Autópályás közlekedési szituációk tömörítése Variációs Autoenkóderrel manőver klasszifikáció céljából.

Olivér Rákos*. Szilárd Aradi.**
Tamás Bécsi*

**Budapesti Műszaki és Gazdaságtudományi Egyetem, Közlekedés- és Járműirányítási Tanszék*
1111 Budapest, Stoczek, Egry József u. 2. (e-mail: rakos.oliver@ kjk.bme.hu).

Abstract: Az autonóm járművek tervezésénél probléma merül fel. Szoftverük három fő rétegre épül: észlelés, tervezés, és végrehajtás illetve működtetés. A tervezési réteg a rövid és hosszú távú helyzet előrejelzésével foglalkozik, amelyek döntő fontosságúak az intelligens járművek számára. Bármilyen módszert is használnak az előrejelzésre, a járművek dinamikus környezetét mindenképpen fel kell dolgozni a pontos hosszú távú előrejelzéshez. Ebben a cikkben egy módszert javasolunk a dinamikus környezet előfeldolgozására autópályás forgalmi helyzetben. A módszer a környező járművek strukturált adatait használja, és átalakítja azokat egy foglalási rácsra (Occupancy Grid), amelyet egy Konvolúciós Variációs Autoenkóder (CVAE) dolgoz fel. A rácsokat (2048 képpont) a enkóder 64 dimenziós látens vektorrá tömöríti, amit a dekóder próbál rekonstruálni. A foglaltsági hálók idősorát is feldolgozzuk 3 dimenziós Autoencoderrel, vagy LSTM rekurrens neurális hálózattal. A kimeneti pixelintenzitásokat úgy értelmezzük, mint egy valószínűséget arra, hogy a megfelelő mező egy járművel van elfoglalva. Ennek a módszernek az az előnye, hogy előzetesen feldolgozza a dinamikus környezet strukturált adatait, és megjeleníti azokat egy alacsonyabb dimenziós vektorban, amely felhasználható a rá épülő további feladatokban. Ez az ábrázolás nem kézzel készített vagy heurisztikus, hanem az adatminták mögötti látens állapotok, melyeket a tanítás során hoz létre az eszköz. Az ábrázolásokat felhasználjuk manőver klasszifikációra.

1. BEVEZETÉS

A hierarchikus felépítés a leggyakoribb megközelítés az önvezető járművek szoftverének tervezésére. E megközelítés szerint három fő rétegre gondolunk: észlelésre, tervezésre és működtetésre (Geng, *et al.* 2017). Az észlelés egyesíti az szenzorokból származó információkat, hogy egy modellt alkosson a környező világról és az abban lévő objektumokról. A tervezési rétegben a járműmodellt és a környezetmodellt használják az útvonal, a pálya, és a manőverek megtervezéséhez, figyelembe véve a megadott vezetési stílust, a célállomást és egyéb beviteli utasításokat vagy kényszer feltételeket. A működtető réteg felelős a tervek megvalósításáért, parancsokat ad az aktuátoroknak, és visszajelzést ad az alacsonyabb rétegeknek az aktuátorok állapotáról. Ennek és az észlelési rétegnek a működése kívül esik e cikk keretein. A hangsúly a második, tervezési rétegen, azon belül is a viselkedés előrejelzésén van. Az autonóm járműveknek folyamatosan dönteniük kell a manőverekkel kapcsolatban, hogy biztonságosan navigálhassanak. Ezek a döntések lehetnek egyéni vagy együttműködőek a forgalmi rendszer alapján (Ploeg, Redmer, Haan, 2019).

Ahhoz, hogy a tervezési réteg megfelelően működjön, azaz biztonságos, pontos, megvalósítható terveket készítsen, képesnek kell lennie értelmezni és megjósolni a környező járművek vezetőinek rejtett szándékait vagy viselkedését. (Llamazares, Molinos, Ocana, 2020). Ezt a feladatot meg lehet oldani a trajektóriák előrejelzésével, a manőver észlelésével és előrejelzésével, illetve ezeket kombinálva multimodális trajektória előrejelzéssel. Mivel az autópályás közlekedés egy interaktív rendszer, a járművezetők döntéseit minden más vezető befolyásolja, és mivel be kell tartani a közös közúti szabályokat, elmondható, hogy jelentős összefüggések vannak a szereplők viselkedései között. Ez azt jelenti, hogy az ágensek szándékai egy forgalmi helyzetben, vagy maga a pálya előre jelezhető az egyes ágensek belső állapotát valahogyan jellemző információk felhasználásával. Ha ezeket figyelmen kívül hagyjuk, akkor korlátozott mértékben lehet a jövőbeli állapotokról előrejelzést tenni (Rákos, Aradi, & Bécsi, 2020). Egy ilyen erősen összekapcsolt rendszerben az adott ágens múltbeli állapotai nem kódolják a forgalmi helyzet belső állapotait, amelyek az előrejelzéshez szükségesek. Ezt úgy is meg lehet fogalmazni, hogy a környezeti információk nélkül pusztán jármű múltbeli pályájából nem következtethetünk a jövőre. A viselkedés előrejelzése nagyon kihívást jelent, és ezekre a megállapításokra nagy figyelmet fordít a szakirodalom.

A trajektóriák mintázatait egy Gauss-folyamat regresszió alapján osztályozni vagy megjósolni lehet az idézett cikk alapján (Trautman, Krause, 2010). Egy másik elterjedt modell egy hierarchikus dinamikus bayesi hálózat, amelyet a viselkedés jövőbeli mintáinak előrejelzésére használnak (Dagli, Brost, & Breuel, 2002).

