Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Ipari adathalmazok hatékony annotálása mesterséges intelligencián alapuló módszerekkel

TDK dolgozat

Készítette:

Szécsényi Nándor

Konzulens:

dr. Lengyel László

2022

Tartalomjegyzék

Kivonat

Manapság a mesterséges intelligencia fejlődésével és terjedésével egyre szélesebb körben alkalmazzák ipari környezetben is, azonban az általa jelentett előnyökhöz gyakran problémák is társulnak. Egyik ilyen jelentkező nehézség a tanításhoz szükséges adathalmaz összeállítása, amely lényegében minták címkézését jelenti. Ezt sok esetben emberi erővel részben könnyedén el tudják végezni, azonban kisebb projektek, fejlesztések esetében nincs erre megfelelő erőforrás. Praktikusnak adódik, amennyiben ez lehetséges, ebben az esetben is a mesterséges intelligenciát, azon belül is a mélytanulást alkalmazni. Ezen cél megvalósítására a dolgozatban bemutatásra kerül kettő, a látens téren alapuló mintakiválasztási módszer, amelyek a kiértékelések alapján akár mindössze pár 100 adat annotálásával képesek megfelelő klasszifikációs teljesítményt elérni. A javasolt módszereket összehasonlítottam egy naiv, mesterséges intelligenciát nem használó, módszerrel is a verifikálás érdekében. Maga a kiértékelés több metrika alapján, 3 különböző adathalmazon történt, melyek közül az egyik ipari adathalmaz, míg a másik 2 mesterségesen előállított úgy, hogy a valóságos eseteket közelítsék. Ezen kívül a módszerekhez szükséges látens tér szerkezetét 2 különböző modellarchitektúra alapján állítottam elő, így végeredményben 12 összeállítást megvizsgálva. Az így definiált kísérleteket, modelleket 10 alkalommal feltanítva és kiértékelve egy átfogó képet kaptam arról, hogy az egyes felvázolt módszerek és eljárások milyen hatékonysággal és megbízhatósággal rendelkeznek a különböző összetettségű adathalmazok esetén, illetve milyen előnyökkel és esetlegesen előforduló hátrányokkal bírnak.

Abstract

Due to the rapid development and growth of Artificial Intelligence in recent times, it is also being used more and more frequently in industrial environments, however there are many problems besides its advantages. One of them is the difficult process of dataset construction which often involves manual sample annotation. This task is often performed by human labor at larger companies with relative ease, smaller projects or developments however usually lack the necessary resources to do so, thus making it seem advantageous to use Artificial Intelligence, more precisely Deep Learning, for this problem too. To overcome this issue and further democratize AI, in this thesis, I present 2 latent space based sampling methods, which from the acquired results are able to achieve reasonably good classification performance with only annotating a few 100s of samples. Moreover, to aid in the understanding of the results, the presented techniques are compared to a naive method representing the baseline approach. The scoring process is done using multiple metrics on 3 datasets, from which one is a genuine industry example, while the other two are artificially tailored to represent real life dataset distributions. Beyond this, the required latent space structure is produced by using 2 model architectures, thus all in all resulting in a rigorous evaluation of the presented techniques. This complex scoring includes training each model 10 times resulting in a complete picture of the introduced approaches' efficiency and confidence on the uniquely structured datasets. Furthermore, it also aids in getting to know the proposed methods' advantages and what potential problems can occur.

1. fejezet

Bevezetés

A mesterséges intelligencia, azon belül is a mélytanulás rohamos fejlődésével az iparban is egyre szélesebb körben kezdik alkalmazni, akár már az alapvető gyártási folyamatok során is: csak az elmúlt időszakban több ezer ipari mesterséges intelligencia alkalmazásokról szóló cikk jelent meg évente [1]. Ilyen gyártási lépés a minták osztályozása különböző szempontok szerint, amely kifejezetten fontos a termékek minőségének garantálásához. Praktikusnak adódik tehát itt is alkalmazni a mély neurális hálózatok képességeit. A gyakorlatban sok széles körben elérhető hálózatarchitektúra rögtön használatra kész állapotban alkalmazható egy ilyen feladat esetén, azonban ezzel a probléma még nem kerül teljesen megoldásra. További nehézséget jelent ugyanis az adathalmaz előállításának a költsége, hiszen sokszor a körülmények egyedisége miatt egy hálózatot a saját mintáinkon kell tanítani (még előtanítás mellett is) a megfelelő teljesítmény eléréséhez. Magát az adathalmazt tehát nekünk kell előállítanunk, azonban mivel érdemes minél több mintával tanítanunk, így sok-sok adatot kell felcímkéznünk az ideális teljesítmény eléréséhez. Ez általában nem jelent problémát, kisebb fejlesztések során azonban, a szükséges erőforrások hiánya miatt, a saját kezű elvégzés rengeteg időbe kerülhet rögtön a projekt elején. A dolgozatom során ezt a szükséges címkézést törekszem megkönnyíteni azzal, hogy lehetőleg minél kevesebb mintát legyen szükség kézzel annotálni úgy, hogy közben a teljesítmény az elvárt szinten maradjon.

Érdemes megjegyezni, hogy másodlagos problémaként általában az ipari adathalmazok osztályeloszlása erősen egyenlőtlen, mivel sokszor a megfelelő minőségű termékből áll a halmaz akár 99%-a, így dominálva az egészét. Erre a szakirodalomban már sokfajta megoldás létezik, amelyek közül a legkézenfekvőbb az alulreprezentált minták túlmintavételezésének technikája, az *Over-Sampling*, amelyet sokféleképpen lehet alkalmazni a gyakorlatban (lásd például [2]-t, ahol egy távolság alapú megközelítéssel alkotják meg a kisebbségi osztályok valószínűségi eloszlását, majd ez alapján választanak mintákat egy algoritmus szerint). Ehhez hasonló lehetséges megoldás és újszerű megközelítés lehet például a megerősítéses tanulás (*Reinforcement Learning*) használata, amellyel megtanulható egy adaptív eljárás arra vonatkozóan, hogy egy adattartomány mely területeiről érdemes mintákat venni a hatékonyság növelésének érdekében [3]. Ezekkel szemben ebben a dolgozatban olyan módszereket mutatok be, amelyek a felügyelet nélküli tanítást (*Unsupervised Learning*), azon belül is a kapott látens teret veszik alapul a mintakiválasztáshoz és az ott definiált tulajdonságok alapján próbálnak minél kiegyensúlyozottabb részadathalmazokat létrehozni, amelyek segítségével lecsökkenthető az annotálásra fordítandó idő és erőforrás. További fontos szempont a fentebb említett teljesítmény változásának a kérdése, ugyanis a mintaszám csökkentésével általában a hálózat osztályozó képessége romlik az adott adathalmaz esetén (lásd konkrét példaként a [4]-es tanulmányt, ahol a feladat alapkőzeti

források keresése volt), így nem elég, hogy csökkentjük a ráfordított időt, hanem azzal együtt az eredeti teljesítményt is minél jobban meg kell őriznünk.

A dolgozat felépítését tekintve az első, bevezető részben a felhasznált hálózatarchitektúrákról és azok építőelemeiről esik szó, külön kitérve az általuk képzett látens tér tulajdonságaira. Ezek után az adathalmazok leírása következik: bemutatásra kerülnek a felhasznált adathalmazok főbb jellemzői és tulajdonságai, amelyek alapján a módszereket hozzájuk igazíthatjuk. Az ismertető részek közül utolsóként a felhasznált metrikák tárgyalása történik, amelyek az elvégzett kísérletek során kerültek felhasználásra a kiértékelés megkönnyítése céljából. Az ezt követő fejezetben történik a javasolt módszerek részletes bemutatása kiegészítve egy alapszintet jelentő naiv megközelítéssel. Az utolsó előtti fejezetben kerülnek ismertetésre az elvégzett kísérletek, azok eredményei és az ezek alapján levont következtetések. Végezetül pedig egy lezáró fejezetben kerülnek összegzésre a kapott eredmények és az azok alapján levont tapasztalatok, valamint a lehetséges továbbfejlesztési irányok.

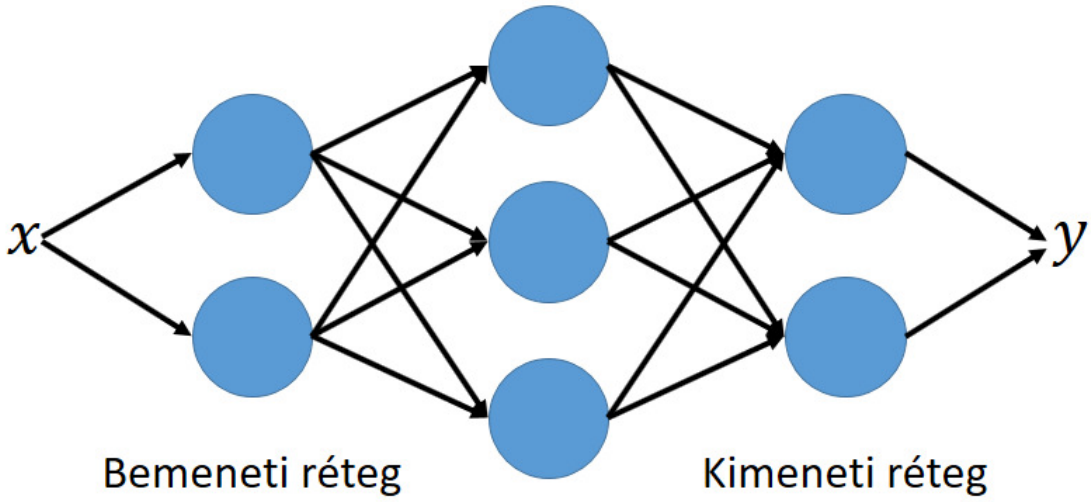
2. fejezet

Felhasznált hálózatarchitektúrák bemutatása

A fejezetben a felhasznált hálózatarchitektúrák kerülnek ismertetésre, mivel alapvetően ezek határozzák meg az elvégzett kísérletek eredményességét. Az alkalmazott neurális hálózat modelleket, a betöltött szerepük alapján, 2 kategóriára tudjuk felosztani. Az első csoportba az osztályozó feladatra használt hálózatarchitektúra tartozik, ami lényegében minden kísérleti összeállítás alapeleme: ezzel történik majd az egyes módszerek által adott adathalmazok kiértékelése. Másik kategória pedig a látens tér előállításához használt architektúrák, amelyek segítségével az eredti adathalmaz szerkezetet tudjuk egyszerűsíteni, hatékonyan tömöríteni. Ebbe a csoportba tartozik az autoenkóder és a SimCLR modell [5], amelyek más-más megközelítéssel próbálják minél pontosabban leképezni az eredeti mintapontokat a kialakított látens tér pontjaiba: az autoenkóder minél pontosabb visszaállításra törekszik, míg a SimCLR esetében a cél, hogy a hasonló minták minél közelebb kerüljenek egymáshoz a látens térben. Annak érdekében, hogy ezen hálózatarchitektúrákról hatékonyabban beszélhessünk, a fejezet elején bemutatásra kerülnek a neurális hálózatok alapvető építőelemei.

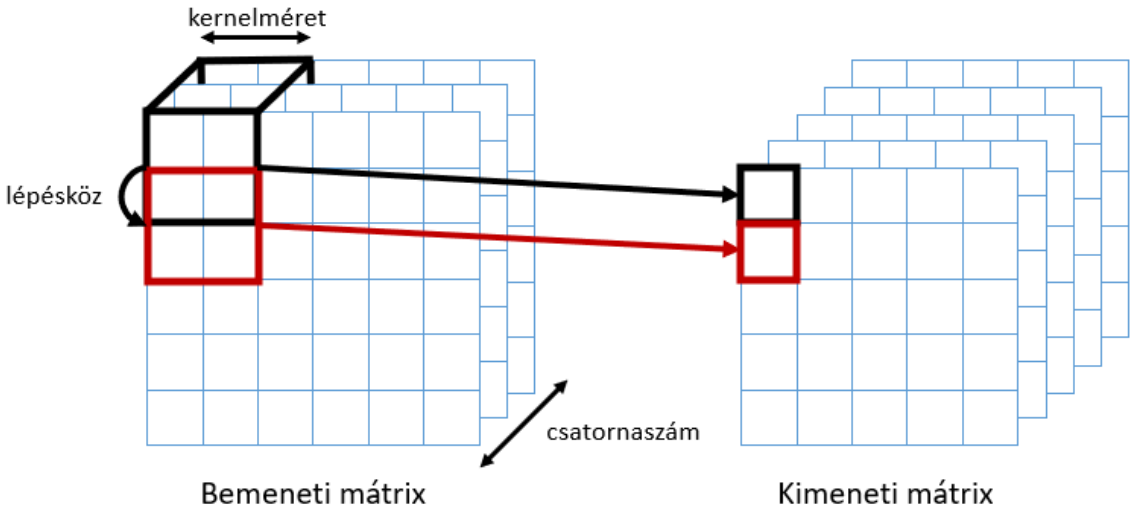
2.1. Alapvető rétegtípusok bemutatása

A neurális hálózatok alapelemei a neuronok, amelyek elnevezésüket az emberi idegsejtek után kapták, hiszen képesek információt továbbítani megadott útvonalon egy adott csomóponttól egy másik felé: kezdetben éppen ez a biológiai működés ösztönözte a kutatókat a legelső architektúrák megalkotására (perceptron, az egy neuronból álló modell [6]). Lényegében egy neuron tekinthető egy olyan függvénynek, amely a bemenet (x) és egy belső súlyváltozó (w) segítségével állít elő egy kimenetet (y) úgy, hogy a súly értékét változtatva a kimenet értéke is változzon. A felépítéséből következik, hogy könnyedén képesek vagyunk több ilyen neuront egymás után helyezni, így kialakítva az általuk reprezentált függvények kompozícióját. Ha kiegészítjük a neuronok kimenetét egy nemlineáris aktivációs függvénnyel (azaz a teljes függvénykapcsolat: $z = f(y) = f(wx)$), akkor belátható, hogy már nem csak lineáris függvényeket tudunk megvalósítani, hanem akár tetszőleges függvényalakokat is, ráadásul minél több neuront helyezünk egymás után, annál komplexebb feladatokat tudunk megoldani. A következő lépés a teljesen összekötött rétegek (*Fully Connected Layer*) bevezetése, amelyek esetében több különböző neuron kapja meg ugyanazt a bemenetet, valamint minden neuron kimenete össze van kötve a következő réteg összes bemenetével, így egy sűrű (*Dense*) hálózatot eredményezve, amely felépítését a 2.1 ábra szemlélteti.



2.1. ábra. Teljesen összekötött neurális hálózat szemléltető ábrája

A mély tanulás (*Deep Learning*) fejlődésével egyre több rétegből álló hálózatokat kezdtek el alkalmazni, így növelve azok teljesítményét, illetve létrejöttek a teljesen összekötött rétegen kívül más, adott feladattípusra specifikált rétegtípusok is. Ilyen például a konvolúciós réteg, amely kifejezetten képfeldolgozási feladatokra került kidolgozásra, hiszen segítségével a több dimenziós bemenet helyinformációi nem vesznek el azzal (megőrizzük a pixelek közötti lokális koherenciát), hogy egy egyszerű vektorrá alakítjuk, hanem egy adott méretű ablakot, kernelt csúsztatunk végig a bemeneti képmátrixon, így képezve egy általunk meghatározott méretű kimeneti tenzort[1]. Maga a réteg sok paraméterrel rendelkezik a kimeneti tenzor méretének meghatározása és az értékek képzése szempontjából, amelyek közül a legfontosabbak a 2.2 ábrán láthatóak. A konvolúciós réteget és az abból felépülő hálózatokat a dolgozat során szinte minden modellben felhasználtam, hiszen segítségével könnyen lehet kevés paraméterű (a réteg paraméterszáma nem függ a bemenet térbeli méretétől), jól teljesítő képosztályozó neurális hálózatokat megvalósítani.



