában a vizsgált jármű sávjában mérhető összegjel. Emellett egy $B \times L \times T$ méretű tenzort is képzünk a neurális hálózat bemeneteként, ahol L a sávok száma. Ez a maszk bináris, 0 és 1 értékű bemeneteket tartalmaz. A vizsgált jármű sávjában és időablakában lévő mintavételek esetében 1 értéket, különben 0 értéket tartalmaz. A 4.2 ábrán lévő idő-normalizált megnyúlás grafikonok jól bemutatják a neurális háló bemeneteit. Egy kétsávós eset látható az ábrán, ahol sávonként láthatóak a csatornák. A csatornák a T9-T16, T12-T16, T1-T8 és T1-T5 szenzorok összegejeleiből lettek képezve. Látható, hogy minden csatorna jele a jármű sávjában mérhető összegjellel lett leosztva, így megkapva a normalizált nyúlásértékeket. Így a jármű sávjában mérhető összegjel terjedelme az ablakon belül 1 (piros kettős nyíl jelzi). Az eseten az is látszik, hogy egy kétsávós eset mennyire bonyolult tud lenni. Ebben az esetben sávonként 2-2 csatorna van, tehát a csatornaszám (CH) ebben az esetben 4, a sávszám (L) pedig 2. A sávonkénti maszk is látszódik az ábrán kék (cyan) színnel. Ahol a háttér színes, ott a maszk értéke 0, különben a maszk értéke 1. Ahol a maszk értéke 1, ott kirajzolódik egy jármű áthaladása az SVK→HU sávban.

A konvolúciós neurális hálózat a nyúlásadatok tenzorát és a maszk tenzorát összekonkatenálva, egy $B \times (CH + L) \times T$ méretű tenzorként kapja meg, amin a megfelelő transzformációk végrehajtnak. A többi lépés innentől kezdve feladat-specifikus, ezek a továbbiakban kerülnek kifejtésre. A konvolúciós neurális hálózatainkban használtunk reziduális kapcsolatokat [36].

A neurális hálók tanítása során használtunk adataugmentációt is [6]. A jelekhez Gauss-zajt kevertünk hozzá minden esetben különböző mértékben. Az augmentációs algoritmusnak egy σ_{max} érték volt megadva. A zaj szórása a $[0, \sigma_{max}]$ intervallumból lett egyenletesen mintavételezve minden bemenet esetén. Így a neurális hálózat látott közel tiszta és zajos jelet is a tanítás során.

4.1. Konvolúciós neurális hálózat alapú sebességbecslés

Ebben az alfejezetben bemutatásra kerül egy újszerű sebességbecslő konvolúciós neurális hálózat. Először a megoldás újszerűsége kerül bemutatásra, majd a sablon specifikálása történik meg erre a modulra.



4.2. ábra. A grafikonokon a neurális háló egy példa bemenete van megadva sávonkénti bontásban. A normalizált nyúlásértékek az idő függvényében vannak megadva. A maszk is kiolvasható a kék (cyan) háttérű részek segítségével. Mind a két sávhoz tartozik egy-egy maszk, ahol a háttér kék színű, ott a maszk 0 értéket vesz fel, a többi helyen pedig 1 értéket. Így látható, hogy a vizsgált jármű az SVK→HU sávban közlekedett.

4.1.1. Motiváció

A 3.2.2 fejezetben már említésre került, hogy a sebességbecslés akár egy keresztmetszet szenzorai alapján is lehetséges [3, 14]. Az a megoldás 3,23 km/h átlagos hibával állapította meg a járművek sebességét, aminél mi jobb eredményeket kívánunk elérni. Mi erre a megoldásra építve egy újszerű konvolúciós neurális hálózatot használó modult fejlesztettünk ki, ami képes lehet felvenni a versenyt azokkal a megoldásokkal, amik két keresztmetszeti szenzort is használnak. Mivel csak a sebességbecslés miatt van szükség második támogató szenzorsor telepítésére, ezért egy ilyen megoldás ki tudná váltani annak telepítését, ami így az alacsonyabb szenzorszám miatt alacsonyabb telepítési költségeket is eredményezne.

4.1.2. Architektúra

A konvolúciós neurális hálózat bemenete már bemutatásra került. A hálózat ezek után egy hagyományos konvolúciós neurális hálózatba kerül, ami konvolúciós blokkokból épül fel. A konvolúciós háló kimenete egy $B \times L \times T$ méretű tenzor lesz, ezen a következő utólagos transzformációt kell még végrehajtani, hogy megkapjuk a neurális hálózat B méretű kimenetét:

$$\text{Speed}_i = \frac{\sum_{l=1}^L \sum_{t=1}^T \text{Mask}[i, l, t] * \text{OutputTensor}[i, l, t]}{\sum_{l=1}^L \sum_{t=1}^T \text{Mask}[i, l, t]} \quad (4.1)$$

Látható, hogy lényegében a maszk (Mask) által látható részen belül történik egyfajta átlagolása a konvolúciós neurális háló kimenetének (OutputTensor), és így kapjuk meg a sebességet (Speed). Ez az átlag lesz a neurális hálózat kimenete, lényegében egy-egy skálár érték minden járműre. Ez a kimenet adja meg a sebességet. Ezek után a postprocess lépésben általában csak arra van szükség, hogy a neurális hálózat kimenetét a megfelelő mértékegységre átváltssuk, mivel esetünkben a neurális hálózat kimenete m/s mértékegységben adja meg a sebességet. A neurális hálózatban a nemlinearitás Leaky ReLU függvény volt [37].

4.1.3. Tanítás

Az eddigiek során bemutattuk a háló architektúráját. Ebben a szakaszban bemutatásra kerül a háló tanítása. A tanítás során az egyik legfontosabb kérdés, hogy milyen veszteségfüggvényt használunk. Az L_1 hibára esett a választás:

$$L_1(\hat{y}, y) = \frac{1}{B} \sum_{i=1}^B |\hat{y}_i - y_i| \quad (4.2)$$

A képletben egy batch adat kiértékelése látszik: $\hat{y}_i$ jelöli az i . jármű esetén a számított sebességet, y_i pedig az elvárt sebességértéket.

Az L_1 hiba mellett még az Symmetric Mean Absolute Percentage Error (SMAPE) hibafüggvénnyel kísérleteztünk, azonban ezzel nem sikerült jobb eredményeket elérni. Az SMAPE hibafüggvény a következő:

$$\text{SMAPE}(\hat{y}, y) = \frac{1}{B} \sum_{i=1}^B \frac{|\hat{y}_i - y_i|}{\frac{|\hat{y}_i| + |y_i|}{2}} \quad (4.3)$$

Még érdemes megjegyezni, hogy a [3] cikkben Mean Squared Error (MSE) veszteségfüggvénnyel történt a tanítás, azonban mi nem emellett döntöttünk, mivel az az outlier értékekre nagyon érzékeny.

A tanítás során Adam optimizert és cosinus alapú tanulási ráta ütemezőt is használtunk [38], hogy jobb eredményeket érjünk el. Az eredmények a későbbiek során az 5.1 fejezetben kerülnek kifejtésre.

4.2. Konvolúciós neurális hálózat alapú tengelydetektálás

Ebben a fejezetben az általunk fejlesztett tengelydetektáló konvolúciós neurális hálózat kerül bemutatásra, amely a sebességbecslőhöz hasonlóan szintén egy külön modul a BWIM rendszerekben. A probléma rövid ismertetését követően az architektúra bemutatása következik, majd a hálózat tanítása során tapasztaltokról esik szó.

4.2.1. Motiváció

A tengelydetektálás vonatkozhat a hídon áthaladó jármű tengelyeinek számának meghatározására vagy ezen tengelyek időbeli elhelyezkedésére is. Természetesen az utóbb említett folyamat komplexebb, azonban ennek helyes implementálása maga után vonja a tengelyszámok meghatározását is.