Egy lehetséges kategorizálási megközelítés szerint a mozgás-előrejelző rendszereket a fizikai alapú, manőverelapú és interakciótudatos modellek közé lehet sorolni (Lefevre, Vasquez, & Laugier, 2014). A legegyszerűbb a fizikai alapú,

mivel csak a fizika törvényét veszik figyelembe a járművek mozgásában. Fejlettebb modellt kapunk, ha figyelembe vesszük a járművezetők szándékait is, ezek a manőverelapú modellek. Az interakciótudatos modellek célja, hogy felismerjék és figyelembe vegyék a közlekedési helyzetben lévő járművek közötti kölcsönös függőségeket. Ebben a cikkben egy módszert mutatunk be, amely az ágensek hatását az adott forgalmi helyzetben egy foglaltsági háló (occupancy grid) kódolásával veszi figyelembe. Egy ilyen kép nagy dimenzióval rendelkezik, ezért némi tömörítésre van szükség. A jól bevált Variational Autoencoder séma a pályák másolására (Rákos, Aradi, & Bécsi, 2020) csak akkor használható az előrejelzési feladathoz, ha a bemeneti adatok környezeti információkat tartalmaznak, amelyekből a dekóder rész megfelelő előrejelzéseket vonhat le a jövőbeli mozgásról. Ennek lényege, hogy a pályából kialakított kontextusvektor kiegészül a környezeti információkat kódoló "szituáció" vektorral. A enkódert és a dekódert úgy tanítják, hogy az elvárt kimeneten a jövőbeli trajektória legyen. Kim és munkatársai egy modellt javasolnak, amely megjósolja a 0,5 s, 1 s és 2 s időhorizonton járművek jövőbeli elhelyezkedését a foglaltsági háló térképen (Kim *et al.* 2017). Ezzel szemben e cikk célja csak a lehető legértékesebb és legszűkebb információk kiszűrése a környező járművek helyéről.

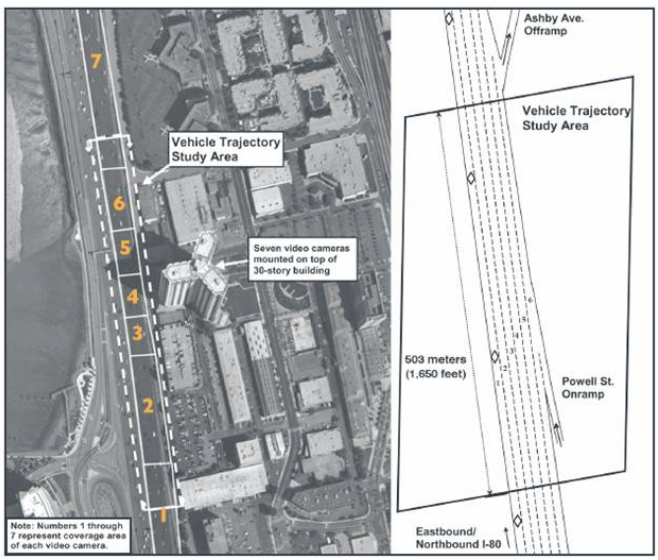
A szituácivektor többféleképpen határozható meg. Az egyik megközelítés a forgalmi környezetét tartalmazó strukturált adatok . A (Deo, Trivedi, & June, 2018) tanulmány egy újszerű megközelítést mutat be a járművek autópályán történő mozgásának előrejelzésére. A modell kimenete a pályák multimodális eloszlása. A bemenet az EGO hosszirányú és oldalirányú pozicióiból épül fel, és ehhez hat környező jármű EGO-hoz számított relatív pozíciója van hozzáfűzve megadott sorrendben: két autó az ego előtt és mögött, és további két pár a szomszédos sávokban. Ez a bemeneti pozíciókat tartalmazó tenzor meghatározott strukturált sorrendben van szervezve, hogy a enkóder megtanulhassa a megfelelő kontextusábrázolást. Megjegyezzük, hogy a bemeneti vektorok egy 64 dimenziós fully-connected réteggel vannak beágyazva az LSTM rétegek elé, így a szitució ábrázolás 64 dimenziós. Nagyon hasonló ábrázolás található Feng és mtsai kutatásában. (Feng, *et al.* 2019). Conditional Variational Autoencodert javasolnak a szándékon alapuló trajektória előrejelzéséhez. A bemenetek hasonlóak az előző példához. Egy strukturált historikus adat sorozatot veszünk, amely tartalmazza az EGO és öt környező jármű elmozdulását, sebességét és az ego és a másik jármű közötti hosszirányú távolságot. A különbség az, hogy a hátsó jármű nem szerepel a strukturált bemenetben, mivel szerintük az első jármű nem vállal felelősséget a hátsó mozgásáért a forgalomszabályozás szerint.

Az ágensek a szomszédaik mozgásával számolva igazítják útjukat és trajektóriájukat, miközben többi környező ágens befolyásolja őket. Az autópályán minden járműnek különböző számú szomszédja lehet, és zsúfolt forgalmi helyzetben az összefüggések száma olyan magas lesz, hogy nem lehet egyszerűen kiszámítani. Ezért vezették be az úgynevezett „Social Pooling” folyamatát (Alahi, *et al.* 2016), hogy egyesítsék az összes szomszédos állapotból származó információkat. A zsúfolt helyen mozgó gyalogosok trajektóriájának előrejelzésére és kezelésére használták először. Ezt az ötletet továbbfejlesztették a multimodális járműtrajektória-előrejelzéshez a következő cikkben: (Deo, Trivedi, 2018), és a szerzők egy új konvolucionális „Social Poolingot” alkalmaztak a környező járművek historikus adatait kódoló LSTM-állapotok tenzorainak Fully Connected rétegei helyett.