2.2. ábra. Konvolúciós réteg szemléltető ábrája

[1]A tenzor lényegében a mátrix 3 dimenziós általánosítása, a mesterséges intelligenciában gyakran előkerülő fogalom, ugyanis maguk a bemeneti képek is 3D-sek: szélesség×magasság×színcsatornák száma

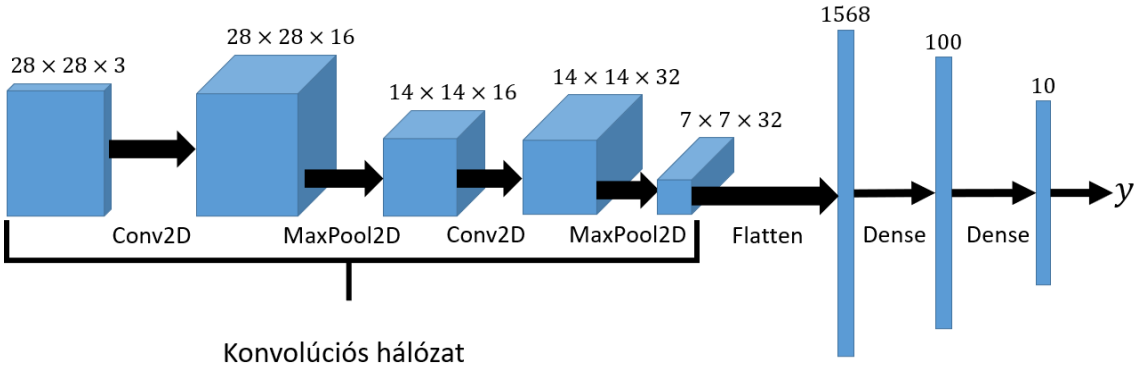
Az előzőekben röviden bemutatott 2 rétegtípuson kívül, ahogy korábban az említésre került, számtalan más, specifikus réteg létezik (ezekre példát az egyes architektúráknál láthatunk), azonban a dolgozat során felhasznált hálózatok, modellek fontosabb rétegei a teljesen összekötött vagy a konvolúciós rétegek valamelyike.

2.2. Osztályozó hálózat

A tárgyalt hálózat egy általános osztályozási feladatra kialakított architektúra: célja, hogy a bemenetre adott képet besorolja az előre megadott kategóriák valamelyikébe.

Első lépésként a bemeneti mintát egy sor konvolúciós blokkon keresztül engedjük át, amelyeknek a feladata a fontosabb információk kinyerése úgy, hogy közben az adat méretét lecsökkentse. Ezt úgy tudjuk hatékonyan elérni, hogy minden konvolúciós réteg után egy *Batch Normalization* és egy *Max Pooling* réteget helyezünk. Ezek normalizálják az adatok eloszlását az aktiváció előtt, illetve kiválasztják egy adott ablakon belül a legnagyobb, azaz a maximális aktivációs értéket kiváltó adatpontokat, eldobva a többit, így egyidejűleg csökkentve az eredeti méretet és megőrizve a lényeges információkat. Ezután a megfelelő méret elérésekor sorosítjuk a kapott tenzort azzal, hogy kiterítjük az egyes sorait és egymás után illesztjük őket (*Flatten Layer*). Magát a vektorosítást végezhetjük egy *Global Pooling* réteg segítségével is, amely a megadott csatornák mentén választja ki például a legnagyobb elemet vagy az elemek átlagát képezi. Erre a transzformációra azért van szükség, hogy az ezek után következő teljesen összekötött rétegek -amelyek az osztályozást fogják elvégezni- bemenetként vektorokat várnak nem pedig tenzorokat, szemben a korábban alkalmazott konvolúciós rétegekkel.

A fentebb bemutatott architektúrára mutat egy példát a 2.3 ábra, amelyen fel lettek tüntetve az egyes rétegek bemenetének méretei is, így segítve megérteni a hálózat működését. Fontos megjegyezni, hogy az ábrán látható konvolúciós hálózat első részében szereplő blokkok ismétlődhetnek többször is, a példában egyszerűen a bemeneti képek kis mérete miatt nem volt célszerű tovább csökkenteni a tenzorokat.



2.3. ábra. Egyszerű osztályozó hálózat szemléltető ábrája tenzor méretekkel

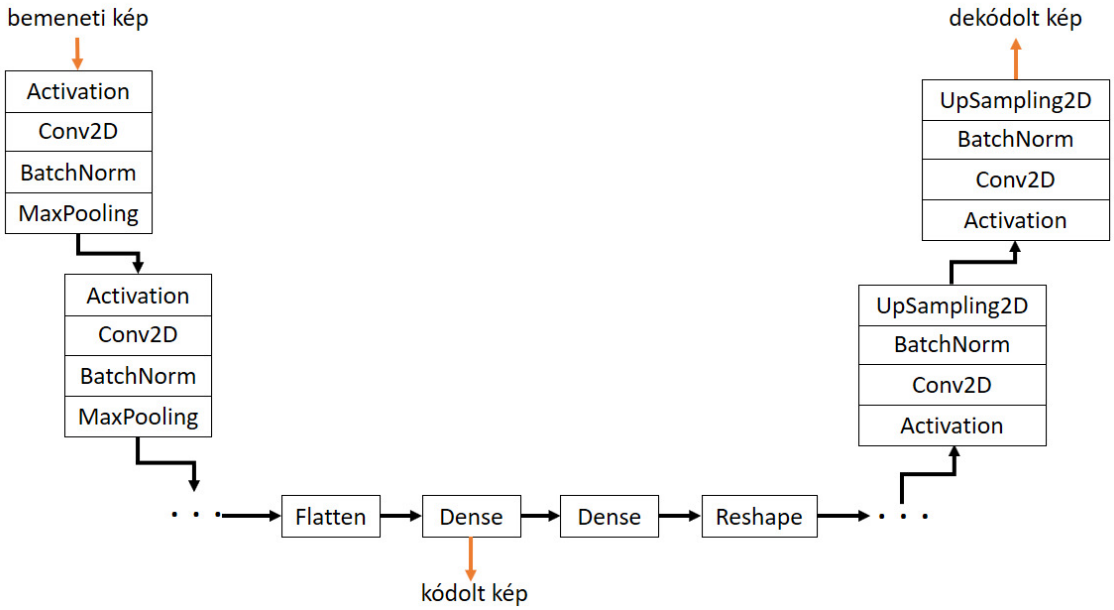
2.3. Autoenkóder struktúra és látens tere

Az előző részfejezetben bemutatott architektúra alkalmas volt a bemeneti képek osztályozására, azonban bizonyos esetekben nem áll rendelkezésünkre az adathalmaz tényleges címkézése, így enélkül kell megpróbálnunk az egyes kategóriákat megkülönböztetni, majd pedig a mintákat szétosztani. Ezt a feladattípust a mélytanulás területén felügyelet nélküli

tanításnak (*Unsupervised Learning*) nevezzük, és a másfajta feladatból adódóan különleges megközelítést kíván el a hálózat struktúrája, felépítése szempontjából.

A dolgozat során a fent említett feladatra a módosított U-Net struktúrán[7] alapuló autoenkóder architektúrát alkalmaztam, amely hatékonyan képes a bemenetére adott képekből kinyerni a szükséges információkat, amik alapján osztályozni lehet őket. Maga a hálózat két részből áll: egy enkóder és egy dekóder részmodellből, amelyek összekapcsolódva alkotják a teljes modell szerkezetét. Az enkóder rész feladata a bemeneti kép fontosabb információinak kinyerése méretcsökkentéssel egymás utáni konvolúciós blokkok segítségével, így előállítva egy kódvektort, amelyet tekinthetünk egy köztes kimenetként. Az így kapott kódolás, melynek hosszát a megelőző *Dense* réteg neuronszáma határozza meg, tartalmazza a hálózat által a képből kinyert és -a költségfüggvény szempontjából- fontosnak tartott tulajdonságokat, jellemzőket. Ahhoz, hogy meg tudjuk határozni a kód hatékonyságát, fel kell használnunk a dekóder modul kimenetét, amely lényegében az enkóder által elvégzett tömörítést fordítja vissza, azaz megpróbálja csak a kódolás alapján előállítani a bemeneti képet. Ahhoz, hogy ezt elérjük, célszerű a hibafüggvényt az eredeti és az így előállított kép különbségeként definiálni, és ez alapján frissíteni a hálózat súlyait addig, amíg nem lesz megfelelő a kódolás minősége annyira, hogy kis hibával képes legyen a dekóder visszafejteni belőle az eredeti információt.

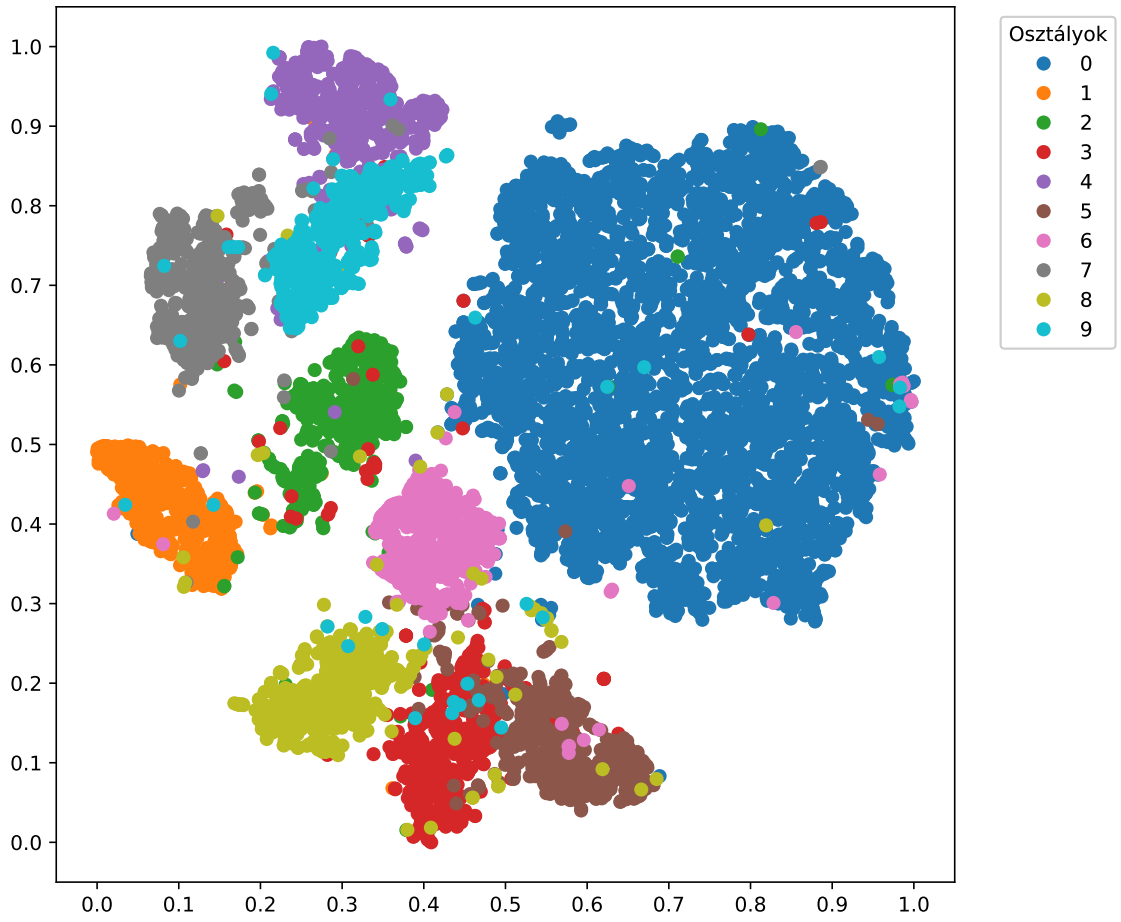
A fentiek szemléltetésére szolgál a 2.4 ábra, ami bemutatja az autoenkóder struktúra vázlatos felépítését, illetve látható rajta a kód képzésének menete és a különböző rétegtípusok nevei. A hálózat alakja és a leírás alapján könnyen érthető az elnevezése is: a lefelé és felfelé terjesztő ágak a betű két szárát, míg a kódolást végző rész az aljának felel meg, így egy „U” betűre hasonlítva.



2.4. ábra. Autoenkóder struktúra szemléltető ábrája

Alapvetően az autoenkóder struktúra előnye, hogy akár egy címkézetlen adathalmaz esetén is képes a kódolás előállításával valamilyen osztályozást, szétválasztást elvégezni a minták között, amelyet a látens tér elemzésével tudunk megvizsgálni. A látens tér alatt az enkóder kimeneteként előálló kódteret kell érteni, amely a kódszóhosszal megegyező dimenziós tér, amelyben az egyes bemeneti mintákhoz tartozó kódvektorok egy-egy pontnak feleltethetőek meg a kapott koordinátáik szerint. Az így kapott térben tehát minden egyes mintának egy kitüntetett pont feleltethető meg és az egyes minták egymáshoz való viszo-

nyát tükrözi a látens térben elfoglalt helyzetük. Például, ha két minta egymástól merőben eltérő (a költségfüggvény tekintetében), akkor egy jó kódolás őket minél távolabb helyezi, szemben két, vélhetően azonos osztályba tartozó, így vizuálisan is hasonló mintapárral, amelyeket feltehetően közel tesz egymáshoz. Ezt a jelenséget szemlélteti a 2.5 ábra, ahol az MNIST adathalmaz 10000 mintájának (lásd 3.1 fejezet) egy feltanított autoenkóder által létrehozott kódterét ábrázoltam a t-SNE algoritmus[8] segítségével, amely lehetővé teszi egy részletesen paraméterezhető transzformációval[9] a sokdimenziós látens tér ábrázolását 2 dimenzióban arra törekedve, hogy lényeges információveszteség és torzulás ne történjen a dimenziószám csökkentésekor. Láthatóak a fentebb említett jelenségek: az egyes számjegyek (most előre ismertek a tényleges címkék) egymáshoz közel helyezkednek el, míg a többi csoporttól különállóan kisebb átfedésektől eltekintve, valamint bizonyos pontok nem tartoznak egy csoporthoz sem, kevés szomszédjuk van, tehát tekinthetőek *outlier* pontoknak.

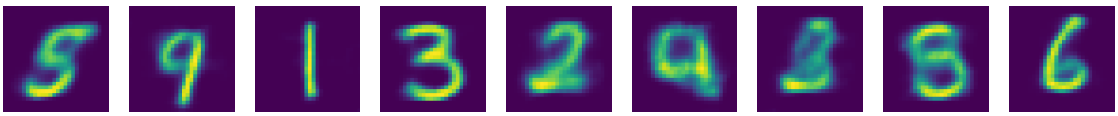


2.5. ábra. Autoenkóder látens terének ábrázolása t-SNE segítségével az MNIST adathalmazon

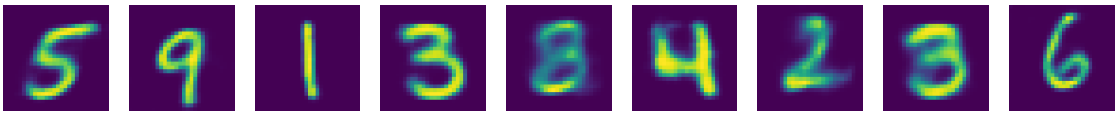
Így a látens teret vizsgálva találhatunk több közeli mintából álló pontfelhőket, amelyek valószínűleg a hasonlóságuk alapján azonos osztályba sorolandók. Ezen kívül látható a többi ponttól sokkal távolabb eső, szomszédokkal nem rendelkező adatpontokat, amelyek valószínűleg túlságosan egyedi, esetleg zajos minták (*Outlier Samples*). Ezeket nem célszerű figyelembe venni az osztályok kialakításakor, mivel az egyediségük rontaná az adott csoport egységét. Tehát a fentiek alapján a célunk az egyes pontokat közelség és szomszédok száma alapján csoportosítani különböző klaszterekbe és így kialakítani egy

lehetséges osztályozást, amire a gyakorlatban rendkívül sokféle algoritmus és módszer létezik, amik mind különböző megközelítést alkalmaznak a probléma megoldására (ezeknek egy lehetséges összehasonlítása itt olvasható [10]).