A 3.2.3 szakaszban már említésre került több megközelítés, melyek a szenzorok által mért jelek feldolgozását segítik elő. Célunk az volt, hogy a megoldásunk vasbeton és

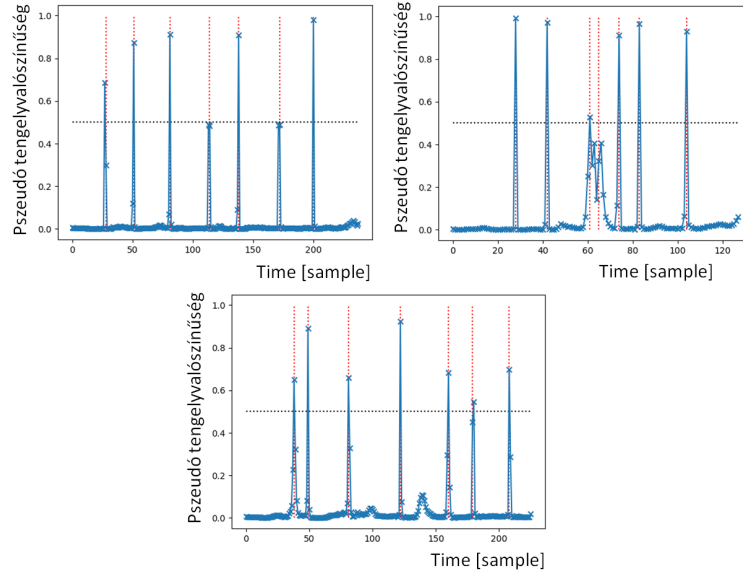
ortotróp acélpályás hidakon is egyaránt alkalmazható legyen, így a hidakon található szenzorok különböző szenzorainak jelei összegezve kerültek feldolgozásra. Emellett célunk volt, hogy a megoldásunk több sávú hidak esetén, tehát komplexebb forgalmi szituációkban is működjön. A dolgozat készítése során a "Virtual Axle Detector Based on Analysis of Bridge Acceleration Measurements by Fully Convolutional Network" [4] cikket vettük alapul és az ott bemutatott megoldást fejlesztettük tovább. Abban a cikkben egy vasúti hídra fejlesztettek egy neurális hálózat alapú megoldást. Ebben az esetben nem nyúlásmérő, hanem gyorsulásmérő szenzorokat használtak. A neurális hálózatnak pedig a jelek wavelet-transzformáltjait adták be bemenetként. Mi ezt a megoldást kívánjuk átültetni, és továbbfejleszteni nyúlásmérő szenzorok esetére, és általánosságban a jármű forgalmat lebonyolító különféle acélpályás és vasbeton szerkezetű hidakra.

4.2.2. Architektúra

A tengelydetektáló neurális hálózat architektúrája nagy részben megegyezik a 4.1.2. fejezetben kifejtett sebességbecslő felépítésével. Ebben az esetben azonban egy osztályozási problémáról beszélhetünk, a modul kimenetként minden egyes mintavételre szeretnénk megkapni, hogy abban a mintavételben haladt-e át tengely, így ez lényegében egy bináris osztályozási probléma. Emellett vannak különbségek a maszk felhasználásban. A maszkot az ablakdetektáló modul biztosítja a tengelydetektáló neurális hálózat számára, mely megadja a jel releváns részét, hogy melyik sávban mikor haladt el egy adott gépjármű a szenzorsor fölött. A maszk segítségével a neurális hálózat kimenetéből kivágjuk a maszk által definiált részt, így kikényszerítve, hogy a tengelyeket a gépjármű ablakokon belül kell keresni, hiszen az ablakok között nem fordulhatnak elő tengelyek. Az adatok előfeldolgozásánál ennél a modulnál is lehetőség van a jel zajosítására, így felkészítve a valós esetekre. Zajosításra a szimulált jelek esetében van szükség (hiszen a szimulált jel nem tükrözi teljes mértékben a valóságot), zajmentesítés pedig a valós jelek esetében lehet fontos, hogy kiszűrjünk belőle zavaró tényezőket. A neurális háló kimenetét a sigmoid aktivációs függvény segítségével a $(0, 1)$ intervallumba kényszerítjük. A neurális hálón kívül szükséges még egy utófeldolgozási (postprocess) fázison is átmennie. A prediktált tömb elemeinek 0, illetve 1 értékekre való transzformációjára egy egyszerű kerekítés most nem elég, ezt mutatja be a 4.3 ábra. Az ábrán látható, hogy egy-egy jármű esetén a neurális háló kimenete milyen valószínűségeket (pszeudó tengelyvalószínűség) ad arra, hogy egy-egy mintavétel során haladt-e át tengely. Mi a [4] cikkhez hasonlóan ezekből a sorozatokból lokális maximum alapú detekcióval találtuk meg a tengelyek pozícióját. Ehhez a SciPy könyvtár find-peaks módszerét használtuk. A prediktált valószínűségek alapján kizárólag olyan mintavételeket osztályoztunk 1-esre, ami nagyobb egy általunk megadott küszöbértéknél, a pozícióban lokális maximum található, és az általunk megadott környezetében (2-3 mintavételnyi távolság) nem található másik releváns lokális maximum. Amennyiben több lokális maximum is egymáshoz közel helyezkedik el, akkor ezek közül a maximálisat választjuk ki. Természetesen a megmaradt értékek osztályait 0 értékre állítottuk a megoldás során.

4.2.3. Tanítás

A tengelyek lokalizálását osztályozási problémaként definiáltuk. A szenzorok által mért minden mintát a következő osztályok egyikével jelöltük meg: ment át a mintavételi időpontban tengely a szenzorsor felett (1. osztály), vagy nem ment át (0. osztály). Így a neurális háló tanítása során használt címkék két dimenziós tömbök voltak, ahol a dimenziók a két sávot reprezentálják, a tartalmazó értékek (0 és 1) pedig megmutatják, hogy az adott gépjármű tengelyei mely időpillanatokban haladtak el a szenzorsor fölött.



4.3. ábra. Egy-egy 7 tengelyes jármű esetén látható mintavételként a konvolúciós neurális háló által számított valószínűség, hogy abban a mintavételben lehet-e tengely. A piros függőleges pontozott egyenesek a tengelyek elvárt helyeit mutatják, míg a vízszintes fekete pontozott vonal a 0,5 értéket mutatja.

Hosszú jelek, de akár egy rövid jel kapcsán is már jóval nagyobb arányban vannak 0 címkéjű mintavételek, mint 1 címkéjű mintavételek. Felmerül a kérdés, hogy az ilyen esetekben az osztályegyensúly hogyan kezelhető. Az osztályegyensúly gyakori probléma sok bináris osztályozási feladatban, ahol az egyik osztály jelentősen nagyobb arányban van jelen a másikhoz képest. A BCE (Binary Cross Entropy) gyakran a többségi osztály felé hajlik, ami a teljesítmény romlásához vezethet. Például, ha a modell minden értéket 0-nak jósol (és nem tudja megtalálni a tengelyeket), akkor BCE-vel közel tökéletes veszteség érhető el, ezáltal a modell megtanulja figyelmen kívül hagyni a tengelyeket. A Focal Loss [4] ezt a problémát úgy kezeli, hogy a jól osztályozott példák veszteségét csökkenti, lehetővé téve a modell számára, hogy a nehezen osztályozható példákra koncentráljon. A Focal Loss képlete a 4.4 képletben látható. Fontos megjegyezni, hogy a jobb eredmények elérése érdekében kísérletezni kell α és γ hiperparaméterek különböző értékeivel, és figyelni a modell teljesítményét a finomhangolás során.

$$FL(p) = \begin{cases} -\alpha(1-p)^\gamma \log(p), & \text{if } y = 1 \\ -\alpha p^\gamma \log(1-p), & \text{otherwise} \end{cases} \quad (4.4)$$

Az α finomhangolása segíthet az osztályegyensúly kezelésében, mivel megfelelő súlyokat rendelhet különböző osztályokhoz. Például a kisebbségi osztály esetében magasabb értékeket lehet megadni α -nak, hogy nagyobb fontosságot tulajdonítson neki a tanítás során.

A γ választása szignifikánsan befolyásolja a veszteségfüggvény viselkedését. Magasabb γ értékek kiemelik a nehéz példák helyes osztályozását, míg alacsonyabb értékek a veszteségfüggvényt hasonlóvá teszik BCE veszteségfüggvényhez. A γ optimális értéke a probléma konkrét jellegétől függően változhat. A mi esetünkben $\alpha = 1/2$ és $\gamma = 2$ paraméter működött ugyanúgy, mint az alapul vett [4] cikkben.

Fontos megjegyezni, hogy a Focal Loss és a BCE közötti választás a konkrét problémától és az adathalmaztól függ. A Focal Loss nem mindig teljesít jobban, mint a BCE, különösen akkor, ha az osztályegyensúly nem jelentős probléma, vagy ha egy kiegyensúlyozott adathalmaz áll rendelkezésre. Bizonyos esetekben a BCE is elegendő lehet.

A kulcs az, hogy mindkét veszteségfüggvénnyel érdemes kísérletezni, és végül azt választani, amely a konkrét problémához a legjobban működik, mivel a veszteségfüggvény hatékonysága változhat az adateloszlás és a feladat jellemzőitől függően. Ezenkívül más tényezők, mint például a hálózati architektúra és a hiperparaméterek is meghatározzák egy adott modell általános teljesítményét.

A tanítás során ebben az esetben is Adam optimizert és cosinus alapú tanulási ráta ütemezőt is használtunk [15], hogy jobb eredményeket érjünk el. Az eredmények a későbbiek során az 5.2 fejezetben kerülnek bemutatásra.

5. fejezet

Elért eredmények

Ebben a fejezetben a két modult kiértékeljük különböző szintetikus és valós adatsorokon. Az elért eredményeket az irodalomban korábban már említett módszerekkel is összehasonlítjuk.