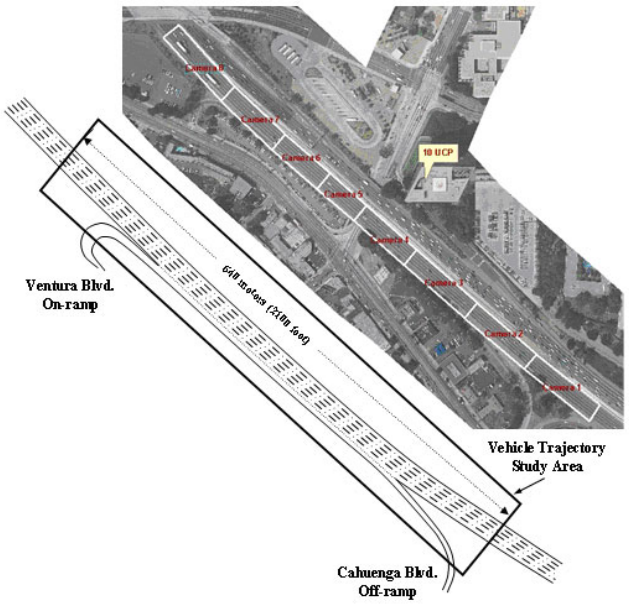
Ez a cikk bemutat egy 2D illetve 3D konvolúciós Variational Autoencodert a foglaltsági hálók másolására és tömörítésére, ezáltal helyzetvektor létrehozására. A mély neurális hálózatok számos területen nagy figyelmet kaptak, mivel ígéretes teljesítményt mutattak a gépi tanulás különböző feladataihoz (Goodfellow, Bengio, Courville, 2016). A 4.1-es alszakaszban az előkészítés szerepel, és leírásra kerülnek a rácsképek adatkészletének felépítésének részletei. A módszertant és a matematikai levezetést a 4.2 alszakasz, a tanítás részleteit pedig a 4.3 fejezetben ismertetjük. A tanítási eredmények és a enkóder minőségének megvitatását és a klasszifikációs képességet az 5. szakasz foglalja össze. A tanulási görbék mellett a klasszifikáció jóságát mérő metrika is szerepel az ábrákon, és tánlázatban is. A 6. szakasz a konklúziót tartalmazza röviden összefoglalva.

2. PROBLÉMA FELVETÉS

Ebben a cikkben a viselkedés előrejelzésével foglalkozunk, melynek lényeges része a forgalmi szituációban szereplő ágensek elemzése. A tervezéshez elengedhetetlen, hogy a lehető legpontosabban megbecsüljük a többi közlekedő jármű lehetséges pályáját.



1. Ábra. Madártávlati kép az Intersection-80 (I-80) autópálya-részletről, ahol az adatfelvétel történt



2. Ábra. Madártávlati kép a US-101 autópálya-részletről, ahol az adatfelvétel történt

A megbízható viselkedés előrejelzéshez szükséges környezeti információk kinyerése egy nagyon fontos részfeladat, foglaltsági hálóból mély tanulás segítségével izgalmas probléma. Ezért ebben a cikkben bemutatunk egy Konvolúciós Variációs Autoenkóder (CVAE) architektúrát és egy eljárást a foglalási rács dimenziójának 3,125 % -ra való csökkentésére. A képek mérete 16-szor 128 pixel, a látens térdimenzió 64. A jármű által lefedett pixelek értéke 1, míg a többi pixel értéke 0. Az adatok előkészítését a 4.1 alszakaszban részletezzük. A használt modellt a 4.2 alszakasz ismerteti, valamint a tanítás részleteit az 4.3 alszakaszban.

3. KONTRIBÚCIÓ

Ebben a dolgozatban bemutatunk egy módszert az autópályás forgalmi szituáció ábrázolására neurális hálózatok számára. A foglaltsági hálók tömörítésével és másolásával a javasolt modell megtanul készíteni egy reprezentációt, amely más feladatokban, például klasszifikációban bemenetként használható. Miközben az autoenkóder lemásol egy 2048 képpont méretű képet, létrehoz egy 64 dimenziós „szituáció” vektor reprezentációt, amely értékes információkat tartalmaz az adott időpontról. A variációs autoenkóderben a dekóder mögötti valószínűség eloszlás Bernoulli típusú, de az enkóder által generált látens tér és a prior eloszlását Gaussinak feltételeztük. A generált szituáció vektorok idősorával manőver detekcióra tanítunk be rekurrens Long- Short Term Memory (LSTM) neurális hálózatot.

Hasonlóan az előző bekezdésben foglaltakhoz, bemutatunk egy 3 dimenziós variációs autoenkódert is. Első két dimenziójában a konvolúciós kernelek a foglaltsági háló két dimenzióján hatnak, a harmadik pedig az időbeli konvolúciót fejez ki. Az enkóder szintén egy 64 dimenziós szituáció vektort szolgáltat, ez azonban már az egész idősor információját tartalmazza rekurrens neurális hálózat alkalmazása nélkül is.

4. MEGVALÓSÍTÁS

Ebben a részben bemutatjuk a 2. szakaszban tárgyalt probléma javasolt megoldásának részleteit, kezdve az adatok forrásával és feldolgozásával a 4.1 alszakaszban. Az alkalmazott modell és a veszteségfüggvény, valamint a hozzájuk kapcsolódó matematikai megfontolások a 4.2 alfejezetben találhatók. Végül a a tanítás részleteit a 4.3 tartalmazza.