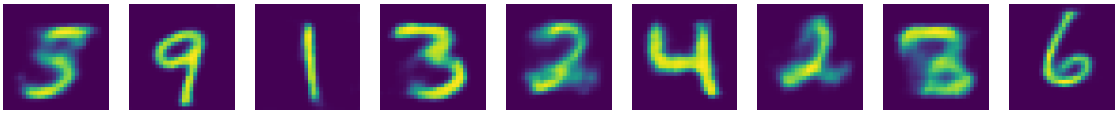
Fontos kérdés még az autoenkóder kódszóhosszának megválasztása: ez lényegében azt jelenti, hogy az enkóder által előállított kódvektor hány elemet tartalmazzon (ezt a *Dense* réteg neuronszáma határozza meg), illetve ezzel összefüggésben mekkora legyen a kapott látens tér dimenziószáma. Alapvetően célszerű minél tömörebb kódolást választani, mégis egy bizonyos hossz után már túl sok információ vész el ahhoz, hogy a dekóder hatékonyan vissza tudja állítani a látens vektor alapján az eredeti adatot. Ezen kívül felső határt adhat a kódszónak a lehetséges kategóriák, osztályok száma, ugyanis például az MNIST adathalmaz esetén, ha minden mintához egy 10 hosszú vektort rendelnénk, akkor lényegében az enkóder modellünk egy osztályozó hálózatot valósít meg, így lehetséges, hogy az ennél nagyobb neuronszám alkalmazása esetén a kódunk egy része a feladat szempontjából felesleges információk tárolására kerül felhasználásra. Vannak azonban esetek, ahol az egyes előre definiált osztályok jellemzői nincsenek teljesen meghatározva, megfelelően körülírva: ilyenkor érdemes lehet az osztályok számánál nagyobb kódszóhosszat választani, hiszen ekkor a kisebb különbségeket jobban ki tudja hangsúlyozni a hálózat, ezzel segítve az osztályszerkezet módosítását, pontosabb meghatározását. A fentiek szemléltetésére szolgál a 2.6 ábracsoport, ahol a neuronszám és így a kódszóhossz értékének hatását lehet végigkövetni a visszaállított képeken (itt most más színskálával a részletek jobb láthatósága miatt).



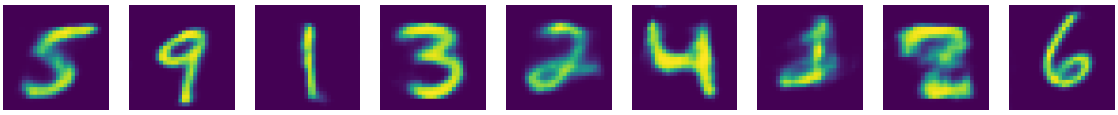
(a) $N = 4$



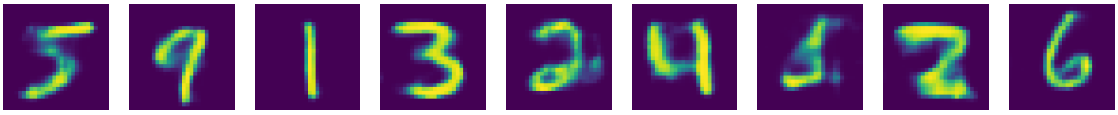
(b) $N = 8$



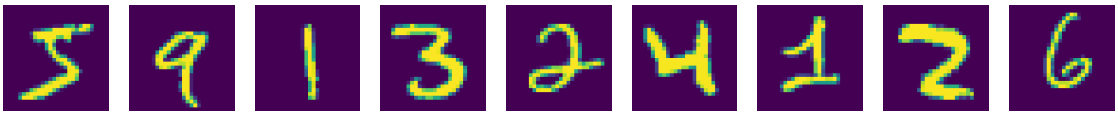
(c) $N = 10$



(d) $N = 16$



(e) $N = 32$



(f) Eredeti minták

2.6. ábra. N kódszóhosszú autoenkóderek visszaállított mintái

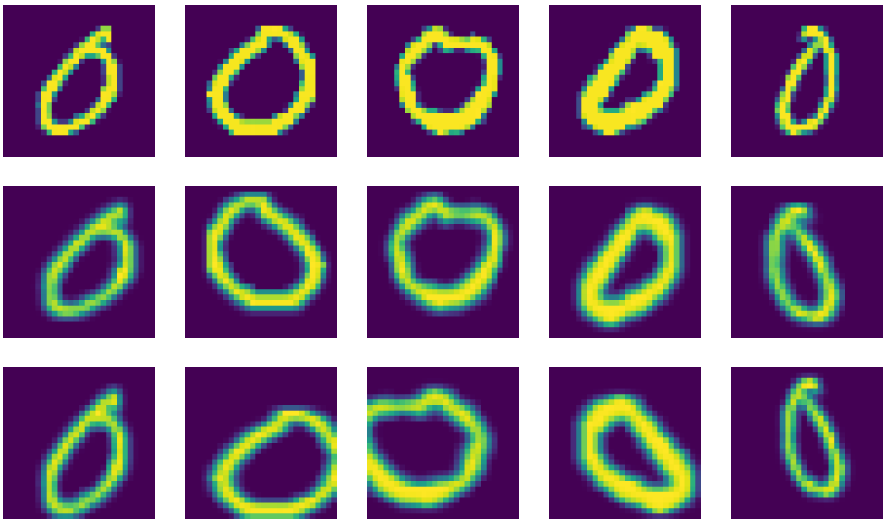
Az ábrák alapján látható, hogy míg a kisebb neuronszámú (2.6a és 2.6b ábrák) autoenkóderek visszaállított képei közül sok esetben csak a főbb kontúrvonalak láthatóak tisztán és a többi, finomabb vonás elmosódik (pl. 2-es és 3-as számjegyeknél), addig a kódszóhossz növelésével az eltárolt információ megnövekedése miatt a hálózat az apróbb részletekre is képes nagyobb figyelmet fordítani: például a 2.6d és 2.6e ábrákon az 5. és a 7. minta kezd kitisztulni, a szokatlan vonásaik is láthatóak a visszaállított mintákon.

2.4. SimCLR hálózat és látens tere

Az utolsóként bemutatott hálózatarchitektúra a SimCLR [5] (Simple Framework for Contrastive Learning of Visual Representations), amely az autoenkóderhez hasonlóan egy olyan látens térbe képezi le az egyes mintákat, ahol a hasonló adatok egymáshoz közel és a különbözőek pedig minél távolabb helyezkednek el, azonban ehhez más megközelítést és költségfüggvényt használ fel.

A fenti cél eléréséhez a SimCLR egyik legfontosabb megközelítése a bemenő adatok transzformációja, amely során lényegében egy adatmintából képi transzformálások során 2 darab új mintát hozunk létre, amelyeket bemenetül adva egy enkóder hálózatnak egy-egy leképezést kapunk. Itt fontos megemlíteni, hogy ugyan célszerűnek tűnhet elsőre minél

többféle transzformálás közül választani a hatékonyabb kódolás elérése érdekében, azonban szem előtt kell tartani az eredeti adathalmaz szerkezete és felépítése jelentette korlátokat. A gyakorlatban ez azt jelenti, hogy nem érdemes olyan képi transzformációt elvégezni, amely alapján esetleg olyan mintát kaphatunk, amely valóságos körülmények között nem fordulhat elő. Ilyenkor fel kell tennünk a kérdést: egy ily módon megváltoztatott minta rendelkezik-e az eredeti osztályának összes tulajdonságával? Példaként egy 6-os számjegyet ábrázoló képen végezhetünk forgatást, azonban csak egy bizonyos mértékig, hiszen „fejre fordítva" már alaphelyzetben más osztályba tartozó mintát kapunk, így feleslegesen nehezítve a hálózat tanulását. Ezek alapján például az MNIST adathalmazon elegendő a véletlen vízszintes tükrözés, transzláció és nagyítás transzformációk kombinációjának végrehajtása, amelyekre példát a 2.7 ábra mutat, ahol az első sor az eredeti, a második a gyengén és a harmadik az erősen augmentált mintákat tartalmazza.



2.7. ábra. Augmentált példaminták az MNIST adathalmazból

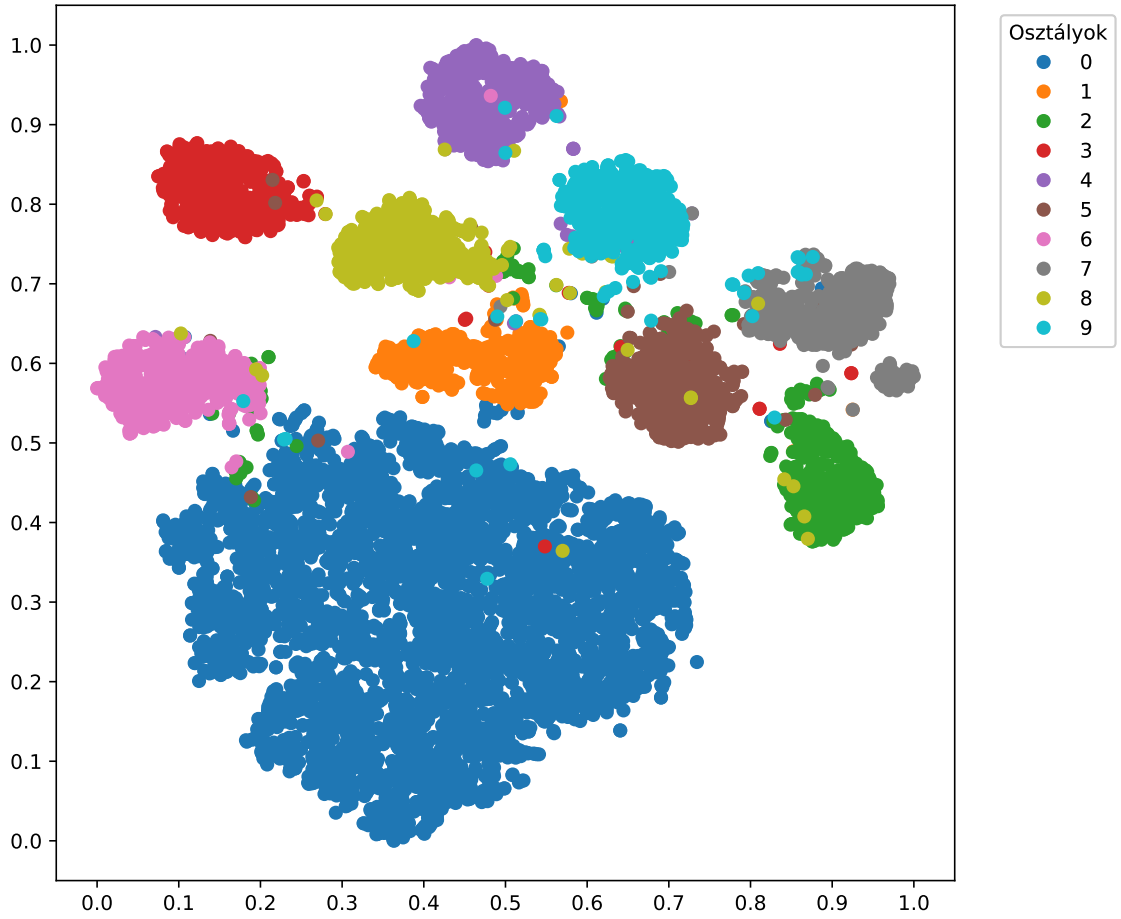
Ezeken a mintákon látható, hogy ugyan változtatható a transzformációk mértéke, súlyossága, a megadott korlátok miatt mégsem hozunk létre egyszer sem olyan mintát, ami normál esetben nem fordulhatna elő.

N mintát kiválasztva és rajtuk elvégezve a leírt transzformációt a kapott $2N$ mintapárban szerepel minden eredeti adathoz egy $\mathbf{x}_i$ és $\mathbf{x}_j$ mintapár úgy, hogy azok az eredeti mintának különböző reprezentációi: ezeket pozitív mintapárnak nevezzük, a csoport (*batch*) többi mintájának e kettővel képzett párja pedig negatívak párnak tekinthető. Tanítás során tehát ennek a 2N mintának a lehetséges kombinációját vizsgáljuk meg és értékeljük a pozitív párokat egy kontraszív költségfüggvénnyel, melynek a célja az egyes párok tagjainak megkeresése. Ez a költségfüggvény egy olyan keresztentrópia, amelyet egy előre megadható együtthatóval skálázhatunk (ezzel súlyozható az egyes mintapárok tanulhatósága, „nehézsége"):

$$l_{i,j} = -\log \frac{\exp(\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq 1]} \exp(\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_k)/\tau)} \tag{2.1}$$

ahol $\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\intercal \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$ a két minta hasonlósága, τ a változtatható együttható és $\mathbb{1}_{[k \neq 1]} \in \{0, 1\}$ egy indikátor függvény, ami 1-et ad ha $k \neq i$, különben 0-t.

A fentieken túl alkalmazhatunk az enkóder hálózat után egy egyszerű projekciós modellt is (ezt később elhagyhatjuk a hálózat végéről), amely kimenetein a tapasztalatok szerint hatékonyabban alkalmazható a definiált minimalizálandó költségfüggvény. Összességében tehát a SimCLR arra törekszik, hogy az egyes mintákon csak a valóban szükséges ismertető jeleket tanulja meg az enkóder hálózat a véletlenszerű képtranszformálások miatt és képes legyen minél jobban megkülönböztetni a különböző osztályba tartozó mintákat. Ezt szemlélteti a 2.8 ábra, ahol az MNIST adathalmazon feltanított SimCLR struktúra látens terének t-SNE ábrája látható.



2.8. ábra. Az MNIST adathalmazon feltanított SimCLR hálózat látens terének t-SNE ábrája

Az ábra alapján látható, hogy valóban sikerült egy jól strukturált látens teret kapnunk, ahol az egyes kialakult klaszterek viszonylag különállóak és csak az adott osztály mintáit tartalmazzák. Jól szemlélteti a SimCLR hatékonyságát az autoenkóder struktúránál kapott t-SNE ábrával való összehasonlítás (lásd 2.5 ábra), amely még helytállóbb amiatt, hogy a SimCLR enkóder hálózata szinte teljesen megegyezik az autoenkódernél alkalmazottal. Ezek alapján elmondható, hogy az autoenkóder látens teréhez képest javulás értünk el mind a klaszterek összetétele, mind az egymáshoz képesti elhelyezkedésük alapján.

3. fejezet

Az adathalmazok bemutatása

Ebben a fejezetben a kísérletekhez felhasznált adathalmazok kerülnek bemutatásra, hiszen már az alapvető tulajdonságaikból is lehet következtetni az esetlegesen várható eredményekre, nehézségekre. Az első két adathalmaz a gyakorlatban osztályozó neurális hálózatok tesztelésére széles körűen használt adathalmazok, míg a harmadik egy ipari adathalmaz, amely mintáin gyártás során keletkező hibák keresése a feladat. Mivel a dolgozat szerves része a látens tér szerkezetének módszerekkel történő elemzése, ezért az adathalmazokat ebből a szempontból is megvizsgáljuk, illetve összehasonlítjuk egymással, így egy viszonylag intuitív nehézségi sorrendet felállítva közöttük.

3.1. Az MNIST adathalmaz bemutatása

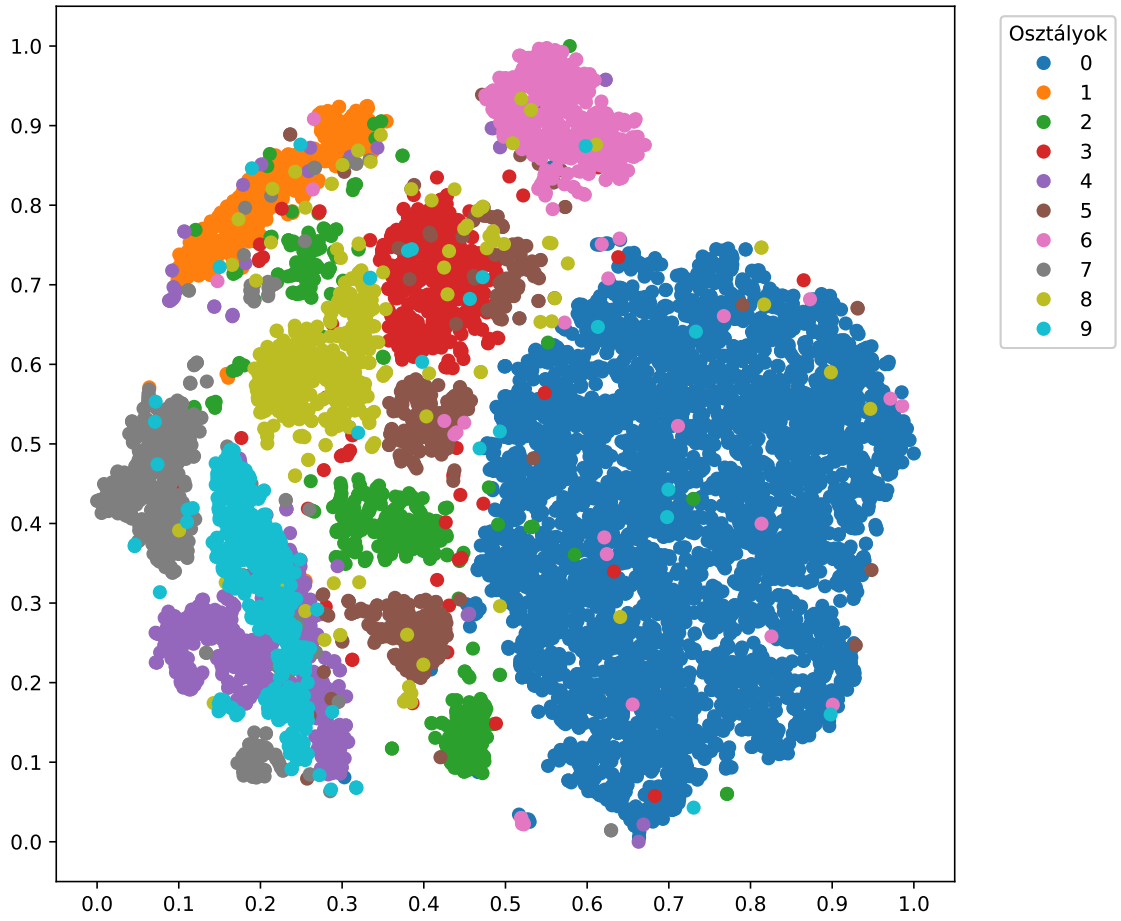
Mivel a kapott metrikák értékelése szempontjából előnyösebb egy egyszerűbb, „könnyebb” adathalmazt választani -ekkor használhatunk kisebb hálózatokat, amik tanítása gyorsabban elvégezhető-, ezért elsőként az MNIST adathalmazt [11] választottam, ami a mesterséges intelligencia, azon belül is a mélytanulás témakörében máig népszerű a kora ellenére az egyes modellek teljesítményének szemléltetésére, működésük kipróbálására.