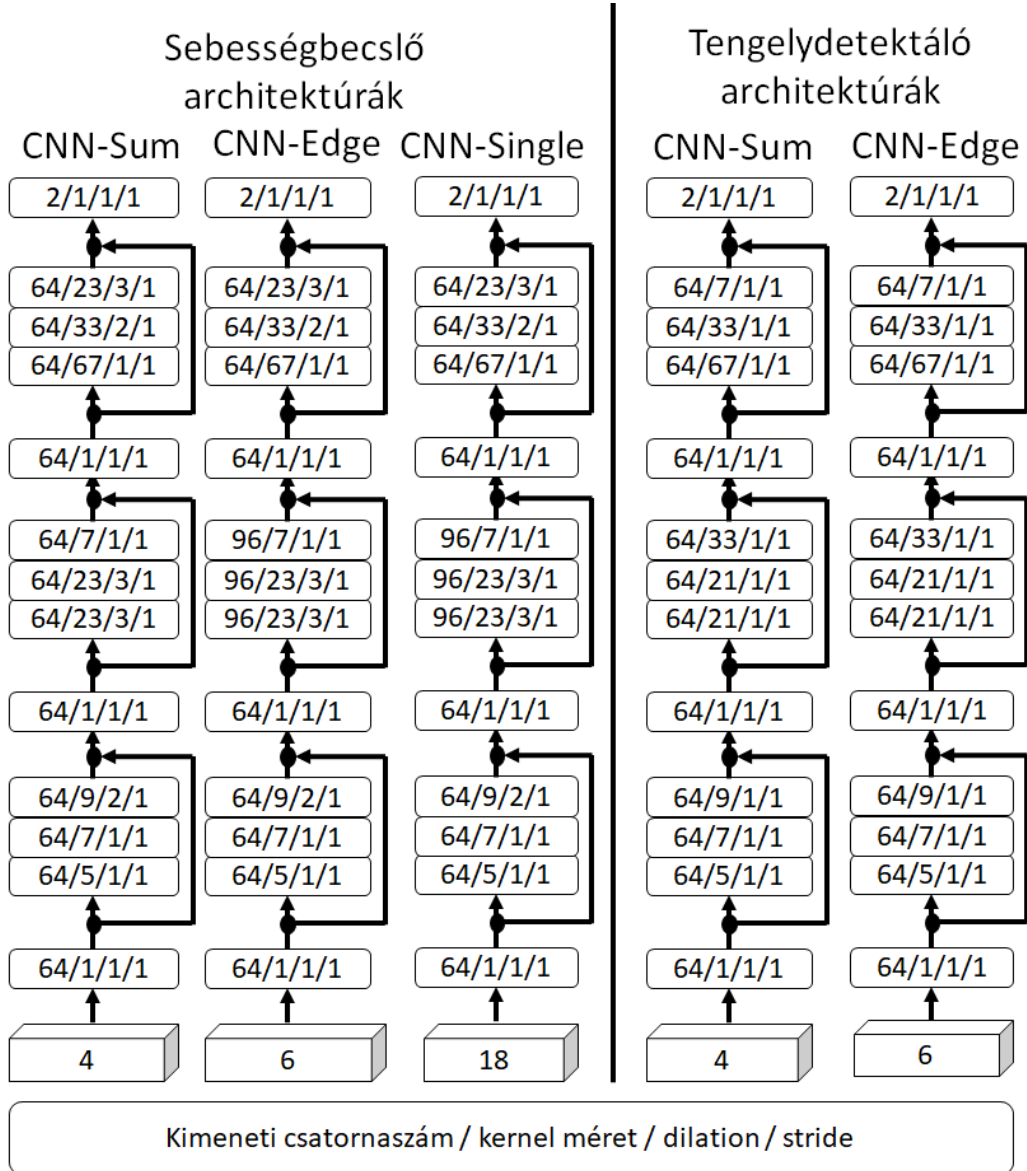
A 5.1 ábra mutatja be az egyes neurális hálózatok architektúráját. A CNN-Sum megoldások bemenetként a sávonkénti összegjelet és a maszkot kapták meg összekonkatenálva bemenetként, tehát egy 4 csatornaszámú tenzort. A CNN-Edge megoldások a sávonkénti összegjelet, és a sávok szélén lévő 5-5 szenzor jelét kapták meg bemenetként, ami így egy 6 méretű tenzort eredményez a maszkkal együtt. A CNN-Single megoldások minden egyes szenzor jelét megkapták, így egy 18 csatornaszámú tenzort kapva bemenetként. A CNN-Sum és CNN-Edge megoldások vasbeton szerkezetű hidakon is működőképesek lehetnek, míg a CNN-Single csak acélpályával rendelkező hidakon. Éppen ezért a tengelydetektálás esetén a CNN-Single esetet nem is vizsgáltuk.

5.1. Sebességbecslés

A sebességbecslés esetében a megoldásunkat egy hagyományosnak mondható lokális maximum alapú megoldással hasonlítjuk össze, valamint a korábban említett hasonló egy keresztmetszet alapján számító módszerrel. A lokális maximum alapú sebességbecslőt mi is implementáltuk, ennek egy vázlatos bemutatása következik, majd az eredmények összehasonlítása.

5.1.1. Lokális maximum alapú sebességbecslés

Az általunk implementált lokális maximum alapú sebességbecslés a következő módon működik: feltételezzük, hogy korábban a jármű ablaka a fő szenzorsorban már kiszámításra került. A fő szenzorsor által szolgáltatott jelekből ezt a $[w_{begin}, w_{end}]$ időintervallumot kimetszük. A rendszer feltételezi, hogy a jármű sebessége a $[v_{min}, v_{max}]$ határokon belül van, esetünkben $v_{min} = 10$ m/s és $v_{max} = 30$ m/s, a fő szenzorsor és a támogató szenzorsor előjeles távolsága $L = \pm 6m$, az előjel a jármű haladási irányától függ. A támogató szenzorsor által szolgáltatott jelekből a $[min(w_{begin} + L/v_{min}, w_{begin} + L/v_{max}), max(w_{end} + L/v_{min}, w_{end} + L/v_{max})]$ időintervallumot metszük ki. Ezek után mindkét keresztmetszetből kimetszett jelben a lokális maximumokat kell detektálni, ami után ezeket a két keresztmetszetben össze kell egyeztetni. Legyen a fő keresztmetszetben talált n darab lokális maximum $V = v_1, v_2, \dots, v_n$, és jelölje $t(v_i)$ a v_i lokális maximum időbélyegét. Hasonlóan jelölje $U = u_1, u_2, \dots, u_m$ a támogató keresztmetszetben számított m darab lokális maximumot. Hasonlóan $t(u_j)$ jelölje az u_j lokális maximum időbélyegét. Az összeegyeztetés úgy működik, hogy lehetséges Δt időeltolásokot próbálunk ki. Egy lehetséges Δt időeltolás esetén egy $G_{\Delta t} = (V, U, E)$ páros gráfot kreálunk, ahol V és U a csúcsok két osztálya, és



5.1. ábra. A sebességbecslő neurális hálózatok, illetve a tengelydetektáló neurális hálózatok felépítése.

E az élek halmaza. Ebben a páros gráfban egy u_i és v_j csúcs akkor van összekötve, ha a $|t(v_i) + (t(u_j) - \Delta t)|$ különbség kisebb, mint egy megadott küszöbérték. Ez a küszöbérték általában 1-2 mintavételyi eltérést jelent, ami esetünkben kb. 1-2 századmásodpercet jelent. Ebben a $G_{\Delta t}$ gráfban jelölje $M_{A,G_{\Delta t}}$ az A algoritmus által talált párosítást a gráfban. A feladat így a következő módon írható át:

$$\max_{\Delta t} |M_{A,G_{\Delta t}}|, \text{ ahol } \Delta t \in [L/v_{max}, L/v_{min}] \quad (5.1)$$

Így lényegében egy párosítási problémára vezetjük vissza a problémát. Az A algoritmus többféle lehet. Akár lehetne az Egerváry-algoritmus is, azonban mi egy mohó algoritmust használunk, ami az Egerváry-algoritmusnál egyszerűbb és gyorsabb, miközben nem ad jelentősen rosszabb eredményeket. Egy esetben akár több lehetséges Δt érték esetén is felveheti az optimalizálni kívánt kifejezés a maximumát, ekkor ezen Δt értékek átlaga

lehet a számított időeltolás, amiből már a sebesség kiszámítható. Emellett akár az élek súlyozásával, és így a feladat átfogalmazásával is élhetünk a feladat megoldása céljából.

5.1.2. Eredmények összehasonlítása

Metrika	Rövidítés	Számítási mód
Mean Absolute Error	MAE	$\frac{1}{n} \sum_{i=1}^n \hat{y}_i - y_i $
Root Mean Square Error	RMSE	$\sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$
Standard deviation	σ	$\sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - \mu)^2}$, ahol $\mu = \frac{1}{n} \sum_{i=1}^n \hat{y}_i$
Mean Absolute Percentage Error	MAPE	$\frac{1}{n} \sum_{i=1}^n \left \frac{\hat{y}_i - y_i}{y_i} \right $
Mean Signed Percentage Error	MSPE	$\frac{1}{n} \sum_{i=1}^n \frac{\hat{y}_i - y_i}{y_i}$

5.1. táblázat. Az egyes metrikák, rövidítéseik, és számítási módjaik, ahol $\hat{y}$ és y egy-egy n hosszú vektor, ahol $\hat{y}_i$ és y_i az i . jármű számított illetve elvárt sebességét jelöli.