4.1 Tanító adatkészlet

A tanulási feladathoz szükséges adatokat az NGSIM trajektória adatbázisából vettük. Ez az adatbázis két egyesült államokbeli autópályaszakaszon, az US-101-en és az I-80-ason áthaladó járművek trajektóriáit tartalmazza (Colyar, Halkias, 2007), (Colyar, Halkias, 2006). A járművek pontos helyei, sebességei és gyorsulási értékei 0,1 másodpercenként jelennek meg. 11,8 millió regisztert tartalmaz, amelyek mindegyike egy járművet reprezentál egy adott időpontban. Az 1. és 2. ábrán a I-80 illetve az US-101 autópálya részletek láthatók, ahol az adatfelvétel történt.

A foglalási háló képeit a következő lépésekben készítettük el. Először is, az algoritmus minden járművön át halad. Az adatbázisban szereplő minden járművet EGO járműnek tekintünk, és végig iterálunk a hozzá tartozó idősoron. A foglaltsági háló 0,5 méteres négyzetekből áll. Az a tér, amelybe a jármű kiterjed, foglaltnak tekintendő. Minden négyzet egy képpont, így egy 1 csatornás képet kapunk. Minden kép közepén az ego jármű hátsó részének központja található. Az algoritmus ezután megkeresi a járművet, amely a T időpontban az ego jármű közelében van, és beilleszti a rácsba. A mintavétel során 6 másodperc hosszú idősorokat is alkottunk, melyeket 3 one-hot címkével láttunk el manőver szerint. Amennyiben az idősor végén bekövetkezik egy sávváltás (jobbra vagy balra külön) akkor sávváltás, ha nem következik be akkor sávtartásnak címkéztük.

Oldalirányban (x) 4 méter távolságot veszünk figyelembe, jobbra és balra egyaránt. Hosszirányban (y) 32 méteres távolságot veszünk figyelembe, így a minták mérete 16, 128.

4.2 Módszer

A Variációs Autoenkóder (VAE) és az Adversarial Autoencoder (ADVAE) neurális hálózatokat sikeresen tanítottunk a foglaltsági hálók másolására és tömörítésére. A következőkben röviden összefoglaljuk az elméleti megfontolásokat. Az enkóder és a dekóder valószínűségi értelmezéséből kiindulva leírjuk azt az érvelést, amely a megfelelő veszteségfüggvény kiválasztásához vezet.

A VAE (Doersch, 2016) rendelkezik egy inherens regularizáló képességgel, mert az enkóder a bemenetet a látens tér fölötti eloszlássá alakítja, nem pedig egyetlen pontra. A dekóder mintavételezi ezt az eloszlást, és megpróbálja a lehető legpontosabban rekonstruálni az eredeti bemenetet. Ez a látens

eloszlás korlátozás nélkül konvergálhat egy Dirac-deltához, amit egy külön tag akadályoz meg a veszteség függvényben.

Az úgynevezett versengő tanulás (Adversarial Training) során egy diszkriminátor neurális hálózat verseng az enkóderrel (Makhzani, Shlens, Jaitly, Goodfellow, & Frey, 2015). Az enkóder maga a generatív folyamat, és mintákat generál a látens eloszlásból, hasonlót a prior eloszláshoz. Eközben a diszkriminátort arra optimalizálják, hogy megkülönböztesse a generált és a prior elosztási mintákat. Más szóval, az enkóder megpróbálja félrevezetni a diszkriminátort, és ezzel egyidejűleg az utóbbi megpróbálja megkülönböztetni a generáltat a priori mintáktól (Blei, *et al.* 2016).

Meghatározható egy $p(z)$ prior eloszlás a látens tér felett, ahol z a látens vektor. A valószínűségi dekódert és enkódert a $p\,(x \mid z)$ és $p\,(z \mid x)$ feltételes valószínűségi eloszlásokkal jellemezhetjük. A valószínűségi dekóder az x dekódolt változó feltételes eloszlását jelöli a z kódolt változó alapján, míg a valószínűségi enkóder ennek az ellenkezője. Továbbá szükséges, hogy a prior ismert eloszlás legyen, hogy könnyen lehessen imntavételezni a tanítás során. A mintavételi eljárás nem akadályozhatja meg a tanítási veszteség visszaterjedését. Az (1) egyenletben a prior egy normális eloszlás, ami azt jelenti, hogy elvárjuk, hogy a látens vektor komponensek függetlenek legyenek, és nulla várható értékkel rendelkezzenek egységnyi szórással.

$$p(z) = \mathcal{N}(0,\, I) \tag{1}$$

A enkóder eloszlását a Bayes-tétel fejezi ki,

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)} \tag{2}$$

Az integrálást nem lehet elvégezni, ezért közelítéssel kell élni. A $p(z \mid x)$ értéket a normál eloszlás közelíti.

$$q_x(z) \equiv \mathcal{N}\big(m(x), s(x)\big)\ m \in M; s \in S. \tag{3}$$

A M, S és Θ függvénykészleteket enkóder neurális hálózatokkal paraméterezhető függvények halmaza. Az optimumot az (m^*, s^*) függvény paraméterek adják, amelyekkel a Kullback-Leibler divergencia (KLD) (Kullback, Leibler, 1951) minimális.