Történetét tekintve az MNIST adathalmazt a kezdetektől fogva a mesterséges intelligencia fejlesztésére, tanítására használták: postai csomagokon kézzel felírt címek számjegyeinek felismerése volt az eredeti feladat. Ebből adódóan az adathalmaz mintái kézzel írt számjegyekről készített képekből állnak, amelyekhez tartozik egy-egy címke a mintán szereplő számjegy értékével, így segítve az osztályozás megkezdését. Mindegyik minta fekete-fehér (egy színcsatornás), méretük 28×28, így valóban egy egyszerűen kezelhető adathalmaz, amelyen könnyen érhető el jó eredmény egyszerűbb hálózatstruktúrák használata esetén is. Ezt szemlélteti a 3.1 ábra, ahol 0-tól 9-ig látható egy-egy példa minta a számjegyekre.

3.1. ábra. Példaminták az egyes számjegyekre az MNIST adathalmazból

Maga az adathalmaz $60,000$ mintát tartalmaz, melyből alapértelmezetten $50,000$ a tanító, $10,000$ a validációs halmaz része. A kísérletek elvégzéséhez a tanító minták közül kiválasztásra kerül $10,000$ minta úgy, hogy az így kapott részhalmaz egy ipari adathalmaz szerkezetéhez hasonlítson: a Pareto-eloszlásnak egy egyszerűsített esetét (valójában ez az $\alpha = \log_4 5 \approx 1.16$ alakparaméterű eloszlás) követve az egyik osztály 80%-os súlyt kap (ez felel meg a megfelelő kategóriának), míg a többi számjegy $20 - 20\%$-os részt képvisel (ezek a különböző hibás minták). Ez a megközelítés abból a szempontból is előnyös, hogy így a hibás minták varianciája nagy lesz, ami ugyancsak közelíti a valóságos körülményeket. A feltanított hálózatok értékeléséhez a validációs adathalmazt használjuk fel, azon nem változtatunk, így abban az egyes osztályok eloszlása egyenletes (minden osztály egyenlő súllyal szerepel), ezzel kihívást jelentve a később definiált mintakiválasztási módszerek számára.

A képzett halmaz látens terét tekintve ugyancsak szembetűnő az adathalmaz szerkezetének egyszerűsége, hiszen csak magán a nyers adatokon egy t-SNE transzformációt elvégezve is egy viszonylag jól strukturált elrendezést kapunk, amely a 3.2 ábrán látható.



3.2. ábra. A képzett MNIST adathalmaz közvetlen t-SNE transzformáltja

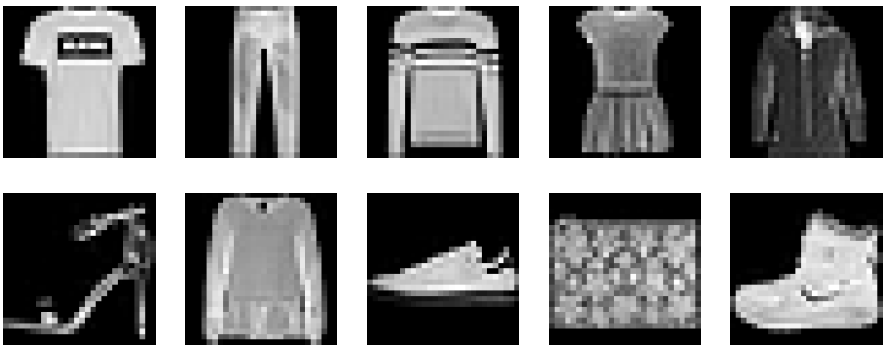
A látens tér ilyenfajta rendezettsége az egyes minták kinézetéből következik: az összes számjegy azonos, egyszínű háttér előtt van, így az ezt kódoló adatsorból könnyen kihagyható lényegesebb információ veszteség nélkül, ezzel csökkentve a szükséges kódhosszat. Ezen kívül maguk az egyes osztályok halmazai is nagyrészt sűrű mintacsoportokból (egymáshoz a térben közeli pontok) állnak, így egy dimenziócsökkentő vagy egy klaszterező algoritmus ezt a tulajdonságot könnyedén kihasználhatja egy jobb eredmény elérésben. Egyedüli nehézségként talán az osztályokon belüli *outlier* mintákat tekinthetjük, amelyek egy-egy számjegy (a többihez képest) „szokatlan” írásmódjából erednek, azonban ezek száma viszonylag csekély, így nem befolyásolja jelentősen a végeredményt.

A fentiek alapján tehát kijelenthetjük, hogy a módosított MNIST adathalmaz egy megfelelő viszonyítási alapot képez a kísérleteink elvégzéséhez, hiszen egyszerűen kezelhető, illetve viszonylag könnyedén tudunk jó minőségű látens teret előállítani hozzá.

3.2. A Fashion-MNIST adathalmaz bemutatása

Az előző bevezető adathalmaz után következzen egy valamivel összetettebb, mégis hasonló felépítésű adathalmaz tárgyalása: a Fashion-MNIST [12]. Az előző részben ismertetett halmazzal való egyértelmű összeköttetés miatt az elemzés során sokszor érdemes párhuzamot vonni a kettő között, így még jobban hangsúlyozva a számunkra lényeges tulajdonságokat.

Nevéből eredően a Fashion-MNIST adathalmaz is azonos szerkezetű, mint az MNIST: 10 osztályból áll és a minták 28×28-as fekete-fehér képek. A különbség abból ered, hogy itt már nem számjegyek, hanem különböző ruhadarabok vannak egy internetes oldalról kategória szerint kigyűjtve. Ezen ruhaosztályokra egy-egy példa minta a 3.3 ábrán látható.



3.3. ábra. Példaminták az egyes ruhadarabokra a Fashion-MNIST adathalmazból

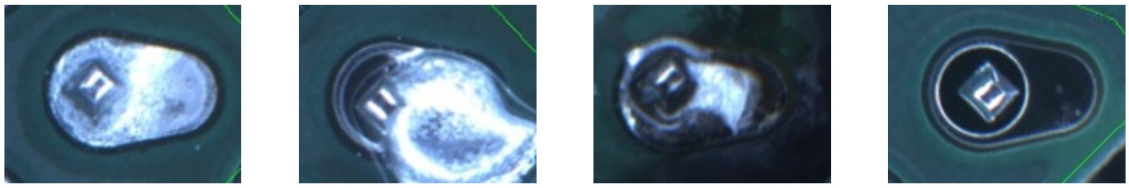
A fenti adatokon is megfigyelhető, hogy az így definiált osztályozó feladat nehezebb, mint a számjegyek esetében, hiszen ugyan az egyes kategóriák rendelkeznek ismertető jegyekkel, mégis minden ruhadarab alapvetően más és más stílusú, alakú, akár egy azonos kategórián belül is. Ez alapvetően megnehezíti a feladatunkat, valamint emiatt itt már nem lehet egyszerűen a tiszta adatokat egy jól strukturált látens térre leképezni. Ennek ellenére mivel a háttér minden mintánál azonos, valamint csak egy színcsatornával kell dolgoznunk, így csak egy szinttel komplexebb a probléma, mint az MNIST esetében. Jól mutatja ezt az is, hogy ezen az adathalmazon is lehet viszonylag kevés paraméterű, egyszerű szerkezetű neurális hálózattal jó eredményt elérni, habár az MNIST-hez képest növelni kell a modell paraméterszámát a jobb teljesítmény eléréséhez. Az előző adathalmazhoz hasonlóan itt is az ipari körülmények szimulációja érdekében egy olyan részhalmaz kerül felhasználásra, amely $10,000$ mintát tartalmaz az $50,000$ tanító adatból úgy, hogy az egyik osztály a halmaz 80%-át, a többi $20 - 20$%-át alkotja. Emellett a $10,000$ mintából álló validációs adathalmazt ebben az esetben is változtatások nélkül használjuk fel az egyes modellek kiértékeléséhez.

3.3. A forrasztási képek adathalmazának bemutatása

Végezetül pedig vizsgáljuk meg az ipari adathalmazunkat, amely autóipari gyártás során keletkező forrasztási képekből áll. Ez képezi a harmadik lépcsőfokot az adathalmazok nehézségi sorrendjében, hiszen ez egy valóságos ipari adathalmaz, valamint emiatt, ahogy látni fogjuk, a minták tulajdonságai közül is máson lesz a hangsúly, mint az eddig tárgyalt két halmaz esetében.

Alapvetően az adathalmaz célja, hogy meg tudjuk különböztetni a hibás forrasztásokat a hibátlanoktól, valamint szeretnénk megmondani a hiba fajtáját is. Ezek alapján 4 minőségi osztályt különböztetünk meg: a megfelelő forrasztásokat, a kevés vagy a túl sok pasztát tartalmazóakat, valamint a megégetett mintákat. Összességében ezek közül a hibátlan forrasztásokból van a legtöbb, míg az egyes hibaosztályok mintaszáma csekély. Elsőként vegyük a kevés forraszpasztát tartalmazó mintákat: itt a paszta hiánya abból

látható, hogy az nem tölti ki a teljes forrasztási furatot, egyes részeken feketén látszik a furat. Ennek ellentéte a túl sok paszta esete, ahol az egyszerűen felpúposodik és túllépi a megadott határokat, akár el is folyik, ezzel esetlegesen rövidzárat okozva. Végezetül pedig vannak a megégetett lemezek, ahol a forrasztás környékén fekete égés nyomok láthatóak, gyakran magán a forraszpasztán is. Ezekre egy-egy példa a 3.4 ábrán látható.



3.4. ábra. Példaminták az egyes minőségbeli osztályokra a forrasztási pontok adathalmazából

Maguk a képek az előző halmazok mintáit mind méretben (96×128), mind a színcsatornák számában (színes képek) felülmúlják, azonban ezt ellensúlyozza, hogy alapvetően a háttér és a lényeges terület viszonylag állandó (a nyák és a forrasztás helye, színe kevésbé változik), így hatékonyan lehet tömöríteni a rendelkezésre álló adatokat információ veszteség nélkül. Ez kifejezetten előnyös a látens tér előállításánál, hiszen az lenne a cél, hogy egy olyan kódteret hozzunk létre, ahol a különböző osztályok elkülönülnek, azaz sikeresen megtaláltuk az egyes kategóriák jellemző tulajdonságait. Ezekből adódóan ugyan nagyobb paraméterszámú modellt kell használnunk a megfelelő eredmény eléréséhez, mégis alapvetően az osztályozás feladatára elegendő tud lenni az egyszerű konvolúciós struktúra. A kísérletekhez használt adathalmaz itt a valós viszonyokat tükrözi azzal, hogy a jó forrasztások alkotják a halmaz több mint felét, míg ehhez képest a többi (hibás) minta száma sokkal kisebb. A validációs adathalmaz ebben az esetben csak 1000 mintát tartalmaz a kis mintaszámú hibaosztályok miatt, azonban itt is az egyes kategóriák egyenlő súllyal szerepelnek.

Összegezve a fentebb leírtakat, jogosan tekinthető a 3 adathalmaz közül a forrasztási képeket tartalmazó a legnehezebbnek, mégis a szerkezete okozta könnyítési lehetőségek miatt reális nehézségbeli ugrást jelent a látens tér vizsgálata szempontjából az előző szintet képviselő Fashion-MNIST adathalmazhoz képest.

4. fejezet

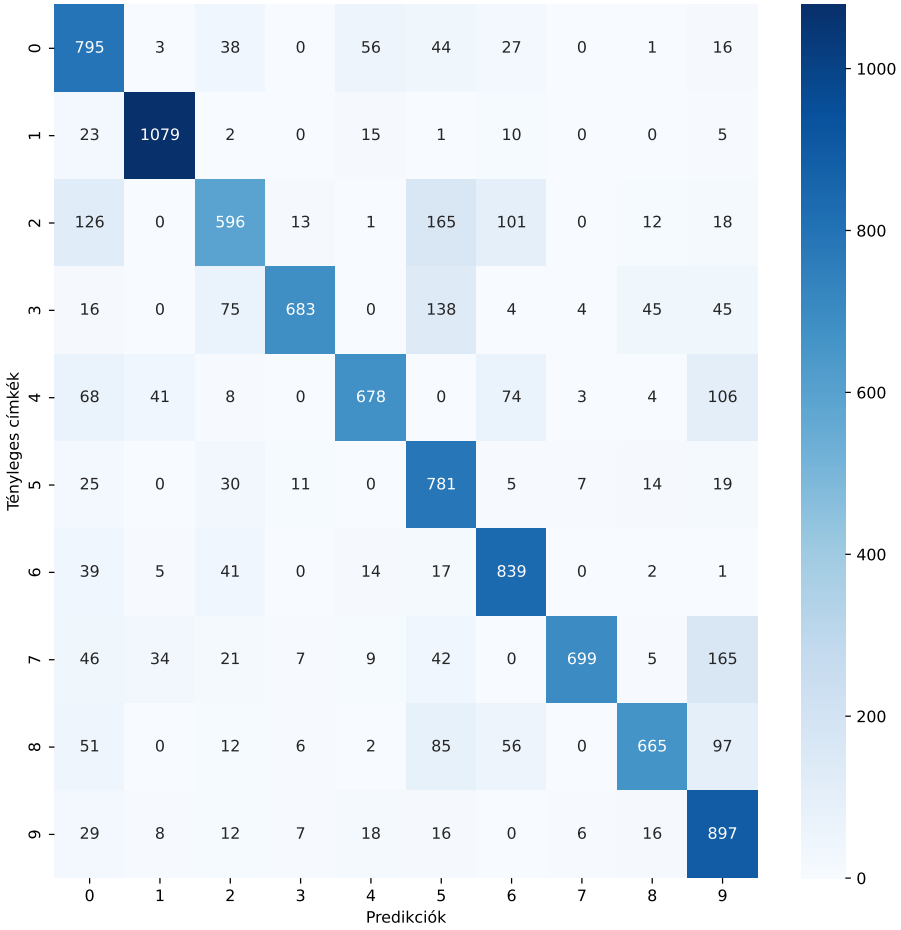
Az alkalmazott metrikák bemutatása

A fejezetben a különböző kísérleti összeállítások értékeléséhez használt metrikák bevezetése és ismertetése történik, így segítve a később kapott eredmények értelmezését. A mélytanulás és alapvetően a mesterséges intelligencia területén még mindig sokszor nehézkes az igazán jó metrika kiválasztása, amellyel egyszerre lehet kiértékelni és a külső felhasználó számára is jól reprezentálni az aktuális teljesítmény szintjét, azaz azt számszerűsíteni, hogy mi történik a hálózat „fekete dobozán" belül [13]. A dolgozatban igyekeztem két egyszerűen értelmezhető metrikát (pontosság és AUC érték) választani, amelyek jól szemléltetik az osztályozó hálózat teljesítményét. Ezen kívül ráadásul valamilyen szinten össze is függenek, így segítve a megértésüket. Alapvetően a felhasznált két metrika lényegüket tekintve egymásból származtathatók, így nem érdemes őket külön tárgyalni, hanem célszerűbb egymás után, összefüggően leírni a meghatározásukat.