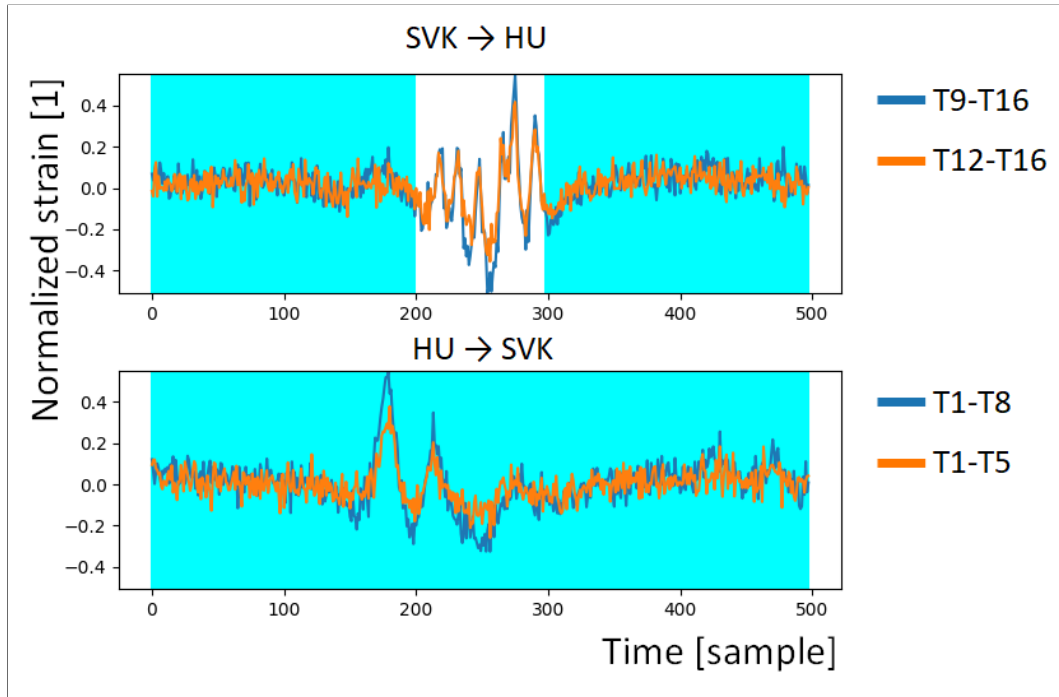
Három különböző architektúrával dolgoztunk, ezek a CNN-Sum, CNN-Edge és a CNN-Single. Ahogy korábban már bemutattuk, a CNN-Sum a sávonkénti összegejelek alapján dolgozik, a CNN-Edge a sávonkénti összegjelek, és a sávok külső szélén lévő szenzorok összegjelei alapján, míg a CNN-Single a szenzorok jeleit egyesével kapja meg.

Az 5.2 táblázat mutatja be, hogy az egyes megoldások milyen eredményeket értek el a BME-S1 adathalmaz tisztán szimulált jelein. Az egyes metrikákat a 5.1 táblázat mutatja be. Az LM-Edge az előbb bemutatott lokális maximum alapú sebességbecslő egy implementációja, ami csak a jármű sávjának szélén található 5 szenzor adatainak összegjelét használja. Az eredményekből látszik, hogy a két keresztmetszet szenzorait használó megoldás jobb MAE értéket ad, de a konvolúciós neurális háló alapú sebességbecslők még így is tolerálható hibával (1,34-1,54 km/h átlagos hiba) képesek megmondani a sebességet.

Metrika	Sebességbecslő			
	CNN-Sum	CNN-Edge	CNN-Single	LM-Edge
MAE [km/h]	1.54	1.36	1.34	0.84
RMSE [km/h]	2.61	2.30	2.15	2.80
σ [km/h]	2.60	2.29	2.11	2.80
MAPE [%]	2.20	1.91	1.95	1.15
MSPE [%]	0.43	-0.10	-0.69	0.24

5.2. táblázat. A különböző sebességbecslő megoldások hibametrikái a BME-S1 adathalmazon, amikor nem lett zaj hozzáadva a jelekhez.

Az 5.3 táblázat azt mutatja be, hogy a jelek terjedelméhez viszonyított 5% szórású zaj milyen szinten rontja a sebességbecslő hálózat metrikáit a CNN-Edge esetben. A táblázatból kiolvasható, hogy emiatt az átlagos hiba (MAE) kb. a harmadával nőtt meg, ami konkrétan most 0,42 km/h különbséget jelent. A sebességbecslő még ebben az esetben is tolerálható hibával képes általában megmondani a sebességet. Itt az 5% szórású zaj azt



5.2. ábra. A grafikon azt mutatja be, hogy az 5% szórású zaj esetén hogyan változott meg a neurális háló bemenete. A 4.2 ábrán ugyanez az eset van ábrázolva zaj nélkül.

jelenti, hogy a korábban a neurális háló normalizált bemenetének minden csatornájához egy 0,05 szórású zajt adtunk hozzá. A 5.2 ábra bemutatja, hogy az 5% szórású zaj hogyan változtatja meg a bemenetet.

CNN-Edge			
Metrika	Adathalmaz		Különbség
	Tiszta jel	Zajos jel	
MAE [km/h]	1.36	1.78	0.42
RMSE [km/h]	2.30	2.68	0.38
MAPE [%]	1.91	2.51	0.60

5.3. táblázat. A CNN-Edge sebességbecslő hibametrikái a BME-S1 adathalmazon, amikor a jelekhez zaj lett hozzáadva.

A 5.4 táblázat a CNN-Edge sebességbecslő hibametrikáit mutatja BME-S1 részadathalmazaira lebontva. Az egysávos (OL), kétsávos (BL) esetben is az egyes részadathalmazokra le lett mérve a modul pontossága. Ezek mellett a kétsávos esetben egy olyan mérést is végeztünk, amikor csak a legnehezebb átlagos tengelysúlyú jármű esetén értékeltük ki a kapott adatokat. Ezt az esetet "többsávos - nehéz járművek" (Both-Lane-Heavy, BL-H) esetnek neveztük el. Az eredményekből látszik, hogy az egysávos esetben a sebességbecslő pontosabb, mint kétsávos esetben. Az egysávos esetben mért 1 km/h átlagos hiba alacsonynak mondható. A kétsávos esetekben látszik, hogy a nehezebb járművek sebességét pontosabban lehet becsülni. Ezt is várnánk, hiszen a nehéz járművek által gerjesztett jelet a másik sávban haladó könnyebb jármű nehezebben zavarja. Ez azért is fontos, mivel a híd élettartamának és fáradásának előrejelzése szempontjából a nehéz tengelyterhelésű és összsúlyú járművek érdekesek, mivel a fáradás nagy részét ezek okozzák.

CNN-Edge			
Metrika	Adathalmaz		
	Egysávos	Többsávos	Többsávos - nehéz járművek
MAE [km/h]	0.66	2.05	1.38
RMSE [km/h]	0.88	3.12	1.94
σ [km/h]	0.85	3.12	1.93
MAPE [%]	0.93	2.89	1.92

5.4. táblázat. Különböző sebességbecslő megoldások hibametrikái a BME-S1 adathalmazon, tiszta jeleken.

Az 5.5 táblázat azt mutatja be, hogy a már korábban bemutatott kis méretű valós adathalmazon milyen eredményeket sikerült elérni. Minden esetben a jármű elvárt sebessége 50 km/h volt. Nyilván az 50 km/h tartása és helyszíni mérése már önmagában vitt valamennyi hibát a mérésbe, nem olyan tiszta adatokkal dolgozunk, mint a szintetikus adathalmaz esetében. Minden esetben csak az egyik sávban ment jármű, így azt lehet mondani, hogy a mért átlagos hibák magasabbak a vártnál minden sebességbecslő esetében, viszont a konvolúciós neurális háló hibája és a lokális maximum alapú becslő hibájának az aránya ebben az esetben is hasonló, mint a szintetikus adathalmazon.

Metrika	Sebességbecslő			
	CNN-Sum	CNN-Edge	CNN-Single	LM-Edge
MAE [km/h]	0.96	1.29	0.90	0.69
RMSE [km/h]	1.15	1.64	1.11	0.80
MAPE [%]	1.91	2.57	1.79	1.38

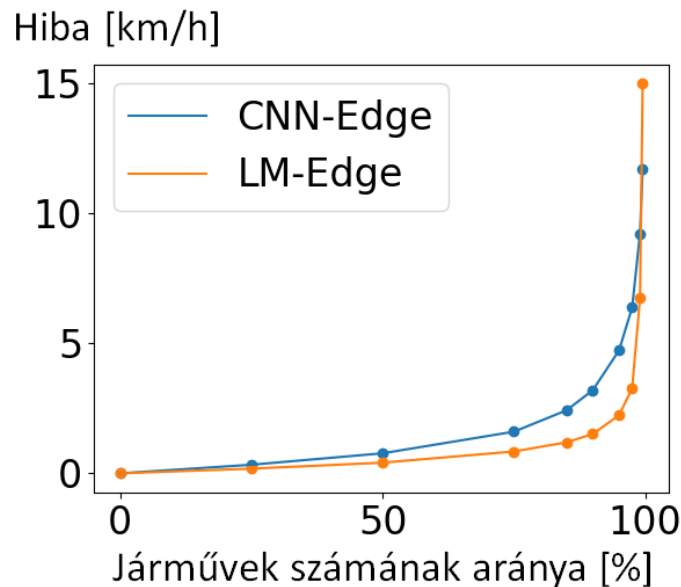
5.5. táblázat. A valós annotált adathalmazon elért hibametrikák különböző sebességbecslőkkel.