A KLD meghatározása és a (2) egyenlet szerint

$$(\mathrm{m}^*, \mathrm{s}^*) = \underset{(\mathrm{m},\mathrm{s}) \in \mathrm{M} \times \mathrm{S}}{\operatorname{argmin}} \left(\log\big(\mathrm{q_x(z)}\big) - \mathrm{E}\left(\log \frac{\mathrm{p(x|\,z)p(z)}}{\mathrm{p(x)}} \right) \right), \tag{4}$$

ahol a tagok tovább faktorizálhatók

$$(m^*, s^*) = \underset{(\mathrm{m},\mathrm{s}) \in \mathrm{M} \times \mathrm{S}}{\operatorname{argmin}} E\Big(log\big(q_x(z)\big) - E\big(logp(z)\big) - E\big(logp(x|z)\big) + E\big(logp(x)\big)\Big). \tag{5}$$

Az utolsó tag független az optimalizálási paraméterektől, a többi pedig átrendezhető úgy, hogy

$$(m^*, s^*) = \underset{(m,s) \in M \times S}{argmin} E\Big(-logp(x|z) + KL\big(q_x(z), p(z)\big)\Big) \tag{6}$$

A második tag hatása a regularizáció. A KLD minimalizálásával priori és a generált $q_x(z)$ minták közötti távolság nem lehet túl nagy. Ezzel arra ösztönzi a enkódert, hogy hasonló bemeneti mintákhoz hasonló látens vektorokat rendeljen. Az első tag a dekódereloszlás negatív log-likelihoodja, amely még mindig nem számítható ki; ezért kihasználjuk, hogy mit lehet tudni az adatokról és mit várunk el a rekonstuált adatoktól. Az x bemeneti kép egy mátrix nulla és egy értékkel, mivel minden pixel eggyel van jelölve, ha foglalt, és nulla, ha üres. A dekóder kimenete 0 és 1 között van a kimeneti rétegben lévő szigmoid nemlinearitás miatt, így $f(z)$ úgy értelmezhető, hogy mekkora a valószínűsége annak, hogy az eredeti képpontérték 1. Emiatt a Bernoulli-eloszlás ésszerű feltevés a dekódereloszlásra vonatkozóan.

$$p(x|z) = d(z)^x\big(1 - d(z)\big)^{1-x}, \text{ahol } d \in D \tag{7}$$

Itt a D a dekóder neurális hálózatokkal paraméterezhető függvények halmaza. A feladat a dekóder log-likelihoodjának maximalizálása, vagy más szóval a negatív log-likelihood minimalizálása:

$$(m^*, s^*, d^*) = \underset{(m,s,d) \in M \times S \times D}{argmin} -E\big(x \log\big(f(z)\big) + (1-x)\log\big(1 - f(z)\big)\big) + KL\big(q_x(z), p(z)\big)\,. \tag{8}$$

Végül a várható értékeket sokaság általggal közelítve megkapjuk az alkalmazandó veszteség függvényt.

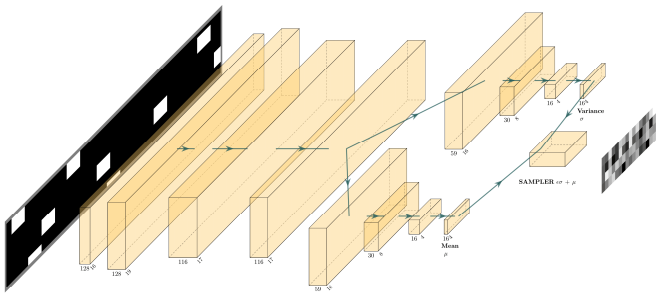
$$L = \sum_i -\big(x_i \log(d(z_i)) + (1 - x_i)\log\big(1 - d_{i(z_i)}\big)\big) + \frac{1}{2}\sum_i \big(s(x_i) - m(x_i)^2 - \log\big(s(x_i)\big) - 1\big) \tag{9}$$

Versengő tanítás során a második tag, ami a regularizációért felel el van hagyva, mert ezt a diszkriminátor hálózat végzi el. A diszkriminátor paraméterei arra optimalizálógnak, hogy az enkóder generált látens eloszlását különböztesse meg a prior eloszlástól. Ez egy bináris classzifikációs probléma. A kimenet egy dimenziós szigmoid nemlinearitással. Az 1 értéket akkor várjuk, ha a bemenet priori minta, 0 kimenetet akkor, ha a bemenet generált minta. Emellett a diszkriminátornak van egy másik szerepe is. Az enkódert a rekonstrukciós feladaton kívül arra is optimalizáljuk, hogy

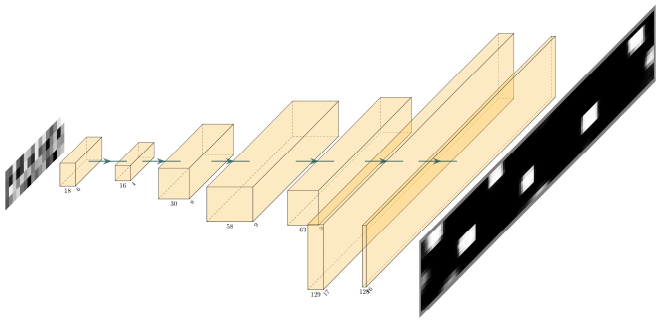
megtévessze a diszkriminátort. Ebben a lépésben a diszkriminátor súlyai nem módosulnak, csak az enkóderé, és a kimeneten a generált mintákra 1 kimenetet várunk – ez felel meg annak, hogy az enkóder átverni próbálja a diszkriminátort. Egyensúlyban minden prior és generált mintára $p = \frac{1}{2}$ értéket ad a diszkriminátor.