4.1. Alapfogalmak a bináris osztályozásban

Először definiáljunk néhány alapvető fogalmat, amely gyakran előkerül az osztályozási problémák terén. A következőkben mindig bináris (tehát 2 kategória közül választhatunk) osztályozási feladat esetén közlöm az egyes fogalmakat, azonban ezeket könnyedén ki lehet terjeszteni több osztály esetére is: olyankor egyszerűen minden kategória esetén külön kell vizsgálni azokat a mintákat, amelyek odatartoznak, illetve külön azokat, amelyek nem. A 2 kategóriát tekinthetjük úgy, mintha egy adott elemről el kellene döntenünk, hogy igaz-e rá egy adott tulajdonság vagy sem. Ezek alapján definiálhatunk pozitív (P) és negatív mintákat (N) aszerint, hogy rendelkeznek-e a megadott tulajdonsággal (pl. az adott képen van-e kutya vagy nincs), majd elvégezhetjük valamilyen módszer szerint az adatok osztályozását és megkapjuk eredményként az algoritmus által helyesnek gondolt címkéket. Ezeket összehasonlítva a tényleges címkékkel kaphatunk számszerű értékelést arról, hogy hogyan teljesített a választott módszer, illetve összehasonlíthatjuk más módon kapott eredményekkel. Ehhez osszuk fel a teljes adathalmazt a predikciók és a tényleges címkék viszonya szerint 4 csoportra. Elsőként vegyük azokat a mintákat, ahol a kapott címke megegyezik az eredetivel, azaz helyes volt a módszer kimenete: itt két csoportot tudunk megkülönböztetni aszerint, hogy pozitív mintáról van szó (True Positive - TP), vagy negatívról (True Negative - TN). Ezekkel ellentétben vannak olyan adatok, ahol a módszerünk hibás kimenetet adott, azaz pozitívat, mikor negatívot kellett volna (False Positive - FP), illetve negatívot, amikor pozitívat vártunk el (False Negative - FN). Ezek alapján létre tudunk hozni egy táblázatot, amiben a sorok lehetnek a tényleges címkék, míg az oszlopok a predikált értékek és kitöltve a mezőket az azoknak megfelelő csoportok

darabszámát kapjuk. Ezt a táblázatot konfúziós mátrixnak nevezzük és gyakran használjuk több osztályos feladatok esetén is, amire a 4.1 ábra mutat példát az MNIST adathalmaz alapján.



4.1. ábra. Példa konfúziós mátrixra az MNIST adathalmazon

A modellünk teljesítményét az szemlélteti, hogy hány minta található az átlón kívül: ha tökéletes osztályozást érnénk el, akkor csak ebben az átlóban szerepelnének 0-nál nagyobb számértékek.

4.2. A pontosság és az AUC metrika

Miután fentebb tárgyaltuk az alapvető fogalmakat és kapcsolatukat az osztályozási feladattal, következhetnek a tényleges metrikák definíciói, amelyek közül a pontosság (*Accuracy*) egyszerűbb, míg az AUC érték összetettebb számítási móddal rendelkezik.

Elsőként vegyük a pontosság definícióját, amely az alábbi képlet szerint kapható meg:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{4.1}$$

ahol $\mathrm{TP}, \mathrm{FP}, \mathrm{TPN}$ és FN az előzőekben ismertetett csoportok mintaszámai, valamint ACC a számított pontosság értéke. Maga a képlet egyszerűen értelmezhető, hiszen vesszük a valódi pozitívak és negatívak mintaszámainak összegét és elosztjuk a teljes adathalmaz méretével, így egy 0 és 1 közé eső értéket kapva, amelyet értelmezhetünk százalékos formában is. Az így származtatott pontosság érték lényegében azt határozza meg, hogy a választott módszerünk milyen százalékban volt képes helyesen kategorizálni az egyes mintákat a megfelelő osztályokba. Alapszintnek a véletlen osztályozás tekinthető, amikor egyszerűen 50%-ban egyik, 50%-ban a másik osztályba soroljuk az adatokat, ami bináris osztályozás esetén 50%-os pontosságot képvisel, ha az egyes kategóriák eloszlása egyenletes. Ha ezt alulmúlja a módszerünk vagy esetleg minimálisan teljesít csak jobban, akkor szükséges annak módosítása, újragondolása a jobb érték eléréséhez, viszont fontos szem előtt tartani az adathalmaz sajátosságait is, ugyanis míg egyes adathalmazokon viszonylag könnyebb magas (95% fölötti) pontosságot elérni (pl. ahogy az MNIST esetében láttuk), addig sokkal összetettebb feladatok esetén már kisebb pontosság értéket is sokkal nehezebb elérni (pl. az ImageNet adathalmazon versenyzők között már a 90%-os eredmény is kimagasló [14]). Ezek alapján jól használható tehát a pontosság metrikája az osztályozó neurális hálózatok teljesítményének tömör, gyors értékelésére. A fentiek után következzen az ROC (Receiver Operating Characteristic) (bővebb leírásért lásd [15]) bemutatása, amely nevét a rádiós alkalmazásokból eredően kapta még a II. világháború idején, amikor is az ellenséges objektumok észlelése (pozitív, ha van, negatív, ha nincs) volt a feladat. Ebben a környezetben kifejezetten fontos volt a megfelelő észlelés, ugyanis bár a hamis pozitív eredmény sem optimális, mégis ehhez képest a hamis negatív következménye és ebből adódóan költsége rendkívül jelentős volt. Ebből következően alakították ki az ROC görbét, hiszen a minták asszimmetriájára való kisebb érzékenysége miatt annak elemzésével egy adott módszert alaposabban lehet jellemezni, mint az egyszerű pontosság értékkel. Bevezetéséhez azonban először szükségünk van a valós pozitív ráta (True Positive Rate, TPR), más néven Recall és a hamis pozitív ráta (False Positive Rate, FPR) vagy Fall-out meghatározásához, illetve az ezekkel párban lévő TNR (True Negative Rate vagy Specificity) és FNR (False Negative Rate vagy Miss Rate) fogalmakra, amelyek az alábbi képletekkel számíthatóak:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = 1 - \mathrm{FNR} \tag{4.2}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = 1 - \mathrm{TNR} \tag{4.3}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = 1 - \mathrm{FPR} \tag{4.4}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = 1 - \mathrm{TPR} \tag{4.5}$$

ahol a rövidítések az előzőekben megadott alapfogalmakat jelölik.

Ezek alapján az ROC teret, ábrát úgy vesszük fel, hogy a függőleges tengelyen a TPR, míg a vízszintes tengelyen az FPR értéke szerepel, így egy adott módszert értékei alapján egy pont határoz meg. Ezen kívül az ábrán kitüntetett szerepe van az átlós egyenesnek, amely a teljesen véletlenszerű módszer szintjét jelöli a valós kategóriák eloszlásának függvényében (ha egyenletes az eloszlás, akkor a $(0.5, 0.5)$ pont felel meg a véletlen

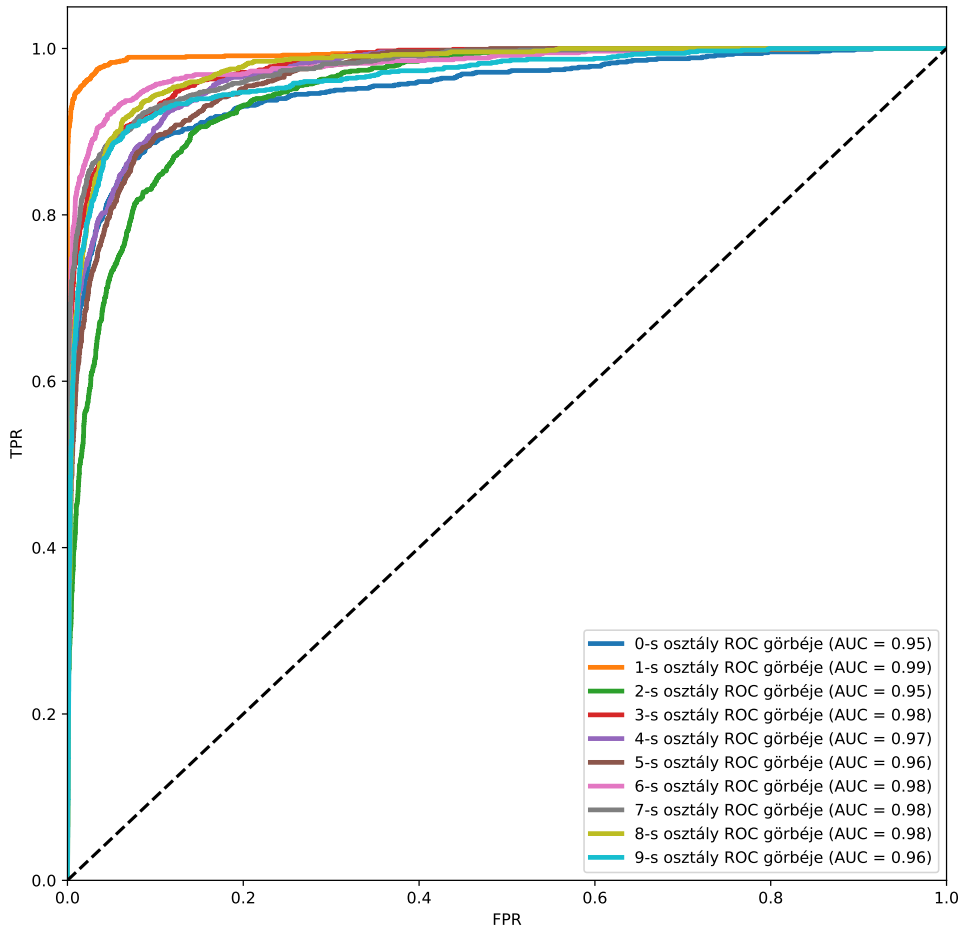
választásnak). Ha ezen egyenes feletti pontot kaptunk akkor jobb a módszerünk és annál pontosabb, minél inkább közelít a tökéletes pontosságot jelentő $(0, 1)$ ponthoz (ilyenkor TPR $= 1$ és FPR $= 0$, azaz TNR $= 1$ és FNR $= 0$). Ha a diagonális egyenes alatt teljesítünk, akkor a módszer rosszabb a véletlen jóslásnál, azonban invertálva a kimenetét könnyedén tükrözhetjük a jó oldalra. Ezek alapján tehát a kapott pont helyzete alapján tudjuk jellemezni a teljesítményt, sőt több módszert is összevethetünk egy ábra alapján.

A fentieken túl lehetőség van még görbék megjelenítésére is az ROC ábrán, amikor is minden pont egy-egy helyzetet, állapotot jelent. Ehhez vegyünk egy X folytonos eloszlású valószínűségi változót, amelyhez egy T küszöbértéket rendelve eldönthetjük. hogy pozitívnak ($X > T$) vagy negatívnak ($X < T$) tekintjük-e az értéke alapján. Emiatt, ha a pozitív X-ek sűrűségfüggvénye $f_1(x)$ és a negatívaké pedig $f_0(x)$, akkor a TPR és az FPR értékek a következően definiálhatóak a T küszöb függvényében:

$$\mathrm{TPR}(T) = \int_T^\infty f_1(x)\,dx \tag{4.6}$$

$$\mathrm{FPR}(T) = \int_T^\infty f_0(x)\,dx \tag{4.7}$$

amelyek alapján az ROC görbe paraméteresen ábrázolja a TPR(T) és FPR(T) közötti viszonyt a T küszöbparaméter függvényében. Az előzőekhez hasonlóan itt is az $(0, 1)$ pont elérése lenne a cél, azonban itt nyomon tudjuk követni a görbe alakulásával a módszer hatékonyságát: megvizsgálhatjuk, hogy milyen gyorsan ér el az 100%-os TPR értékhez és közben mekkorára nő a FPR értéke. Erre példát mutat a 4.2 ábra, ahol a konfúziós mátrix esetén már előzőleg bemutatott MNIST adathalmazon feltanított hálózat osztályonkénti ROC görbéi láthatóak.



4.2. ábra. Példa ROC görbékre az MNIST adathalmazon

Az ábrán látható, hogy a példa modell különböző mértékben teljesített jól az egyes osztályok esetén, azonban mindig jóval a véletlen módszert jelentő diagonális egyenes (az ábrán szaggatott vonallal jelölt) fölött sikerült maradnia. Az ábrán ezeken kívül még megjelenik az AUC (*Area Under Curve*) érték is az egyes kategóriák esetén, ami egyszerűen a görbe alatti területet adja meg. Ez azért hasznos számunkra, mert így gyorsan és egyszerűen tudunk összehasonlítani több modellt akár egy számérték alapján (ilyenkor az osztályok mintaszáma szerint súlyozottan átlagoljuk az egyes értékeket), ami mégis összetettebb, mint a pontosság, azonban mégsem igényli az egyes görbék elemzését. Ezek alapján, a különböző kísérleti összeállítások vizsgálatakor és a nagyszámú modelltanítás miatt, az AUC számérték kiszámítása kifejezetten előnyös számunkra, a pontosság értékének vizsgálata mellett.

5. fejezet

Az annotálandó minták kiválasztásának folyamata

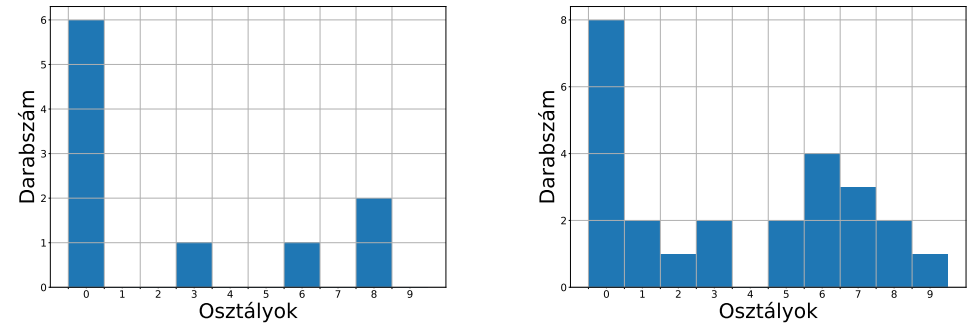
Ebben a fejezetben a munkám új eredményei, a mintakiválasztási folyamatok kerülnek ismertetésre, külön kitérve a látens térrel való kapcsolatukra, illetve az esetleges előnyeikre és problémáikra. Az első bemutatott megközelítés eltér a többitől annyibban, hogy nem kapcsolódik a látens térhez és nem célja az annotálandó minták kiválasztásának hatékony megvalósítása, egyszerűen csak viszonyítási alapként szolgál a későbbiekhez. Ezzel szemben a másik kettő tárgyalt algoritmus igyekszik a látens tér egy-egy előnyös tulajdonságát kihasználni és az alapján egy hatékony módszert definiálni adott számú annotálandó minta kiválasztásához. Itt megemlítem, hogy a dolgozat során a látens teret érintő műveleteket a t-SNE által adott ponthalmazokon végzem el a szemléletes ábrázolás és összzehasonlíthatóság miatt. Természetesen a látens téren elvégezve a kapott eredmények módosulnak, mivel a t-SNE is belevisz egy hibalehetőséget a folyamatba.

5.1. Minták kiválasztása a naiv módszer alapján

Elsőként tekintsük az alapszintet jelképező legegyszerűbb módszert: a naiv mintakiválasztás folyamatát. A módszer segítségével megismerhetjük az egyes adathalmazok tulajdonságait, illetve összehasonlíthatjuk a később bemutatott, összetettebb algoritmusok teljesítményét a használatával kapott eredményekkel.

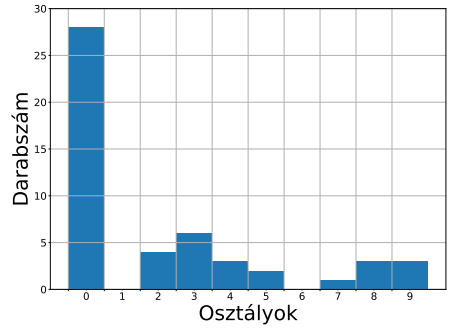
A módszer maga az egész adathalmazból dolgozik, választ ki elemeket, mégpedig teljesen véletlenszerű módon. Ez a gyakorlatban azt jelenti, hogy egy adott méretű részadathalmaz létrehozásához véletlenszerűen kiválasztunk megfelelő számú mintát, ügyelve arra, hogy egy adatminta csak egyszer szerepelhessen. Ezek alapján jogosnak mondható a „naiv" elnevezés, hiszen a legtöbb esetben az osztályozási feladatok témakörében a legalapszintűbb megközelítés a véletlen választás: gondoljunk csak például az ROC ábrán látott diagonális egyenesre, amely alapján értékelhettük a saját módszerünk teljesítményét a véletlenszerűhöz képest.