Az 5.3 ábra és az 5.6 táblázat azt mutatja be, hogy a járművek hány százalékának sebességét sikerült egy bizonyos hibahatár alatt megbecsülni a két sebességbecslő (CNN-Sum, LM-Edge) esetében. Látható, hogy a lokális maximum alapú megoldás jobban becsüli meg az esetek nagy többségében a sebességet, mint a konvolúciós neurális hálózat, azonban a kiugró értékek ebben az esetben magasabbak. Ezek a magas kiugró értékek okozzák a magas RMSE értékeket a lokális maximum alapú becslőnél a kiértékeléskor.

Bár a lokális maximum alapú sebességbecslővel szemben alulmarad a megoldásunk, szinte az összes esetben azt mondhatjuk, hogy a hiba még így is tolerálható, egy ilyen sebességbecslő modul integrálása a BWIM-rendszerbe nem teszi azt használhatatlanná, csak pontatlanabbá, miközben ez a megoldás csak egy keresztmetszet szenzorait használja. A lokális maximum alapú sebességbecslő a komáromi híd esetében 10 szenzorral többet használ, mint a neurális háló alapú megoldás, ami több, mint másfélszer több szenzort jelent. A megoldásunk a korábban már említett Kawatsaku és társai által alkotott megoldáshoz képest jobban teljesít. Az 3,23 km/h átlagos hibával működött, míg a mi megoldásunk 2,05 km/h átlagos hibával egy zajos szintetikus adathalmazon, miközben az RMSE értékek 6,27 km/h és 3,12 km/h voltak. Itt meg kell említenünk, hogy ezek az eredmények nehezen összehasonlíthatóak, mivel nem veszi figyelembe a hidak különböző szerkezetét, a különböző forgalmi szituációkat és az adathalmaz felvételének különbözőségét, de az, hogy a zajos adathalmazon és a valós annotált adathalmazon is jelentősen jobb az eredm-

Percentilis [%]	Sebességbecslő hiba [km/h]	
	CNN-Edge	LM-Edge
25	0.33	0.18
50	0.77	0.41
75	1.60	0.84
85	2.42	1.19
90	3.19	1.51
95	4.72	2.23
97.5	6.37	3.27
99.0	9.21	6.73
99.5	11.70	14.96

5.6. táblázat. A különböző sebességbecslő megoldások percentiliseihez tartozó hibaértékek. Tehát azt adja meg a táblázat, hogy a járművek hány százalékának hibája volt egy bizonyos érték alatt km/h mértékegységben.



5.3. ábra. A grafikon azt mutatja be, hogy a járművek hány százalékának sebességét sikerült egy bizonyos hibahatár alatt megbecsülni a két sebességbecslő esetében.

nyeink, arra enged következtetni, hogy a megoldásunk a [3] cikk által vizsgált hídon sem teljesítene jelentősen rosszabb módon. A megoldás akár egy lokális maximum alapú sebességbecslő megoldás mellett is jól működhet, támogatva azt. Amennyiben két módszer is közel azonos sebességet számít ki egy járműre, akkor nagyobb bizonyossággal mondhatjuk, hogy jól sikerült kiszámítani azt. Mivel a sebességbecslő főleg a nehezebb járművek esetén pontosabban mér, ezért a híd fáradásának előrejelzése szempontjából lényeges adatokat pontosabban mér. A híd fáradásának szempontjából az áthaladó járművek nagy száma alapján számítható statisztikai mértékek a lényegesek, amik esetében kis arányú hibás eset nem okoz jelentős hibát a további számításokban.

5.2. Tengelydetektálás

A tengelydetektálás esetében a neurális hálózatunk teljesítményét egy hagyományosnak mondható lokális maximum alapú megoldással hasonlítjuk össze. A lokális maximum alapú tengelydetektálót mi is implementáltuk. Emellett létrehoztunk egy algoritmust, ami ennek az elméleti korlátait mutatja be. Először ennek bemutatása következik, majd az eredmények összehasonlítása.

5.2.1. Lokális maximum alapú tengelydetektálás korlátai

A lokális maximum alapú tengelydetektálás optimálisnak tekinthető, amennyiben minden tengelyt megtalál ami lokális maximum helyen van, és minden mást ignorál, tehát olyan lokális maximumokat nem vesz bele a tengelyek közé ami nem valós tengely, így a FP (False Positive) száma 0 és a Precision 100% egy optimális esetben. Számunkra az így elért mérések eredményei azt mutatják, hogy egy optimálisan megírt lokális maximum alapú tengelydetektáló ennél jobb eredményeket nem tud elérni az adathalmazunkon. Implementáltunk több, a lokális maximum alapú tengelydetektáló korlátait bemutató algoritmust, ami a következő módon működik: feltételezzük, hogy korábban a jármű ablaka a fő szenzorsorban már kiszámításra került. A fő szenzorsor által szolgáltatott jelekből három különböző megközelítést vizsgálunk. Első esetben minden szenzor jelét figyelembe vesszük a maximum detektálásnál, tehát minden egyes jelen futtatunk egy maximum detektálást (Opt-LM). A második esetben a szenzorok jeleit összegezzük és ezután végezzük el a maximum keresést (Opt-LM-Sum). A harmadik esetben felhasználjuk minden szenzor összegzett jelét, illetve azon sáv szélső öt szenzorának összegzett jelét, ahol az adott gépjármű éppen halad (Opt-LM-Edge). Amennyiben az elvárt tengelypozíciók (illetve ± 1 mintavétel tartományában) lokális maximumot talál a megoldásunk, az adott tengelyt detektálhatónak tekintjük. Egy ilyen algoritmus megmutatja, hogy mi a detektálható tengelyek maximális száma teoretikusan. Egy valós lokális maximum alapú tengelydetektáló nem nyújthat ennél jobb eredményt, mert az azt jelentené, hogy olyan tengelyt tud detektálni, aminek az időpontjában nem mérhető lokális maximum, ez pedig nyilván ellentmondás lenne.

5.2.2. Eredmények összehasonlítása

Metrika	Számítási mód
Precision	$\frac{\#TP}{\#TP + \#FP}$
Recall	$\frac{\#TP}{\#TP + \#FN}$
F1-score	$\frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

5.7. táblázat. Az egyes metrikák, rövidítéseik, és számítási módjaik.

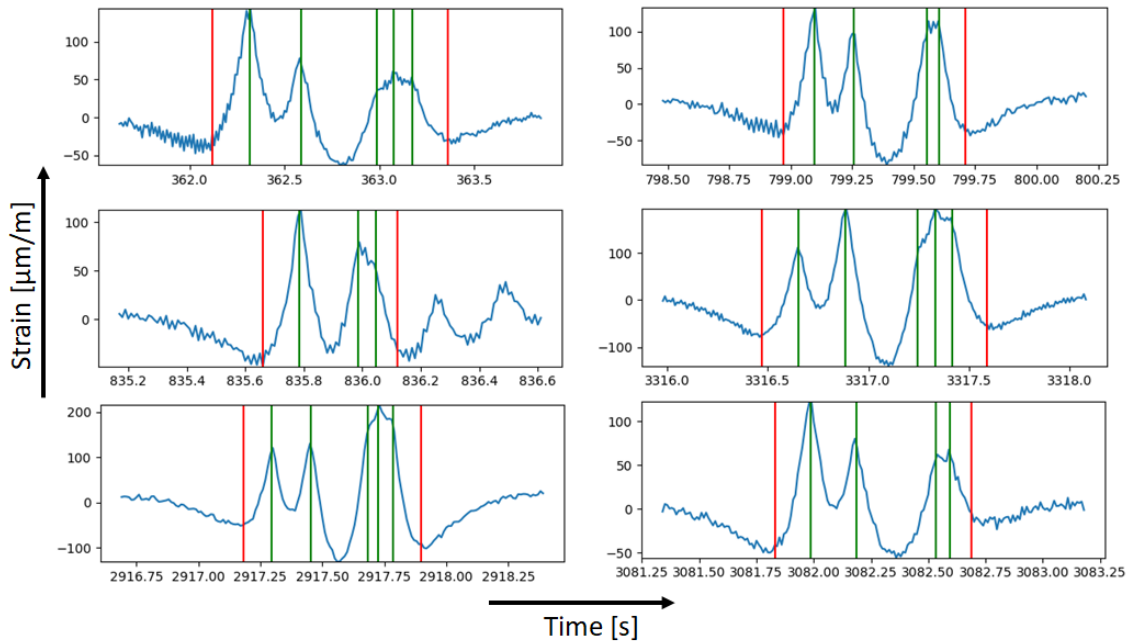
Ahhoz, hogy a szakaszban bemutatott metrikák számítása érthetőbb legyen fontos tisztázni True Positive (TP), False Negative (FN), False Positive (FP), True Negative (TN) értékek jelentését. A TP azt jelenti, hogy az elvárt kimenet 1 volt és a neurális hálózat is 1-et prediktált. A FP azt jelenti, hogy az elvárt kimenet 0 volt, de a neurális hálózat 1-et prediktált. A FN azt jelenti, hogy az elvárt kimenet 1 volt, azonban a neurális hálózat 0-át prediktált. A TN pedig azt jelenti, hogy az elvárt kimenet 0 volt és a neurális

Metrika	Tengelydetektáló				
	Opt-LM	Opt-LM-Sum	Opt-LM-Edge	CNN-Sum	CNN-Edge
Recall [%]	95.66	83.02	84.43	99.04	99.57
F1-score [%]	97.78	90.72	91.56	99.27	99.08

5.8. táblázat. A különböző tengelydetektáló megoldások hibametriái a BME-S1 adathalmazon, amikor nem lett zaj hozzáadva a jelekhez.