4.3 A tanítás részletei

A tanítás pythonban pytorch (…) környezetben valósult meg ADAM optimalizációval (…) 0.001 értékű tanulási rátával. Az autoenkóderek architektúrái a sematikus ábrákon láthatók. Az enkóder a 3. ábrán, a dekóder a 4. ábrán. A megjelenítésben csak a két dimenziós enkóder és dekóder rajzolható fel, de a 3 dimenziós párja nagyon hasonló. Az 1. Táblázat mutatja egyben a 2 dimenziós és 3 dimenziós enkóder paramétereit. A zárójeles paraméterek esetében az utolsó dőlt karakterrel szedett érték csak a 3 dimenziós változatra értendő. A 2. Táblázatban találhatóak a dekókerek paraméterei szintén 2 és 3 dimenzióban. Ezekből is jól látható, hogy az első két dimenzióban, ami a foglaltsági hálók két dimenzióján végeznek konvolúciót, a paraméterek megegyeznek. A „Trans" jelentése Transzpontál Konvolúciós réteg. Minden esetben a a Batch Normalizáció az adott dimenzióhoz és csatornaszámhoz illeszkedik, továbbá a LeakyReLU nemlinearitás paramétere minden esetben 0.2. Az utolsó előtti sor Padding oszlopában az o.p. rövidítés az output padding-nek felel meg. Ennek a céja a helyes tenzorméretek beállítása a dimenzió növelés során.



3. Ábra. A Convolúciós Enkóder sematikus ábrázolása.



4. Ábra. A Convolúciós Dekóder sematikus ábrázolása.

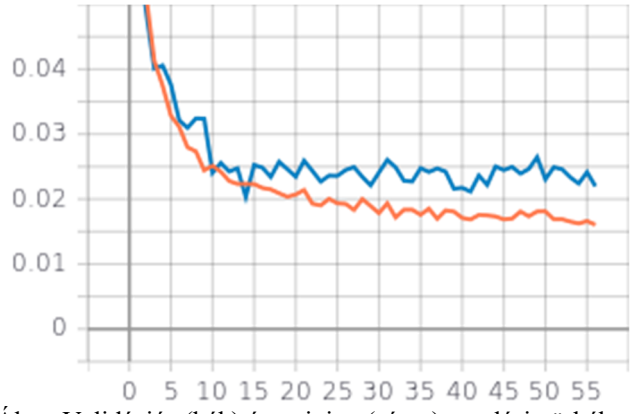
A 2 dimenziós foglaltsági hálók idősorának feldolgozásához LSTM neurális hálózatot alkalmaztunk. A bemenete 64 dimenziós (megegyezik a látens tér dimenziójával) tenzor, melyet egy fully connected réteg csökkent 32 dimenziósra. Egy ReLU aktivációs réteg után ezt kapja bemenetként a rekurrens LSTM cella, melynek a rejtett állapota 32 dimenziós. Az idősorból származó 32 dimenziós állapotot egy Reziduális kapcsolatokkal ellátott Több Rétegű Perceptron (MLP) csökkenti 3 dimenziósra. A hálózat belső dimenziói 24, 16, és a kimenet 3 dimenziós. A klasszifikációhoz az úgy nevezett Focal Loss függvényt használtük, mely segít a nem kiegyensúlyozott tanítóminták kezelésében, és nagyobb súlyt rendel a nehezebben tanulható mintákhoz (Lin, *et al.* 2017).

1. Táblázat. Enkóder paraméterek

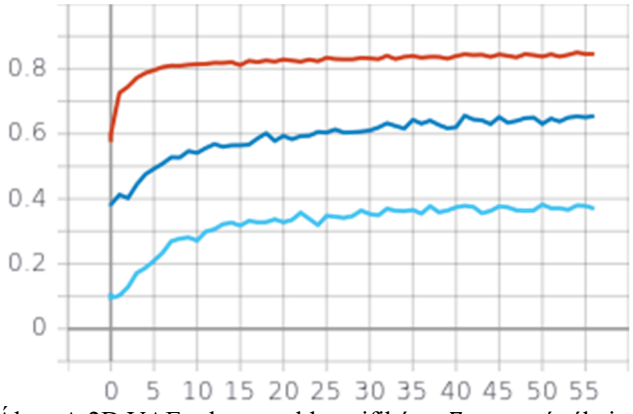
Rétegek	Csatorna Be-kimenet		Kernel	Stride	Padding
Az első szegmens					
Conv 2/*3*D LeakyRelu	1	3	(2,5,*3*)	1	(2,2,*1*)
Conv 2/*3*D BatchNorm LeakyRelu	3	5	(3,8,*3*)	1	1
Conv 2/3D BatchNorm LeakyRelu	5	8	(3,8,*4*)	1	0
A második szegmens					
Conv 2/*3*D BatchNorm LeakyRelu	8	5	(4,4,*4*)	(1,2,*1*)	(1,2,*0*)
Conv 2/*3*D BatchNorm LeakyRelu	5	4	(4,4,*4*)	(2,2,*1*)	(1,2,*0*)
Conv 2/*3*D LeakyRelu	4	3	(4,4,*3*)	(2,2,*1*)	(1,2,*0*)
Conv 2/3D	3	1	1	1	0

5. EREDMÉNYEK

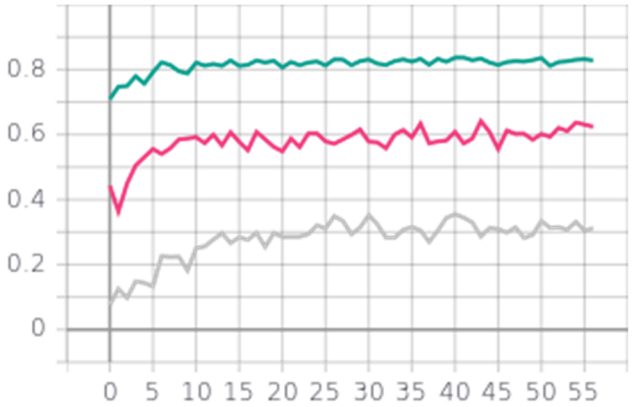
Összefoglaljuk, hogy milyen eredmények születtek a szituáció vektor felhasználásából. Először tekintsük a 2 dimenziós VAE reprezentációinak idősorként való felhasználását manőver detekcióra. Az 5. ábrán látható a veszteség függvény értéke az epoch függvényében, a kék görbe a validációs adathalmazon, a sárga görbe a tanító adathalmazon.