Ugyan maga a módszer működése egyszerű, könnyen érthető, mégis érdemes behatóbban elemezni a tulajdonságait, azon belül is a hátrányait, hiszen ezáltal képet kapunk azokról a problémás helyzetekről, amelyeket az összetettebb módszerekkel meg szeretnénk majd oldani. Ezek közül egyik legfontosabb ilyen a kapott részadathalmazok osztály szerinti eloszlásának kérdése, amelyet a 5.1 ábracsoport szemléltet különböző méretű halmazok esetében az MNIST adathalmazon.

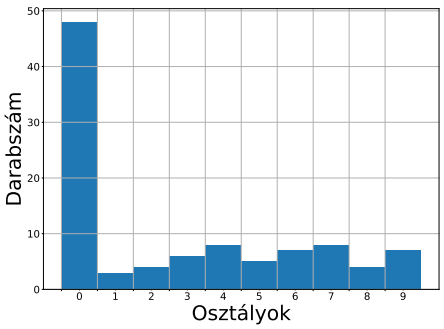


(a) $N = 10$

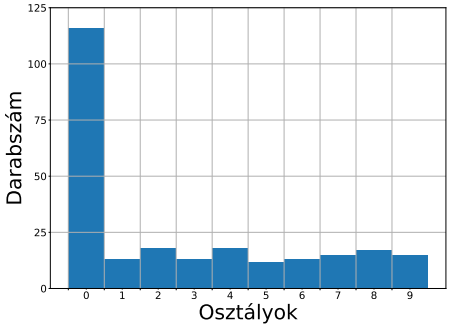
(b) $N = 25$



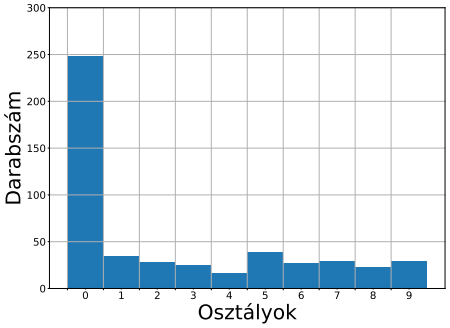
(c) $N = 50$



(d) $N = 100$



(e) $N = 250$



(f) $N = 500$

5.1. ábra. N méretű, naiv módszerrel generált részadathalmazok eloszlása kiindulva az MNIST adathalmazból

Ezeken az ábrákon az látható, hogy a kisszámú adathalmazok esetén az egyes kategóriák eloszlása túlságosan egyenlőtlen: a 0-s számjegy egyszerűen dominálja a halmazt. Ez természetesen a teljes adathalmaz szerkezetéből ered, hiszen ott szándékosan az egyik osztályból (0-ás számjegy) választottuk a teljes méret 50%-át (ez szemléletesen már a 5.1f ábrán is látható). Így, ha elkezdünk véletlenszerűen válogatni a minták közül, akkor a kisebb halmazok esetében könnyen előfordulhat a darabszámbeli különbségek miatt, hogy egy-egy osztály egyszerűen kimarad és helyettük egy 0-s minta kerül kiválasztásra (lásd a 5.1a, 5.1b és 5.1c ábrákat, amelyeken legalább 1 osztály „hiányzik”). Ez a jelenség magával vonja azt a problémát, hogy az így kapott adathalmazon tanított neurális hálózat nem fog bizonyos osztályokat megismeri, jellemzőiket megtanulni, tehát kiértékeléskor nem fog tudni mit kezdeni ezekkel a kategóriába tartozó mintákkal, emiatt sokkal rosszabb eredményt produkálva, mint egy olyan modell, amely minden osztályból egyenlően tanult.

A fentiek alapján tehát az elsődleges célunk a saját módszerek definiálásánál, megválasztásánál az lenne, hogy a kisebb méretű adathalmazok létrehozásánál minél jobban törekedjünk a kiegyensúlyozott osztályeloszlásra az eredeti kategória eloszlás korlátai között.

Másik megfontolandó kérdés a címkézésen kívül belefektetett munka mennyisége, hiszen, ha a kiválasztás legalább annyi erőforrást igényel (ide az algoritmus futási ideje is beleérthető), akkor elbuktuk azt a célt, hogy javítsunk az eredeti helyzeten. Ilyen szempontból a véletlen választáson alapuló módszer ugyancsak egy jó alapszintet képvisel, hiszen a választás teljesen automatikus, a kívánt mintaszámtól függően ugyan változó futási idejű, azonban nem tekinthető lassúnak (a mérettel egyenes arányban függ a futási idő). Fontos tehát, hogy lehetőleg minimalizáljuk a javasolt módszerek futási idejét és az eredményhez szükséges emberi beavatkozás mértékét is.

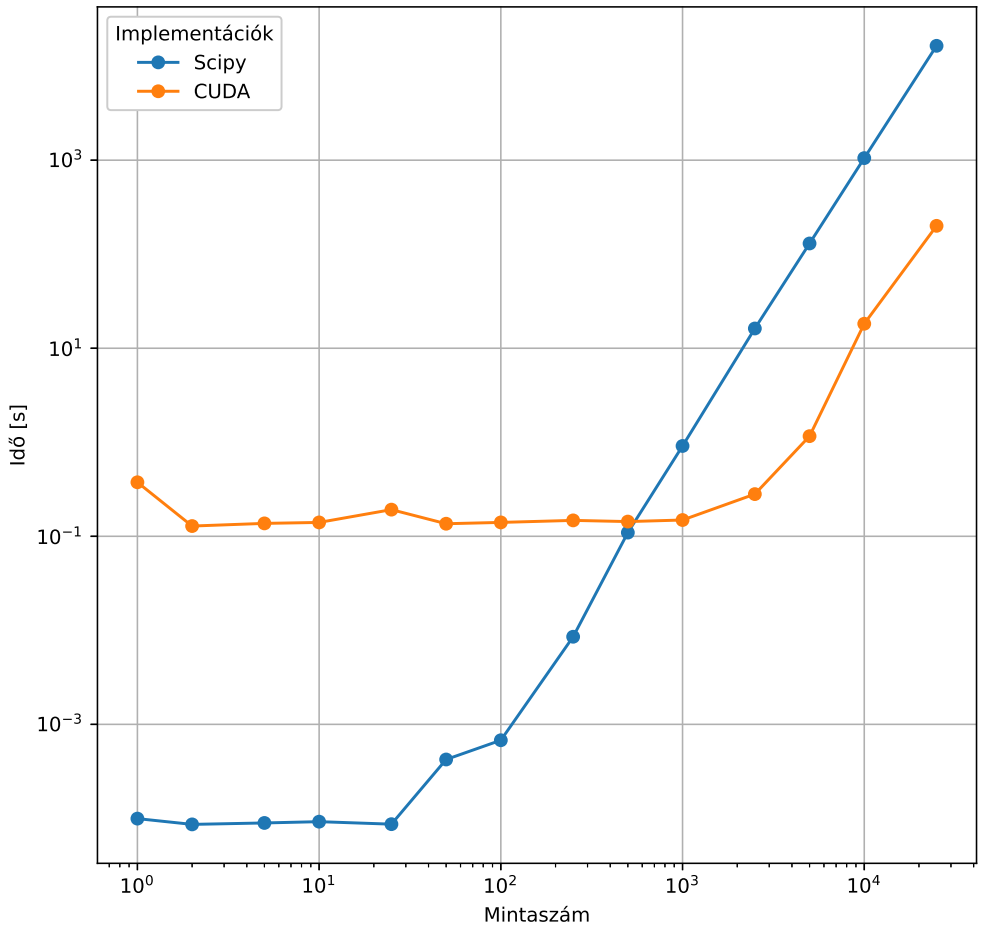
5.2. Minták kiválasztása a szomszédsági szám alapján

Miután tárgyaltuk az alapszintet képező naiv módszert, következhet a minták kiválasztásának egy összetettebb folyamata, amelyet először egy, a szomszédszámon alapuló módszer segítségével fogunk elvégezni. A módszer eredményeként az eredeti adathalmazt adott mértékig szűrjük meg, hogy az osztályozó hálózatot hatékonyabban tudjuk kevesebb mintán feltanítani. A következő részfejezetekben a módszer egyes lépései kerülnek bemutatásra, majd a kapott, csökkentett méretű adathalmazzal történő tanítás kerül összehasonlításra az eredeti, teljessel való tanításhoz képest.

5.2.1. Távolsági mátrix számítása

Első lépésként a távolsági mátrix kiszámítása történik, amely alapján majd a szomszédsági viszonyokat tudjuk megállapítani. Magának a mátrix kiszámításához csak a látens térbeli vektorokra van szükségünk, amelyeket például az előzőleg feltanított autonekóder hálózat enkóder moduljának segítségével tudunk kinyerni.

A távolsági mátrix egy táblázat, amely tartalmazza az egyes adatpontoknak megfelelő látens kódvektorok távolságát (koordinátánként vett négyzetes távolság, norma) az összes többi minta vektorától. Alapvetően a mátrix kiszámítása magától értetődő, azonban nagyobb mintaszámok esetén a szükséges műveletvégzések száma négyzetesen nő: az itt közölt példában az MNIST adathalmaz csak 10000 mintáját használtam fel, mégis a teljes mátrix mérete 10000×10000, amely összesen 10^8 távolságszámítás elvégzését igényli. Látható tehát, hogy célszerű lenne ezt a szükséges lépést minél hatékonyabban és lehetőleg gyorsabban elvégezni, hogy nagyobb adathalmazok esetében sem okozzon problémát a teljes mátrix kiszámítása: erre megoldást jelent az egyes műveletek egyidejű futtatása (megtehető, hiszen egymástól függetlenek), amelyet a GPU-t használva könnyen elérhetünk. Ennek megvalósítására használtam fel a GPU közvetlen programozásának lehetőségét a *CUDA* programcsomag segítségével (erre Python nyelven lehetőség van a Numba csomag segítségével [16]), hiszen ennek köszönhetően lehetőség nyílik kihasználni a videokártya által nyújtott párhuzamosítás lehetőségét, ezzel nagyban felgyorsítva a számításhoz szükséges időtartamot.



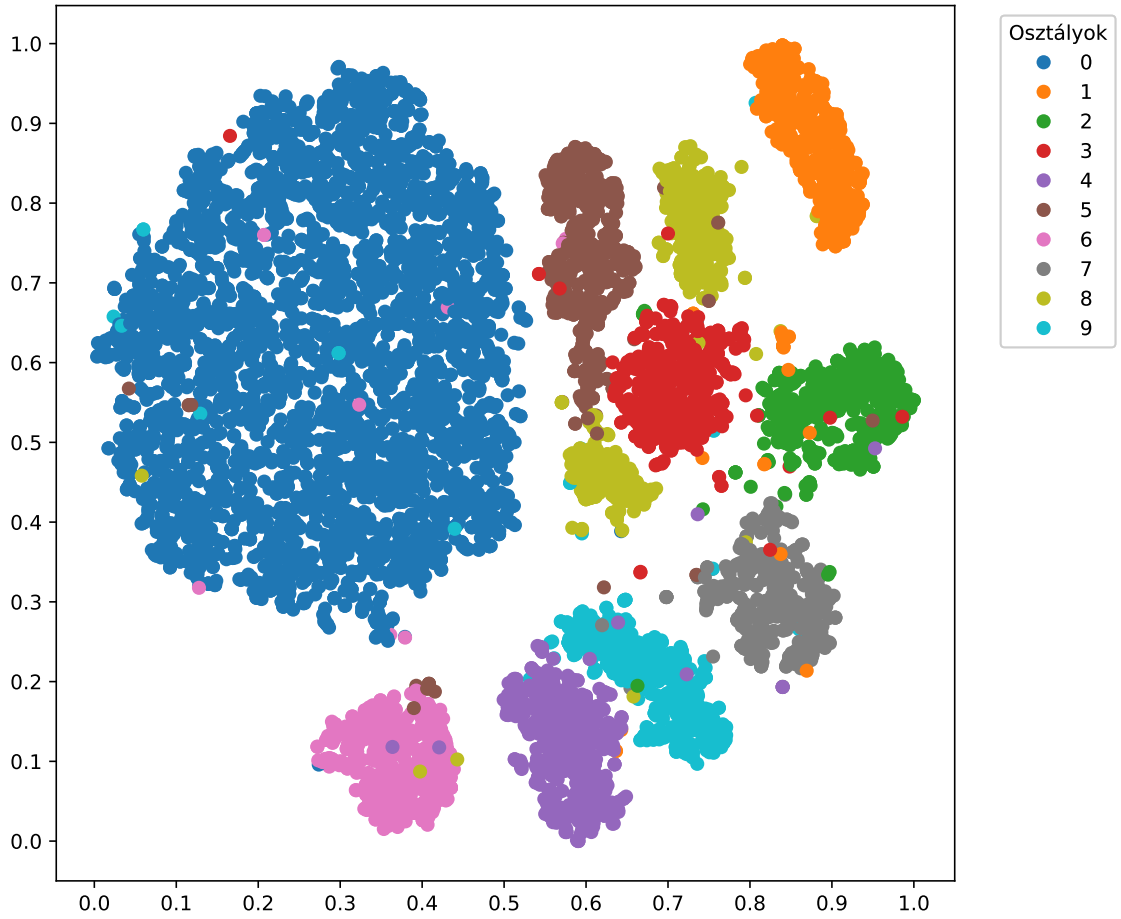
5.2. ábra. A CPU-n (Scipy programcsomagot használva) és a GPU-n futó távolságmátrix implementációk futási idejének összehasonlítása különböző mintaszámok esetén

A gyorsítás szemléletesen a 5.2 ábrán látható, ahol különböző mintaszámok esetében hasonlítottam össze a futási időt a hagyományos (CPU alapú) és a CUDA segítségével megvalósított implementáció között. A kapott eredmények alapján valóban kijelenthető, hogy a párhuzamosítás okozta gyorsítás ugyan igazán csak $10,000$ minta felett érvényesül (előtte a hátrány a járulékos CPU és GPU közötti oda és vissza történő adatmásolás és kommunikáció miatt van), de akkor gyorsaság szempontjából kedvezőbb értékeket eredményez a hagyományos módszerhez képest, hiszen a mintaszám növekedésével nem kezd el azonnal nagy ütemben nőni. Itt érdemes megjegyezni, hogy már a t-SNE algoritmusnak is létezik GPU-ra optimalizált verziója [17], amely ráadásul még több paraméterezhetőséget és szabadságot visz bele a látens tér szerkezetének elkészítésébe, azonban a dolgozat során, mivel kisebb mintaszámokkal (legfeljebb $10,000$) dolgozunk, így nem szükséges az alkalmazása.

5.2.2. Szomszédsági szám meghatározása és hatásai

Miután megkaptuk a teljes távolsági mátrixot, következik a szomszédsági számok meghatározása, amelyek alapján képesek leszünk felmérni a látens térbeli pontok helyzetét és elhelyezkedését egymáshoz képest.

Ehhez először vizsgáljuk meg példaként a látens tér szerkezetét a 5.3 t-SNE ábra segítségével.