CNN-Edge			
Metrika	Adathalmaz		Különbség
	Tiszta jel	Zajos jel	
MAE [mm]	53	72	19
RMSE [mm]	66	92	26
Precision [%]	98.60	97.16	1.44
Recall [%]	99.57	96.59	2.98
F1-score [%]	99.08	96.87	2.21

5.9. táblázat. A CNN-Edge tengelydetektáló neurális hálózat hibametriái a BME-S1 adathalmazon, amikor a jelekhez zaj lett hozzáadva.



5.4. ábra. Az egyes grafikonok olyan valós mérési eredményekből kinyert eseteket mutatnak, ahol a tengelydetektáló algoritmusunk nélkül nem sikerült volna az összegjelekből kinyerni a tengelyek pozícióit helyesen.

hálózat is 0-át prediktált. A #TP, #TN, #FP és #FN értékek az egyes csoportokba tartozó esetek számát adja meg.

A tengelydetektáló esetében két különböző architektúrát próbáltunk ki, ezek pedig a CNN-Sum és CNN-Edge elnevezésűek voltak. Ahogyan már korábban is említésre került a sebességbecslő modulnál a CNN-Sum a sávonkénti összegjelek alapján dolgozik, a CNN-

Edge pedig a sávonkénti összegjelek, és a sávok külső szélén lévő szenzorok összegjelei alapján végzi a tengelyek azonosítását. A méréseket a BME-S1 adathalmaz azon részén végeztük el, amelyek tartalmaznak tandem típusú tengelycsoportokat, mivel ezek az érdekesek számunkra.

A 5.8 táblázat mutatja be, hogy az egyes megoldások milyen eredményeket értek el a BME-S1 adathalmaz tisztán szimulált jelein. A táblázat utolsó két oszlopában megfigyelhető, hogy a neurális hálózat alapú megoldásaink, még az optimális maximumalapú tengelydetektálókat is messze túlteljesítik. Az egyes metrikákat a 5.7 táblázat mutatja be. Ebben az esetben MAE és RMSE értékek azt jelentik, hogy abban az időpillanatban, amikor valóban elhaladt a tengely a szenzorsor felett, akkor a számított érték szerint mennyire volt távol a tengely a szenzorok keresztmetszetétől.

Fontos megjegyezni, hogy az Opt-LM megközelítés ortotróp hidak esetén működőképes megoldás, míg azt Opt-LM-Sum, illetve az Opt-LM-Edge vasbeton hidaknál funkcionál jobban. Miután a tengelydetektáló neurális hálózatunk vasbeton hidakon is nagy megbízhatósággal működőképes lehet, a maximumalapú megközelítésekkel összehasonlítva még szembetűnőbb a különbség.

Az 5.9 táblázat azt mutatja be, hogy a jelek terjedelméhez viszonyított 5% szórású zaj milyen szinten rontja a tengelydetektáló hálózat eredményét a CNN-Edge esetben. A táblázatból kiolvasható, hogy a MAE, RMSE esetében a tengelyek lokalizálásánál kicsit több, mint 30%-os romlást tapasztalunk a zaj bevezetésével. A Precision, Recall és F1-score értékekre is hatással volt a jel zajosítása, ezen esetekben a korábban detektált tengelyek kb. 97 százalékát találjuk meg, azonban még így is jóval felülmúlja a hasonló lokális maximum alapú megoldások korlátait. Itt érdemes azt is megemlíteni, hogy a BME-S1 adathalmaz sok olyan szélsőséges tengelyeloszlású járművet tartalmaz, ami a valós forgalomban ritka. Így valószínűleg valós esetekben a tengelyek nagyobb százalékát találnánk meg helyesen.

A 5.4 ábrán látható, hogy a korábbiakban bemutatott tengelydetektáló neurális hálózat nem csupán szintetikus adatokon volt tesztelve, hanem jól működött valós adatok klasszifikációja során is. Az algoritmus valós forgalmi szituációkban is helyesen tudta azonosítani a gépjárművek tengelyeinek pozícióját.

A [3] cikkel is összemérhető eredményeket érünk el. A tengelyeket egy zajos adathalmazon is átlagosan 7,2 cm átlagos hibával tudtuk lokalizálni, míg az említett cikkben a tengelytávolságok átlagos hibája 35,4 cm volt. A két érték nem pontosan ugyanazt jelöli, mivel a tengelytávolság két tengelylokáció helyének különbségét adja meg, ahol a sebesség hibáját is figyelembe kell már venni. A virtuális tengely módszerrel (Virtual Axle Method) összehasonlítva pedig megoldásunknak előnye, hogy nem függ a sebességbecslés lépésétől.

6. fejezet

Kitekintés, jövőbeli tervek

Ebben a fejezetben célunk, hogy röviden összefoglaljuk, hogy az általunk fejlesztett algoritmusok hogyan épülnek be az általunk fejlesztett BWIM-rendszerbe, majd pedig megfogalmazzuk továbbfejlesztési lehetőségeket. Ez a fejezet vázlatosan bemutat pár olyan modult is, amelyek még a BWIM-rendszerben megtalálhatóak, azonban nem fókusz a dolgozatnak.

6.1. BWIM-rendszer

A 6.1 táblázatban az látszik, hogy az egyes modulok, amik elérhetőek jelenleg a rendszerben, azok melyik korábbi jellemzők kinyerésétől függnnek (✗), és melyik jellemző kiszámítását (✓) végzik el.

Modul	Jellemző			
	Időablak	Sebesség	Tengelypozíció	Tengelysúly
Ablakdetektáló NN	✓			
Lokális maximum alapú sebességbecslő	✗	✓		
Sebességbecslő NN	✗	✓		
Lokális maximum alapú tengelydetektáló	✗		✓	
Tengelydetektáló NN	✗		✓	
Moses-algoritmus	✗	✗	✗	✓
Tengelysúlybecslő NN	✗	✗	✗	✓

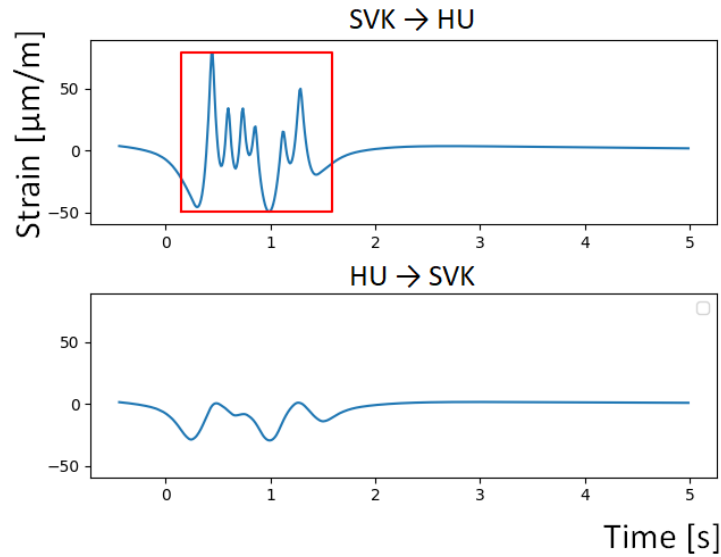
6.1. táblázat. A táblázat azt mutatja, hogy az egyes modulok mely jellemzőt számítják ki (✓), és mely jellemzők kiszámításától függnnek (✗) a bemenet révén. A neurális háló alapú megoldások az NN rövidítéssel vannak jelölve.

Az ablakdetektáló neurális hálózat alább kerül kifejtésre a 6.2 fejezetben. A sebességbecslő és tengelydetektáló modulokat korábban bemutattuk már. A Moses-algoritmus a 3.2.4 fejezetben került kifejtésre, a neurális háló alapú tengelysúlybecslő pedig korábban az [5] cikkben került publikálásra. A Moses-algoritmusnak egy olyan továbbfejlesztése van a rendszerünkben implementálva, ami több járművet is kezelni tud, de nem olyan pontossággal, mint egy neurális háló alapú megoldás. Itt ki kell emelnünk, hogy a Moses-algoritmus

érzékenyebb a zajra, itt egészen pontosan a tengelydetektálás és a sebességbecslés hibáira. A tengelysúlybecslő neurális háló előnye, hogy ezeket a hibákat/zajokat, amik az előfeldolgozó lépésekben keletkeznek, jobban kezeli [33].