5. Ábra. Validációs (kék) és training (sárga) tanulási görbék a 2D VAE rekurrens klasszifikátor esetén



6. Ábra. A 2D VAE rekurrens klasszifikátor F_1-score értékei a tanító mintákon. (Sáv tartás, bal és jobb váltás – piros, kék, cián)



7. Ábra. A 2D VAE rekurrens klasszifikátor F_1-score értékei a validáló mintákon. (Sáv tartás, bal és jobb váltás – zöld, piros, szürke)

A kék görbén látható zajtól eltekintve megállapítható, hogy a veszteség egy platót ért el, további tanítással csupán a túltanulást idéztük volna elő. A 6. illetve a 7. ábrán a klasszifikáció jóságát mérő F_1-score látható az epoch függvényében, melyeken szintén megfigyelhető a telítősés. A színkódok az F_1-score ábrákon rendre a következők. A tanító mintákon piros – kék – cián szín felel meg a sávtartás – bal sávváltás – jobb sávváltás manővereknek. A validációs mintán zöld – piros – szürke.

2. Táblázat. Dekóder paraméterek

Rétegek	Csatorna Be-	Kimenet	Kernel	Stride	Padding
Trans 2/3D BatchNorm LeakyRelu	1	3	(3,3,*1*)	1	0
Trans 2/3D BatchNorm LeakyRelu	4	5	(3,3,*1*)	(1,1,*1*)	(2,2,*0*)
Trans 2/3D BatchNorm LeakyRelu	4	8	(4,4,*3*)	(2,2,*1*)	(1,2,*1*)
Trans 2/3D BatchNorm LeakyRelu	8	12	(4,4,*4*)	(1,2,*1*)	(1,2,*0*)
Trans 2/3D BatchNorm LeakyRelu	12	8	(3,8,*4*)	(1,1,*1*)	(1,1,*0*) o.p.(0,1,*0*)
Trans 2/3D LeakyRelu	8	4	(3,8,*4*)	(2,2,*1*)	(1,2,*0*)
Trans 2/3D	4	1	(2,4,*3*)	1	(1,2,*0*)

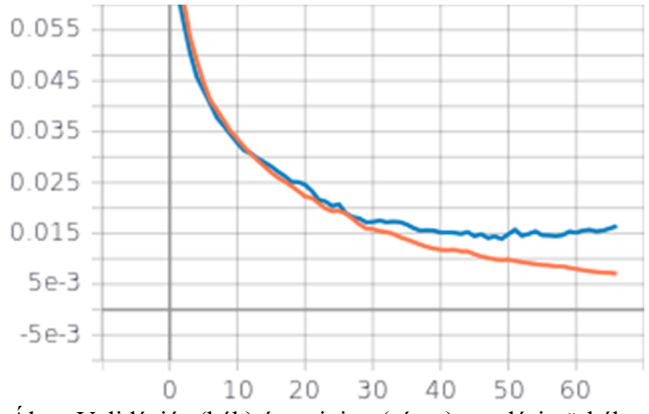
3. Táblázat. A 3D Autoenkóder és a 2D+LSTM tanítás eredmények

	Loss	F_1 sávtartás	F_1 bal váltás	F_1 jobb váltás
VAE3D Train	0.010	0,92	0,81	0,50
VAE3D Valid	0,014	0,92	0,80	0,45
Recurrent Train	0,016	0,85	0,65	0,37
Recurrent Valid	0,022	0,83	0,62	0,31

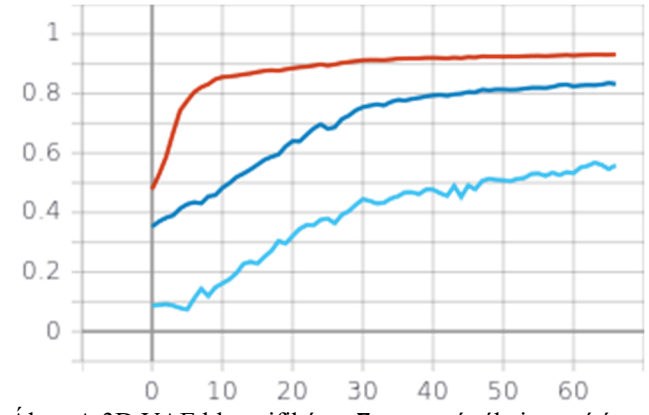
A 3 dimenziós VAE reprezentációit is felhasználtuk manőver detekcióra. Ennek a tanításnak a lényeges része az volt, hogy nem tartalmaz rekurrens hálózatot, az időbeliséged a 3. dimenziós konvolúció kezeli. A tanító és validációs halmazokon mért veszteség vüggvény értéke az epoch függvényében a 8. ábrán látható. Ebben az esetben is megfigyelhető az, hogy a kék validációs görbe eléri minimumát, majd növekedni is kezd, vagyis túltanulás következik be. Látható, hogy mindkét veszteség függvény alacsonyabb értéket vesz fel mint az előző esetben. A 9. és 10. ábra mutatja az F_1-score értékeket az epoch szám függvényében. Itt is megfigyelhető a telítődés, illetve az, hogy számottevően magasabb F_1-score értékek jellemzik mind a 3 osztályt.