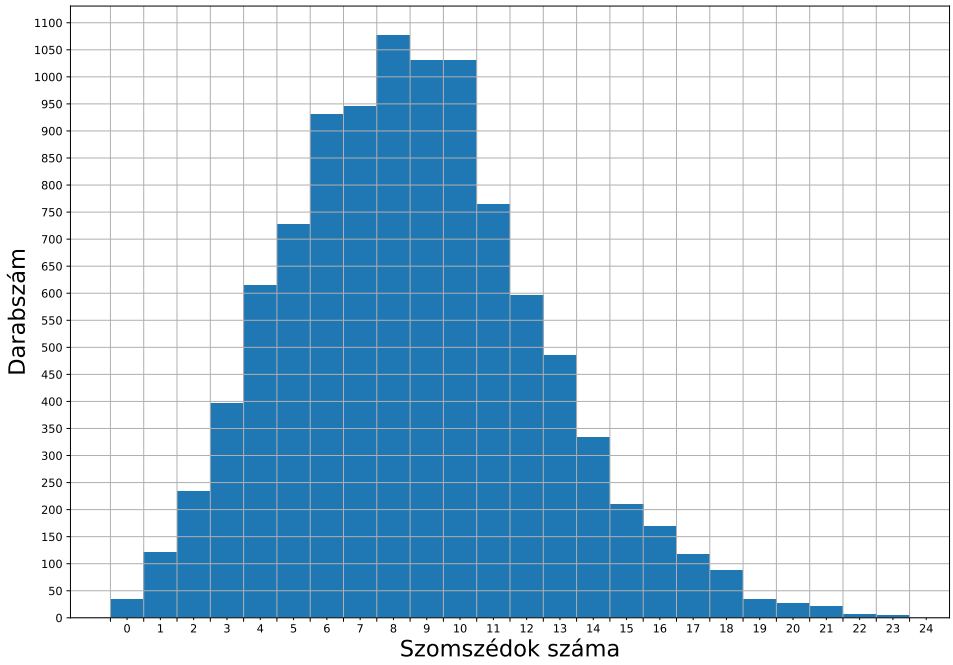
5.3. ábra. A feltanított autoenkóder látens tere az MNIST 10000 mintáján

Az ábrán láthatóak a számjegyeknek megfelelő klaszterek, amelyek ugyan viszonylag jól körülhatárolhatóak, több esetben is összeérnek egymással, így nehezítve a pontos elkülönítést. Másik jelenség, ami észrevehető az ábrán, hogy ugyan az egyes osztálycsoportok középpontja „egyszínű”, azaz csak az adott klaszterbe tartozó mintát tartalmaz, addig az egyes csoportok szélei, határai már sokkal inkább vegyes képet mutatnak. Ezt szeretnénk a szomszédságon alapuló módszerrel kiszűrni, illetve az esetleges leszakadó, kívülálló (*outlier*) pontokat eltüntetni.

A fenti megállapítások után következhet a szomszédság definiálása, meghatározása: legyen két adatpont szomszédos a látens térben, ha az egymástól vett távolságuk egy adott küszöbértéken belül van. Alapvetően a meghatározás magától értetődő hiszen azt szeretnénk, hogy az egyes osztályklasztereken belül és a széleiken lehetőleg csak az adott kategóriába tartozó minták legyenek, így kiszűrve az esetleges bevegyülő más osztályokat. Magának a küszöbérték kiválasztása azonban ennél bonyolultabb, hiszen ez fogja befolyásolni a kapott mintaszámot, a klaszterek méretét és alapvetően a módszer sikerességét,

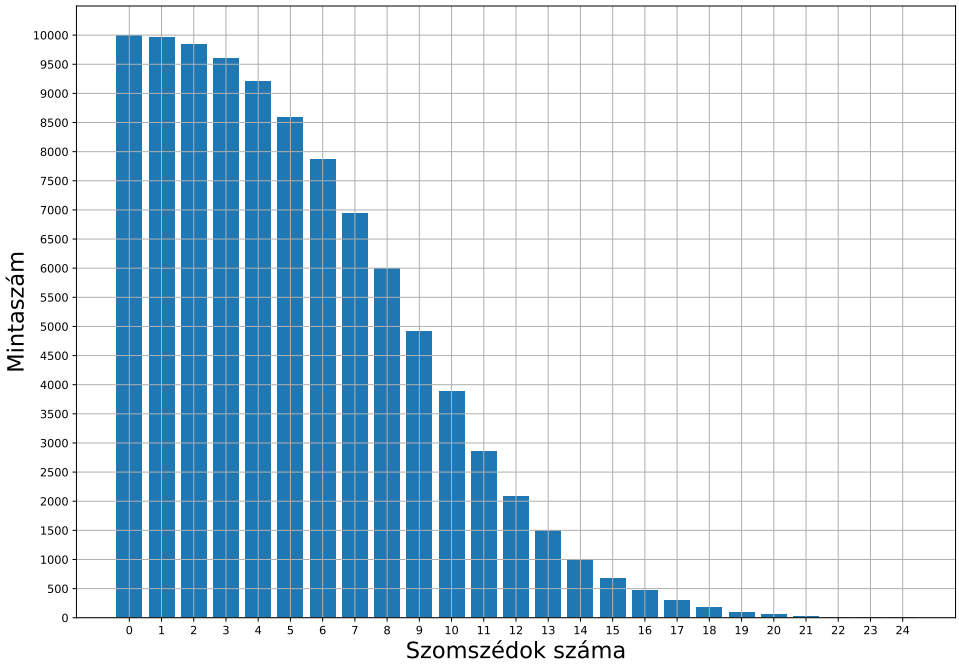
célszerű tehát egy megfelelő értéket választani. Ehhez tekinsük a 5.3 ábrát, ahol a normált (0 és 1 közé skálázott) távolságok alapján könnyebben tudjuk megválasztani a szükséges küszöbértéket. Fontos, hogyha túlságosan kicsi értéket választunk, akkor csak a klaszterek gócpontjait tudjuk igazán megtalálni, míg, ha túlságosan nagyra választjuk a küszöböt, akkor szinte egy adatpontot sem fogunk tudni kiszűrni, tehát a két véglet között kell egy megfelelő értéket találni, amely így minden feladat és adathalmaz esetén más és más lesz. Jelen esetben az ábrán látható klaszterek sűrűsége és egymáshoz való közelsége miatt 0.01-et választottam, hiszen így az egybefolyó csoporthatárokat szét tudjuk választani úgy, hogy közben a klaszterek középpontja épen marad.

A fentiek alapján elvégezve a szomszédságok keresését a t-SNE reprezentáció alapján kapott távolsági mátrix elemein 0.01-es küszöbértéket használva a 5.4 ábrán látható szomszédsági szám eloszlást kaptam.



5.4. ábra. A távolsági mátrix alapján kiszámított szomszédsági számok eloszlása

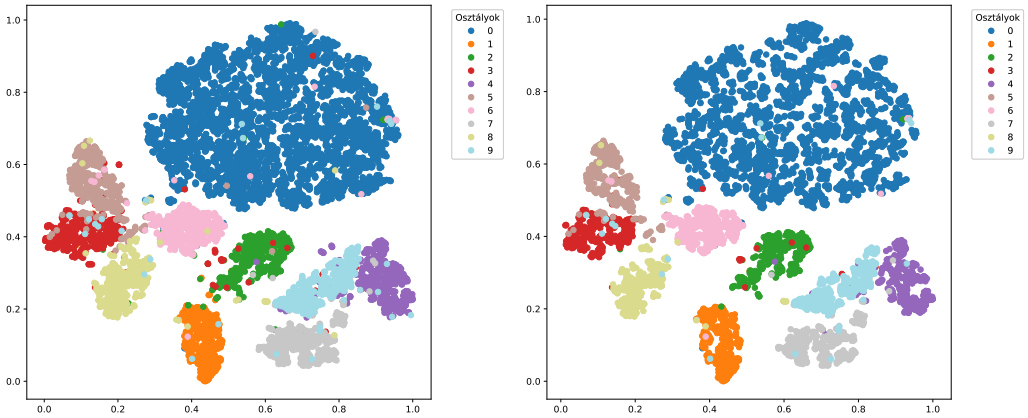
Az ábrán látható, hogy viszonylag kevés szomszéd nélküli vagy csak pár közeli ponttal rendelkező minta van a látens térben, ami az *outlier* pontokat és a több osztályból álló kisebb halmazokat jelentheti. A szomszédsági szám növekedésével egy pontig a darabszám is növekedik, majd a 8-es érték után elkezd először lassabban, majd drasztikusan csökkeni, egészen 24-ig, amely a legnagyobb szomszédsági érték az adathalmaz esetén. Ezek alapján az látható, hogy arányaiban kevés egyedülálló minta van a látens térben, míg az adatpontok nagyrésze a klaszterek részét képezi, azon belül is sok a csoportok belső, sűrűbb területein helyezkedik el, innen eredően nagyobb szomszédsági számmal is rendelkeznek. A látens tér szomszédsági szerkezetének elemzését segíti még a 5.5 ábra is, ahol az egyes szomszédsági számértékeknél nagyobb értékkel rendelkező minták számának alakulása látható.



5.5. ábra. Az adott szomszédsági számok jelentette küszöbök feletti minták számának alakulása

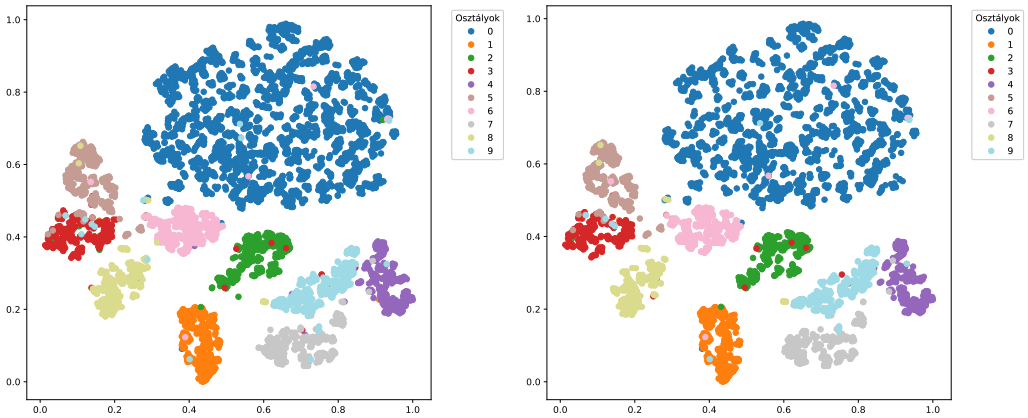
A $10,000$ mintáról elindulva először a szomszédsági küszöb értékét növelve a megmaradó minták száma lassan kezd el csökkeni, majd a 5.4 ábrán látott 8-es értéket közelítve egyre rohamosabb mértékben kezd el fogyni a mintaszám. Az előző ábránál tett megállapításokat kiegészítve csekély számú *outlier* minta (kb. 500 legfeljebb 3 szomszédos minta) van, míg a mintaszám jelentős része (több, mint $50\% - a$) 8 vagy annál több szomszéddal rendelkezik, tehát egy osztályklaszter belsejében helyezkedik el.

A fentiek gyakorlatias szemléltetésére szolgálnak a 5.6 ábrák, ahol az egyes szomszédsági küszöbértékeknél több szomszéddal rendelkező minták t-SNE ábrái láthatóak.



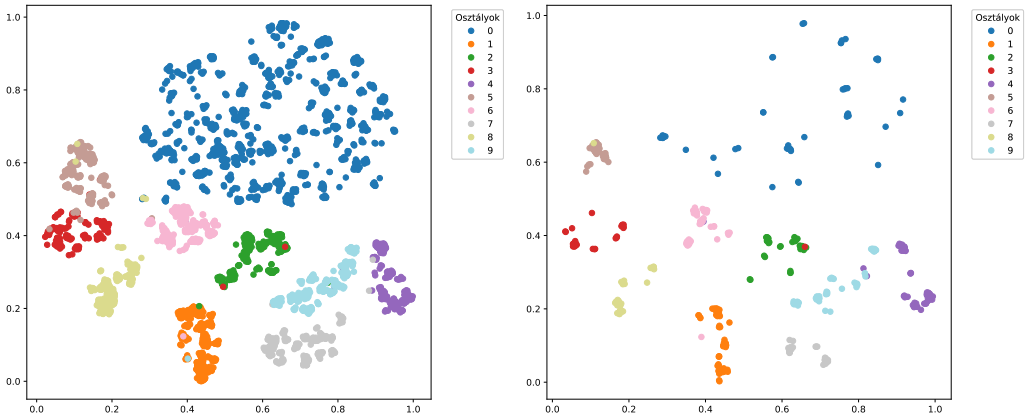
(a) $N = 3$

(b) $N = 6$



(c) $N = 7$

(d) $N = 8$



(e) $N = 10$

(f) $N = 15$

5.6. ábra. A legalább N szomszéddal rendelkező minták ábrázolása

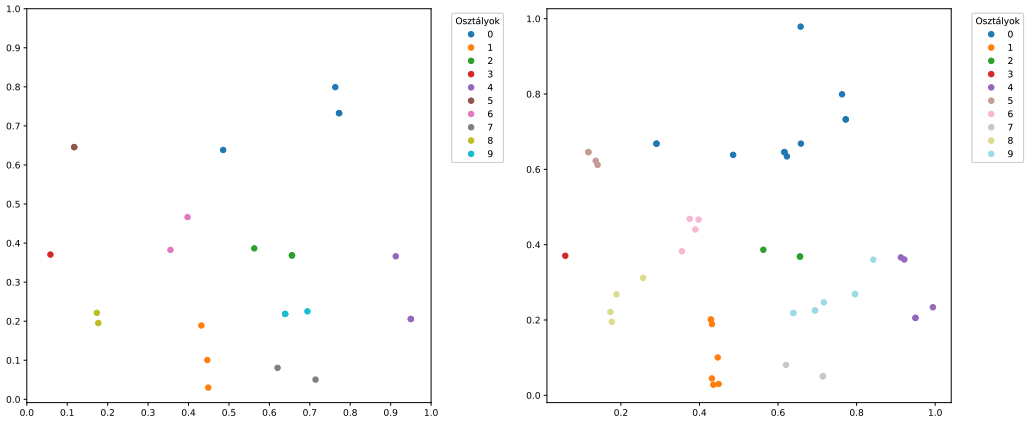
Ezek alapján megfigyelhető a szomszédsági küszöb értékének növelésével hogyan tűnnek el egyre nagyobb mértékben a t-SNE ábrákon látható pontok. A 5.6a ábrát összehasonlítva az eredetivel (lásd 5.3 ábra) látható, hogy egyes nem odaillő pontok eltűntek a klaszterek mellől, azonban túlságosan nagy változás nem tapasztalható, hiszen csak kb. 400 mintát szűrtünk ki. Élesebb váltás látható azonban a 5.6b, ahol már közel 2100 mintát távolítottunk el és így az osztályklaszterek szélei már sokkal markánsabbak, kevésbé lógnak össze egymással, illetve a magányos ponthalmazok nagy része is eltűnt a szomszédsági szám növelésével. Tovább növelve a küszöbértéket egészen a 7-es értéknél lévő határig,

ahol kb. 3000 mintát szűrtünk ki, azt tapasztaljuk, hogy a klasztereink teljesen szétváltak, illetve elkezdtek kisebb szigetekre szakadni. Ez a folyamat figyelhető meg a 5.6d és a 5.6e ábrákon, majd a 15-ös értékig eljutva (5.6f ábra) a klasztereink teljesen eltűntek, egyedül a középpontjuknak megfelelő ponthalmazok maradtak meg, ami összesen kb. 700 mintát jelent. Maga a fenti ábrázolás módszer használható a megfelelő küszöbérték szemléletes beállításához is, hiszen adaptívan láthatjuk, hogyan változnak a klaszterek szerkezetei az egyes szomszédsági számok fölött.

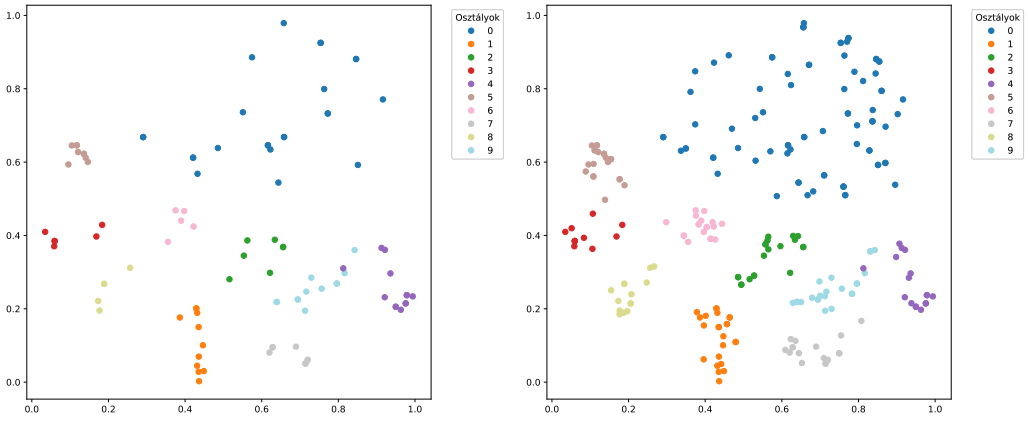
5.2.3. A minták kiválasztásának folyamata

A fentiek alapján képet kaphattunk a látens térben elhelyezkedő minták szomszédsági viszonyainak működéséről és az abból eredő következményekről, azonban csak maga a szomszédsági számérték nem elegendő, szükség van egy folyamatra, amellyel a számunkra fontos mintákat kiválogathatjuk és felcímkézhetjük. Elsőre kézenfekvőnek tűnhet egyszerűen a szomszédsági szám alapján szelektálni: nagyságrendileg csökkenő sorba rendezhetjük a mintákat a szomszédsági értékük alapján, majd mindig az adott adathalmaz méretének megfelelő mennyiséget veszünk el az elejéről. Ezzel azonban az a probléma merülhet fel, hogy a kisebb halmazméretek esetében csak a legnagyobb szomszédsági számmal rendelkező adatpontokat fogjuk beválasztani a halmazunkba, amely könnyen azt eredményezheti, hogy csak egy kategóriából fogunk mintákat látni, hiszen közel voltak egymáshoz a klaszter középpontjában. Ennek kiküszöbölésére célszerű tehát módosítani a kiválasztáson, hogy minél kiegyensúlyozottabb adathalmazokat kapjunk.