6.2. Ablakdetektálás

Miután a szenzorok folyamatosan szolgáltatnak adatokat (olyan esetekben is, amikor nincs forgalom a hídon) fontos feladat kizárólag a jel releváns részeinek megtartása, ezeket a releváns részeket nevezzük gépjármű ablakoknak. Az így kapott gépjármű ablakokból meg lehet határozni a sebességet, a tengelypozíciókat, illetve a tengelysúlyokat.

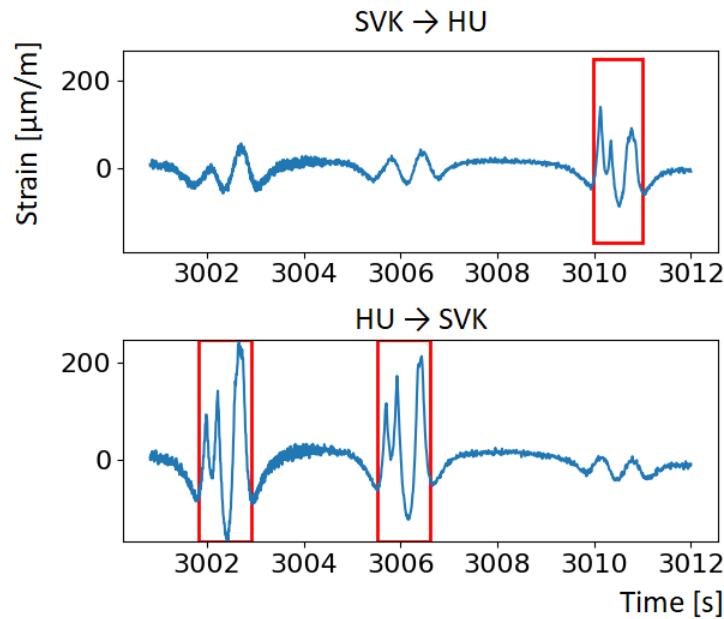


6.1. ábra. Az ábrán egy szimulált minta látható, a két sáv összegjeleit az idő függvényében mutatja. A pirossal bekeretezett részen egy jármű haladt át a hídon.

A 6.1 ábrán a két sáv szenzorainak összegzett jele látható. Megfigyelhető, hogy az SVK→HU sávban elhaladó jármű a másik sáv szenzoraira is hatással van. Amennyiben a gépjármű ablakok detektálásnál egyszerűen csak egy naiv mozgóátlag- vagy lokális szélsőérték-alapú determinisztikus megközelítést alkalmaznánk, akkor ilyen esetekben azt kapnánk, hogy mindkét sávban közlekedett jármű, ami helytelen következtetés. Az ilyen komplexebb összefüggések felismerésére neurális hálók alkalmazása jelentett megoldást számunkra.

A fejlesztett neurális hálózat épített a 4. fejezetben bemutatott sablonra. A korábbiakban bemutatott szimulációs algoritmus segítségével be- és kimenet párokat hoztunk létre a tanításához. A neurális hálózat bemenete az adott szimuláció teljes időtartamában rögzített jel volt, az elvárt kimenete pedig az a maszk, amit a 4. fejezetben is használtunk. Ez a maszk adta meg egy-egy jármű ablakainak határait. Így lényegében a tengelydetektáló algoritmushoz hasonló bináris osztályozási problémát kaptunk. Minden időpontra osztályozni kellett, hogy egy-egy sávban közlekedett-e akkor jármű.

A 6.2 ábrán egy valós mért jel látható, ahol az ablakdetektáló helyesen működött.



6.2. ábra. Egy valós mért jelen bemutatva látszik az ablakdetektáló működése. A neurális hálózat a pirossal bekeretezett időintervallumokban és sávokban talált járműveket.

6.3. Jövőbeli tervek

A jelenlegi rendszerben olyan egyszerű zajszűrő megoldásokat alkalmazunk, mint a Gauss-szűrő vagy a Bilateral Filter szűrő. Mások hoztak létre CWT alapú zajszűrő megoldásokat korábban [39]. Célunk, hogy olyan zajszűrő megoldásokat fejlesszünk ki, amik még pontosabb számításokat tesznek lehetővé.

Emellett arra is fel kívánjuk készíteni a rendszert, hogy akkor is működőképes maradjon, ha egy-egy szenzor meghibásodik. Erre már készítettek korábban egy mélytanulás alapú megoldást [40]. Ez a terület további vizsgáldást igényel. A jelenlegi moduljaink közül a lokális maximum alapú sebességbecslő és tengelydetektáló a szenzorok kiesésére kevésbé érzékeny, mint a neurális háló alapú megoldások. A tengelysúlybecslő megoldások egyike sincs jelenleg felkészítve szenzorok kiesésére.

A továbbiak során célunk nagy mennyiségű adat feldolgozása és kiértékelése, ezek vizsgálata a híd fáradásának szempontjából.

7. fejezet

Konklúzió

Munkánk során két újszerű sebességbecslő és tengelydetektáló modult készítettünk el.

A BWIM rendszerek esetén az iparban jelenleg olyan megoldások elterjedtek, ahol az aszfaltba építhető szenzorokat használnak vagy kamera alapú járműfelismerést alkalmaznak. Az általunk elkészített BWIM-rendszer előnye, hogy az általunk implementált megoldás esetén a nyúlásmérő szenzorok a híd trapézbordáin helyezkednek el, ami egy jóval egyszerűbb üzemeltetést és bővíthetőséget biztosít.

Az újszerű konvolúciós neurális háló alapú sebességbecslő modul egy keresztmetszet nyúlásmérő bélyegei alapján is képes a járművek sebességét az esetek 95%-ban maximum 4,72 km/h hibával megállapítani egy szintetikus adathalmazon. Amennyiben csak a nehezebb járműveket vesszük figyelembe, akkor ez az érték még kisebb lehet. Egy kisebb méretű valós annotált adathalmazon is 0,90 km/h átlagos hibával sikerült a járművek sebességét megállapítani, ami 1,79% átlagos százalékos hibának felel meg. Ezek az eredmények több tekintetben is jobbak, a korábbi hasonló [3] cikkben közölt megoldás eredményeihez képest. Azonban nem érik el a több keresztmetszet jeleit is használó lokális maximum alapú sebességbecslő pontosságát. Ez a megoldásunk megbízhatóan működhet kisebb forgalmú hidakon alacsonyabb telepítési költségek mellett a híd fáradásának monitorozása céljából. Emellett a több keresztmetszetben is szenzorokat tartalmazó hidakon is egy jó második megoldás lehet a feladat ellátására.

A tengelydetektáló neurális hálók esetén azáltal, hogy nem minden egyes szenzor jelét külön-külön használjuk, a megoldás hordozhatóvá válik, így akár vasbeton szerkezetű hidak esetén is alkalmazható lehet, ahol a lokális maximum alapú tengelydetektáló megoldás nem nyújt megbízható eredményeket. A szintetikus adathalmazon kiemelkedő eredményeket értünk el a CNN-Edge hálóval, a recall értéke 99,57%, az F1-score pedig 99,08% lett, szemben az ugyanazon összegjeleket alkalmazó lokális maximum alapú tengelydetektáló optimális eredményeihez képest (84,43% recall és 91,56% F1-score). A CNN-Sum neurális háló esetén a sávok összegjeleit vizsgáljuk. Ebben az esetben 99,04% recall és 99,27% F1-score eredményeket tapasztaltunk, az ehhez tartozó maximum alapú tengelydetektáló (Opt-LM-Sum) esetében pedig ezek az értékek 83,02% és 90,72% voltak. A zajos szintetikus adathalmazon is jobb eredményeket érnek el megoldásaink. A felsorolt eredmények szemléltetik, hogy a neurális háló alapú megközelítések messze felülmúlják a lokális maximum alapú megoldások pontosságát, megbízhatóságát elsősorban vasbeton szerkezetű hidak esetén.

Mivel a megoldásaink a kieső szenzorok problémájára érzékenyebbek, mint a hagyományos algoritmusok, ezért további kutatásaink során a kieső szenzor kezelése kell, hogy fókuszot kapjon.

Köszönetnyilvánítás

Köszönetet szeretnénk mondani Dr. Völgyi Istvánnak, Dr. Kollár Dénesnek és Dr. Joó Attila Lászlónak, a BME Éptömérnöki Kara Hidak és Szerkezetek Tanszékének oktatóinak, hogy támogatták dolgozatunk elkészítését hasznos tanácsokkal, észrevételekkel.