Megállapíthatjuk, hogy az idő szerinti konvolúciót is magába foglaló 3D VAE alkalmasabb a manőver detekcióra a vizsgált

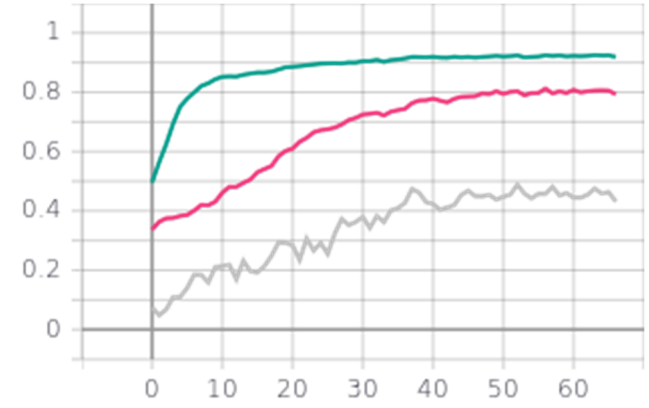
járművel együtt foglaltsági hálók idősorát felhasználva. Meg kell jegyezni, hogy az F_1-score értékek érzékenyek az adatok kiegyensúlyozatlanságára, a sávváltások esetében nagyobb hangsúly van a téves negatív eseteken, mint a helyes pozitív osztályozásokon. A 3. Táblázat tartalmazza az eredményeket összefoglalva.



8. Ábra. Validációs (kék) és training (sárga) tanulási görbék a 3D VAE klasszifikátor esetén



9. Ábra. A 3D VAE klasszifikátor $\boldsymbol{F_1}$-score értékei a tanító mintákon. (Sáv tartás, bal és jobb váltás – piros, kék, cián)



10 Ábra. A 3D VAE klasszifikátor F_1-score értékei a validáló mintákon. (Sáv tartás, bal és jobb váltás – zöld, piros, szürke)

6. KONLÚZIÓ

A 2 dimenziós Konvolúciós Autoenkóder képes jelentősen (2048 pixelről 64-re) csökkenteni a foglaltsági hálók dimenzionalitását úgy, hogy a tömörített adat idősora használható manőver detekcióra. A 3 dimenziós Konvolúciós Autoenkóder helyettesíti az idősor elemzéséhez használt rekurrens LSTM neurális hálózatot, és túl is teljesíti a manőver detekció jósága szempontjából. Emiatt könnyebben tanítható, és rövidebb idő alatt is kiértékelhető.

Az irdalomban sok féle forgalmi-környezet elemző és reprezentáló megoldás létezik. Az itt prezentált megközelítés általános, nem érzékeny a vizsgált járművet körülvevő járművek számára, mivel egy fix méretű ablakot vesz figyelembe és regisztrál minden ágenst függetlenül attól, hány van belőlük. Véleményünk szerint ez más topológiájú adatokra is kiterjeszthető megközelítés, és ez a tanulmány az NGSIM adataira alapozva nyújt koncepció igazolást.

REFERENCES

Geng, X., Liang, H., Yu, B., Zhao, P., He, L. and Huang, R., 2017. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. Applied Sciences, 7(4), p.426.

Ploeg, J. and de Haan, R., 2019, January. Cooperative Automated Driving: From Platooning to Maneuvering. In *SMARTGREENS* (p. 9).

Llamazares, Á., Molinos, E.J. and Ocaña, M., 2020. Detection and tracking of moving obstacles (datmo): A review. *Robotica*, *38*(5), pp.761-774.

Rákos, O., Aradi, S. and Bécsi, T., 2020. Lane Change Prediction Using Gaussian Classification, Support Vector Classification and Neural Network Classifiers. *Periodica Polytechnica Transportation Engineering*, *48*(4), pp.327-333.

Trautman, P. and Krause, A., 2010, October. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 797-803). IEEE.

Dagli, I., Brost, M. and Breuel, G., 2002, October. Action recognition and prediction for driver assistance systems using dynamic belief networks. In *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World* (pp. 179-194). Springer, Berlin, Heidelberg.

Lefèvre, S., Vasquez, D. and Laugier, C., 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, *1*(1), pp.1-14.

Rákos, O., Aradi, S., Bécsi, T. and Szalay, Z., 2020. Compression of Vehicle Trajectories with a Variational Autoencoder. *Applied Sciences*, *10*(19), p.6739.

Kim, B., Kang, C.M., Kim, J., Lee, S.H., Chung, C.C. and Choi, J.W., 2017, October. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 399-404). IEEE.

Deo, N. and Trivedi, M.M., 2018, June. Multi-modal trajectory prediction of surrounding vehicles with maneuver based

lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1179-1184). IEEE.

Feng, X., Cen, Z., Hu, J. and Zhang, Y., 2019, October. Vehicle trajectory prediction using intention-based conditional variational autoencoder. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 3514-3519). IEEE.

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. and Savarese, S., 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 961-971).

Deo, N. and Trivedi, M.M., 2018. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 1468-1476).

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. MIT press.

Colyar, J. and Halkias, J., 2007. US highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, pp.27-69.

Colyar, J. and Halkias, J., 2006. Us highway 80 dataset, federal highway administration (fhwa), vol. *Tech, no. Rep*.

Doersch, C., 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B., 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Blei, D.M., Kucukelbir, A. and McAuliffe, J.D., 2016. Variational Inference: A Review for Statisticians. arXiv 2016. *arXiv preprint arXiv:1601.00670*.

Kullback, S. and Leibler, R.A., 1951. On information and sufficiency. *The annals of mathematical statistics*, *22*(1), pp.79-86.

Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988).