Irodalomjegyzék

- [1] S. H. Lee, L.-S. An, Y. C. Park, and H.-K. Kim, „How to use bwim data in fatigue evaluation of a medium-span steel bridge,” *International Journal of Steel Structures*, vol. 22, no. 6, pp. 1817–1828, 2022.
- [2] P. Burnos and D. Rys, „The effect of flexible pavement mechanics on the accuracy of axle load sensors in vehicle weigh-in-motion systems,” *Sensors*, vol. 17, no. 9, p. 2053, 2017.
- [3] T. Kawakatsu, K. Aihara, A. Takasu, and J. Adachi, „Fully-neural approach to heavy vehicle detection on bridges using a single strain sensor,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3047–3051, IEEE, 2020.
- [4] S. R. Lorenzen, H. Riedel, M. M. Rupp, L. Schmeiser, H. Berthold, A. Firus, and J. Schneider, „Virtual axle detector based on analysis of bridge acceleration measurements by fully convolutional network,” *Sensors*, vol. 22, no. 22, p. 8963, 2022.
- [5] B. Szinyéri, B. Kővári, I. Völgyi, D. Kollár, and A. L. Joó, „A strain gauge-based bridge weigh-in-motion system using deep learning,” *Engineering Structures*, vol. 277, p. 115472, 2023.
- [6] Y. Wu, L. Deng, and W. He, „Bwimnet: a novel method for identifying moving vehicles utilizing a modified encoder-decoder architecture,” *Sensors*, vol. 20, no. 24, p. 7170, 2020.
- [7] D. Kollár, I. Völgyi, and A. Joó, „Assessment of residual stresses in welded t-joints using contour method,” *Thin-Walled Structures*, vol. 190, p. 110966, 2023.
- [8] R. Ma, Z. Zhang, Y. Dong, and Y. Pan, „Deep learning based vehicle detection and classification methodology using strain sensors under bridge deck,” *Sensors*, vol. 20, no. 18, p. 5051, 2020.
- [9] M. Lydon, S. E. Taylor, D. Robinson, A. Mufti, and E. Brien, „Recent developments in bridge weigh in motion (b-wim),” *Journal of Civil Structural Health Monitoring*, vol. 6, pp. 69–81, 2016.
- [10] P. Chatterjee, E. O'Brien, Y. Li, and A. González, „Wavelet domain analysis for identification of vehicle axles from bridge measurements,” *Computers & structures*, vol. 84, no. 28, pp. 1792–1801, 2006.
- [11] M. Sujon and F. Dai, „Application of weigh-in-motion technologies for pavement and bridge response monitoring: State-of-the-art review,” *Automation in Construction*, vol. 130, p. 103844, 2021.

- [12] F. Moses, „Weigh-in-motion system using instrumented bridges,” *Transportation Engineering Journal of ASCE*, vol. 105, no. 3, pp. 233–249, 1979.
- [13] W. He, T. Ling, E. J. OBrien, and L. Deng, „Virtual axle method for bridge weigh-in-motion systems requiring no axle detector,” *Journal of Bridge Engineering*, vol. 24, no. 9, p. 04019086, 2019.
- [14] T. Kawakatsu, K. Aihara, A. Takasu, and J. Adachi, „Deep sensing approach to single-sensor vehicle weighing system on bridges,” *IEEE Sensors Journal*, vol. 19, no. 1, pp. 243–256, 2018.
- [15] Y. Yu, C. Cai, and L. Deng, „Nothing-on-road bridge weigh-in-motion considering the transverse position of the vehicle,” *Structure and Infrastructure Engineering*, vol. 14, no. 8, pp. 1108–1122, 2018.
- [16] H. Zhao, N. Uddin, E. J. O’Brien, X. Shao, and P. Zhu, „Identification of vehicular axle weights with a bridge weigh-in-motion system considering transverse distribution of wheel loads,” *Journal of Bridge Engineering*, vol. 19, no. 3, p. 04013008, 2014.
- [17] X. Jian, Y. Xia, S. Sun, and L. Sun, „Integrating bridge influence surface and computer vision for bridge weigh-in-motion in complicated traffic scenarios,” *Structural Control and Health Monitoring*, vol. 29, no. 11, p. e3066, 2022.
- [18] F. Carraro, M. S. Gonçalves, R. H. Lopez, L. F. F. Miguel, and A. M. Valente, „Weight estimation on static b-wim algorithms: A comparative study,” *Engineering Structures*, vol. 198, p. 109463, 2019.
- [19] R. Pimentel, D. Ribeiro, L. Matos, A. Mosleh, and R. Calçada, „Bridge weigh-in-motion system for the identification of train loads using fiber-optic technology,” in *Structures*, vol. 30, pp. 1056–1070, Elsevier, 2021.
- [20] E. O’Brien, D. Hajializadeh, N. Uddin, D. Robinson, and R. Opitz, „Strategies for axle detection in bridge weigh-in-motion systems,” in *Proceedings of the international conference on weigh-in-motion (ICWIM 6)*, pp. 79–88, 2012.
- [21] S. Ren, K. He, R. Girshick, and J. Sun, „Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [22] S.-Z. Chen, G. Wu, and D.-C. Feng, „Development of a bridge weigh-in-motion method considering the presence of multiple vehicles,” *Engineering Structures*, vol. 191, pp. 724–739, 2019.
- [23] L. Deng, W. He, Y. Yu, and C. Cai, „Equivalent shear force method for detecting the speed and axles of moving vehicles on bridges,” *Journal of Bridge Engineering*, vol. 23, no. 8, p. 04018057, 2018.
- [24] W. He, L. Deng, H. Shi, C. Cai, and Y. Yu, „Novel virtual simply supported beam method for detecting the speed and axles of moving vehicles on bridges,” *Journal of Bridge Engineering*, vol. 22, no. 4, p. 04016141, 2017.
- [25] H. Kalhori, M. M. Alamdari, X. Zhu, B. Samali, and S. Mustapha, „Non-intrusive schemes for speed and axle identification in bridge-weigh-in-motion systems,” *Measurement Science and Technology*, vol. 28, no. 2, p. 025102, 2017.

- [26] A. Lansdell, W. Song, and B. Dixon, „Development and testing of a bridge weigh-in-motion method considering nonconstant vehicle speed,” *Engineering Structures*, vol. 152, pp. 709–726, 2017.
- [27] C. Tan, B. Zhang, H. Zhao, N. Uddin, H. Guo, and B. Yan, „An extended bridge weigh-in-motion system without vehicular axles and speed detectors using nonnegative lasso regularization,” *Journal of Bridge Engineering*, vol. 28, no. 5, p. 04023022, 2023.
- [28] S.-S. Ieng, „Bridge influence line estimation for bridge weigh-in-motion system,” *Journal of Computing in Civil Engineering*, vol. 29, no. 1, p. 06014006, 2015.
- [29] E. J. OBrien, M. Quilligan, and R. Karoumi, „Calculating an influence line from direct measurements,” in *Proceedings of the Institution of Civil Engineers-Bridge Engineering*, vol. 159, pp. 31–34, Thomas Telford Ltd, 2006.
- [30] E. J. OBrien, C. W. Rowley, A. Gonzalez, and M. F. Green, „A regularised solution to the bridge weigh-in-motion equations,” *International Journal of Heavy Vehicle Systems*, vol. 16, no. 3, pp. 310–327, 2009.
- [31] E. J. OBrien, L. Zhang, H. Zhao, and D. Hajializadeh, „Probabilistic bridge weigh-in-motion,” *Canadian Journal of Civil Engineering*, vol. 45, no. 8, pp. 667–675, 2018.
- [32] Z. Wang and Y. Wang, „Bridge weigh-in-motion through bidirectional recurrent neural network with long short-term memory and attention mechanism,” *Smart Structures and Systems*, vol. 27, no. 2, pp. 241–256, 2021.
- [33] B. Szinyéri and B. Kovari, „Comparison of a deep learning-based axle load estimator and the matrix method in strain gauge-based bridge weigh-in-motion systems,” in *2023 10th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, IEEE, 2023.
- [34] X. Zheng, D.-H. Yang, T.-H. Yi, and H.-N. Li, „Development of bridge influence line identification methods based on direct measurement data: A comprehensive review and comparison,” *Engineering structures*, vol. 198, p. 109539, 2019.
- [35] B. Szinyéri, B. Kővári, I. Völgyi, D. Kollár, and A. L. Joó, „Bwim-bme-simulated-i,” 2022.
- [36] K. He, X. Zhang, S. Ren, and J. Sun, „Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [37] A. K. Dubey and V. Jain, „Comparative study of convolution neural network’s relu and leaky-relu activation functions,” in *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, pp. 873–880, Springer, 2019.
- [38] I. Loshchilov and F. Hutter, „Sgdr: Stochastic gradient descent with warm restarts,” *arXiv preprint arXiv:1608.03983*, 2016.
- [39] Z. Zhang, C. Sun, and Y. Huang, „Sparse signal recovery for wim measurements from undersampled data through compressed sensing,” *Measurement*, vol. 151, p. 107181, 2020.

- [40] Y. Zhuang, J. Qin, B. Chen, C. Dong, C. Xue, and S. M. Easa, „Data loss reconstruction method for a bridge weigh-in-motion system using generative adversarial networks,” *Sensors*, vol. 22, no. 3, p. 858, 2022.