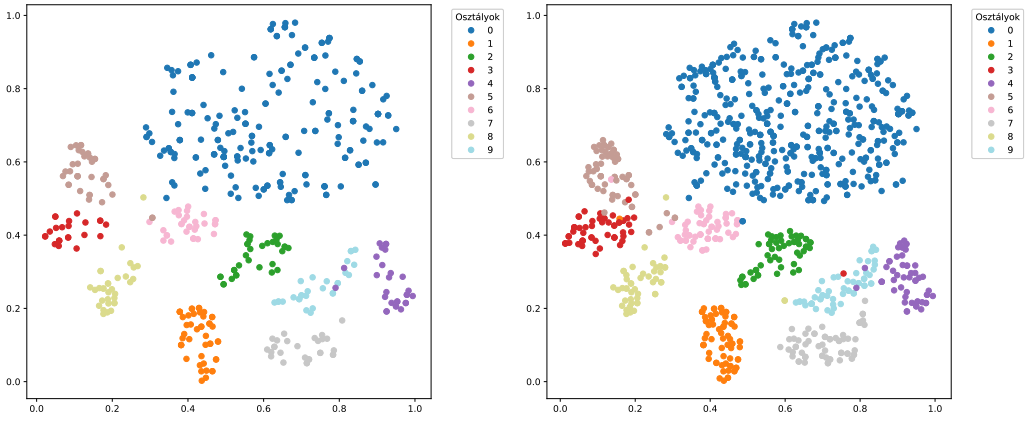
Végezzük el tehát először a minták nagyság szerinti csökkenő sorba rendezését a szomszédsági szám alapján, majd válasszuk ki az elsőt közülük. Ezután keressük meg azokat a mintákat, amelyek a kiválasztottal szomszédosak (azaz egy adott sugarú környezetében vannak), majd azok szomszédsági számát nullázzuk ki. Végezetül pedig rendezzük sorba újból -mostmár a változtatott szomszédságokkal- a mintasort és lépjünk a második mintára. Ezeket a lépéseket ismételjük addig, ameddig a megadott számú mintát sikerült kiválasztanunk a felcímkézendő adathalmazunkba. Ez a kiegészítés a kiválasztás menetében azt eredményezi, hogy az algoritmus egymáshoz közeli adatpontokat kisebb valószínűséggel választ egymás után az adathalmazunkba. Ennek a folyamatnak az eredménye látható a 5.7 ábrákon, ahol az így létrehozott különböző méretű adathalmazok mintáinak a t-SNE ábrán elfoglalt helyét lehet nyomon követni.



(a) $N = 25$ **(b)** $N = 50$



(c) $N = 100$ **(d)** $N = 250$



(e) $N = 500$ **(f)** $N = 1000$

5.7. ábra. Az algoritmus szerint létrehozott N méretű adathalmazok t-SNE ábrái

Ezek alapján az látható, hogy már a kisebb mintaszámok esetében is (lásd 5.7a és 5.7b ábrák) viszonylag kiegyenlített az adathalmaz eloszlása, majd a méret növekedésével az egyes kategóriák viszonya nem változik. Ez kifejezetten az 0-ás számjegy esetében fedezhető fel, amelyből eredetileg ugyan 5000 darab van (míg a többi számjegyből csak 500-500), mégis egészen a 5.7d-es és 5.7e-es ábrákig nem dominálja a kapott adathalmazt.

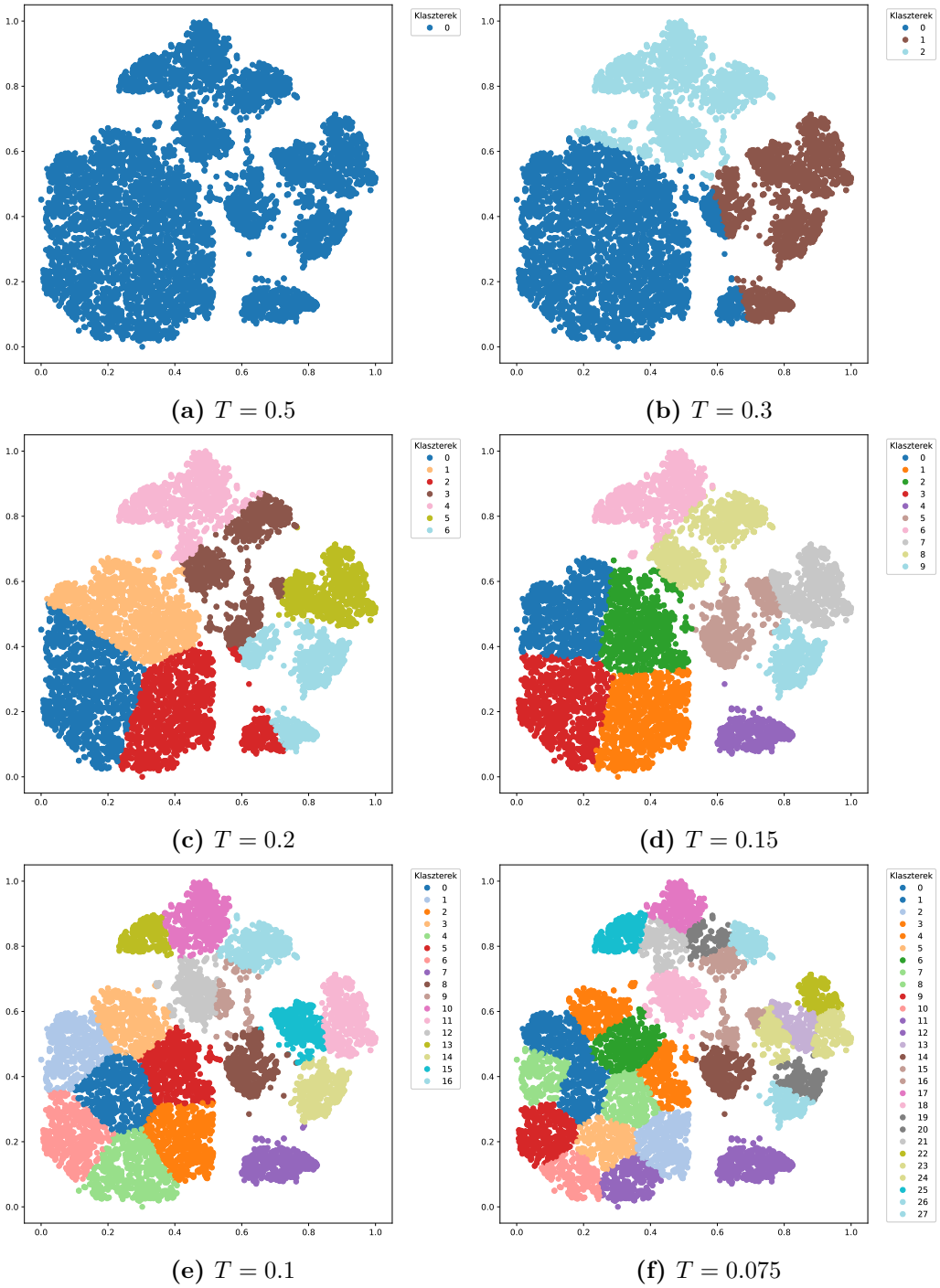
5.3. Minták kiválasztása klaszterezés alapján

Az utolsóként tárgyalt módszer a részfejezet címében is szereplő klaszterezési folyamaton alapuló megközelítés: egy klaszterező algoritmus kimenete alapján, kisebb felhasználói beavatkozások segítségével próbál meg minél megfelelőbb mintaeloszlású és felépítésű részadathalmazokat generálni. A módszer bemutatásához először magának a kiválasztott klaszterezési eljárásnak a részletezése történik, majd ezután a kapott kimenet alapján elvégzendő mintakiválasztás menetének ismertetése következik.

5.3.1. A klaszterezési algoritmus, a BIRCH bemutatása

Elsőként tekintsük a mintaválasztó módszer alapját képező klaszterezési eljárást: a BIRCH algoritmust [18]. Ennél a lépésnél nagyon sokféle választási lehetőségünk van a klaszterező módszerek tekintetében, azonban célszerű egy olyat használni, amely számára megadható a kívánt klaszterszám, így direkt ráhatásunk van a működésre, amely a későbbi lépések során lesz fontos. Ezen felül a BIRCH alkalmas nagyszámú minta csoportosítására is, ami elengedhetetlen egy ipari adathalmaz esetén, amely akár milliós méretű is lehet. Természetesen jelen esetben választhattunk volna ennél sokkal egyszerűbb, akár kézi módszert is a csoportok kialakítására a kisebb adathalmazokon, azonban fontos szem előtt tartani az említett skálázhatóságot, illetve az algoritmus jelentette automatizálást.

Működését tekintve a BIRCH eljárás egy klaszterezési tulajdonság fát (*Clustering Feature Tree*, CFT) épít fel a rendelkezésekre álló adatokból úgy, hogy összetömöríti az egyes adatpontokat egy-egy alklaszterbe, majd ezeket fába szervezi: egyes alcsoportok csomópontokhoz (*Clustering Feature Nodes*, CF Nodes) lesznek rendelve és ugyancsak indulhatnak ki belőlük csomópontok egészen addig, amíg a levelekig el nem jutunk. Az így kapott alklaszter fát egy globális csoportosítónak továbbadva megadható, hogy pontosan hány klasztert szeretnénk kapni és az algoritmus ez alapján vonja össze a fa egyes csomópontjainak alklasztereit, illetve az azokhoz tartozó mintákat. Ha viszont nem adunk meg elvárt csoportszámot, akkor egyszerűen a fa levél alklasztereit fogjuk egy az egyben visszakapni eredményül. Maga az algoritmus első lépése a következő: egy új mintát véve azt beillesztjük a CF fa gyökeréhez (ez eleve egy csomópont), majd abba a gyökérhez tartozó alklaszterhez rendeljük, amelynek így a kapott sugara (euklideszi értelemben) a legkisebb lesz, tehát az önmagához képest legközelebbi csoportba soroljuk. Ha az így választott alcsoport rendelkezik csomóponttal, akkor a lépést addig kell ismételni, amíg egy levélalklaszterig nem jutunk és azt megtalálva rekurzívan frissítjük az alcsoportot és annak szülő alklasztereit a hozzáadott mintaponttal. A fenti működést kiegészíti a klaszterválasztásnál megjelenő két paraméter: a küszöbérték és az elágazási tényező. Ugyanis, ha az új minta és a hozzá legközelebbi alklaszter euklideszi távolsága nagyobb a küszöbérték négyzeténél és már elértük a maximális alklaszter számot, melyet az elágazási tényező határoz meg, akkor létrehozunk egy ideiglenes helyet a számára. Ezután vesszük a két legtávolabbi alcsoportot és kettéosztjuk a többit az alapján, hogy melyikhez vannak közelebb, így két csoportot létrehozva. Ha a kimaradt pontunk rendelkezik már szülő alklaszterrel és ott van lehetőség új alcsoportot létrehozni, akkor megtesszük és kettéosztjuk a szülő alklasztert, ha pedig nem, akkor ugyan ketté választjuk, de a folyamatot tovább folytatjuk rekurzívan egészen a gyökér csomópontig. A fentiek szemléltetésére a 5.8 ábracsoporton látható a küszöbérték hatása a kialakuló klaszterszámra (globális klaszterezést nem használva) az MNIST adathalmazon.



(a) $T = 0.5$ **(b)** $T = 0.3$

(c) $T = 0.2$ **(d)** $T = 0.15$

(e) $T = 0.1$ **(f)** $T = 0.075$

5.8. ábra. T küszöbértékű BIRCH algoritmusok eredményei az MNIST adathalmazon tanított autoenkóder látens terének t-SNE leképezésén

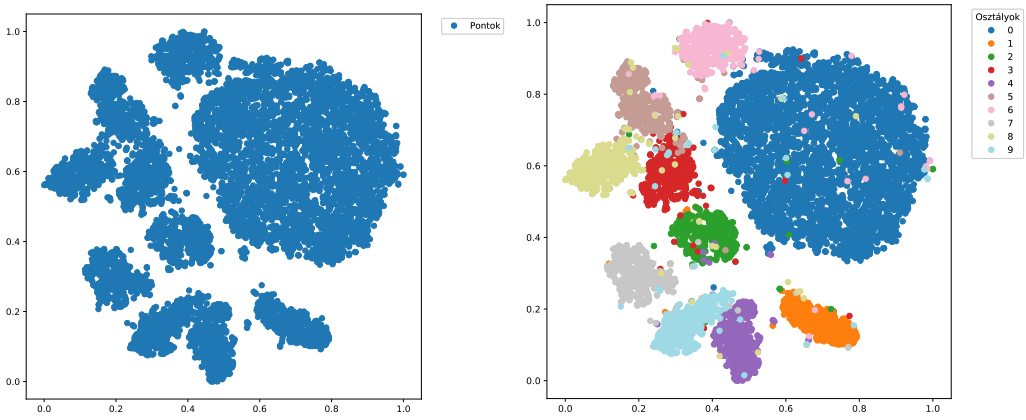
A fenti ábrákon jól látható, hogy ha a küszöbértéket túlságosan nagyra választjuk, akkor az algoritmus egy nagy klaszterbe csoportosítja az egész adathalmazt (5.8a ábra), ha viszont elkezdjük csökkenteni, akkor elkezdenek megjelenni a klaszterválasztó határok, melyek jól definiált egyeneseket jelentenek. Ez lényegében az algoritmus távolságszámításából adódik és bizonyos esetekben kicsit mesterségesen, erőltetetten ható csoportokat hoz létre emiatt (ahogy a 5.8b és 5.8c ábrák klaszterhatárain is látható). Egy bizonyos határ után azonban nem érdemes tovább csökkenteni a küszöbértéket, hiszen túlságosan sok

klasztert eredményez, feldarabolva így az addig egyben lévő ponthalmazokat (lásd a 5.8e és 5.8d ábra nagy halmazát, valamint a 5.8f ábra kisebb halmazait). Célszerű tehát gondosan megválasztani a küszöbérték paramétert úgy, hogy lehetőleg a két szélső véglet között legyünk, azonban, ha alkalmazunk globális klaszterezést, akkor a hatása nem fog ennyire érződni a kapott eredményeken. Érdemes megjegyezni, hogy manapság már léteznek olyan módosítások, amelyek automatikusan egy optimális értékre választják a küszöböt a megadott feltételeknek megfelelően[19], azonban a dolgozat során ez nem szükséges a kívánt klaszterszám megadása miatt, amelynek folyamata a következő részben kerül ismertetésre.

5.3.2. Klaszterezés felhasználói beavatkozással

Az előző részfejezetben bemutatásra került a BIRCH klaszterezési algoritmus, amelyet alkalmazni fogunk, azonban látható volt, hogy nem egyszerű a kívánt klaszterszámot és szerkezetet elérni: ha alkalmazunk globális klaszterezést, akkor kötöttek a kialakuló csoportok határai, míg ha nem használunk, akkor ugyan nagyobb a befolyásunk, mégis sokszor nehéz a kívánt számú klasztert előállítani. Ennek a problémának a megoldására nyújt egy lehetőséget a felhasználói beavatkozással kiegészített klaszterezés, amely ebben a részben kerül bemutatásra.

Első lépésként határozzunk meg egy olyan elérendő klaszterszámot, amely alapján a választott klaszterező algoritmusunk megfelelően tudja kategorizálni a kapott látens teret. A szükséges szám meghatározásához minden esetben a kialakult látens tér szerkezetéből kell kiindulni, hiszen ott akár „szabad szemmel” is láthatjuk, sejthetjük (egy jobb kódtér esetében), hogy mely ponthalmazok alkotnak egy-egy osztályhoz rendelhető csoportot, azonban sokszor ez az algoritmus számára problémát jelenthet, ezért szükséges a számuk növelése a ténylegesnél. Jó példa erre a 5.9 ábrapár, ahol egy látens tér ábrája látható a minták címkéivel és azok jelölése nélkül.

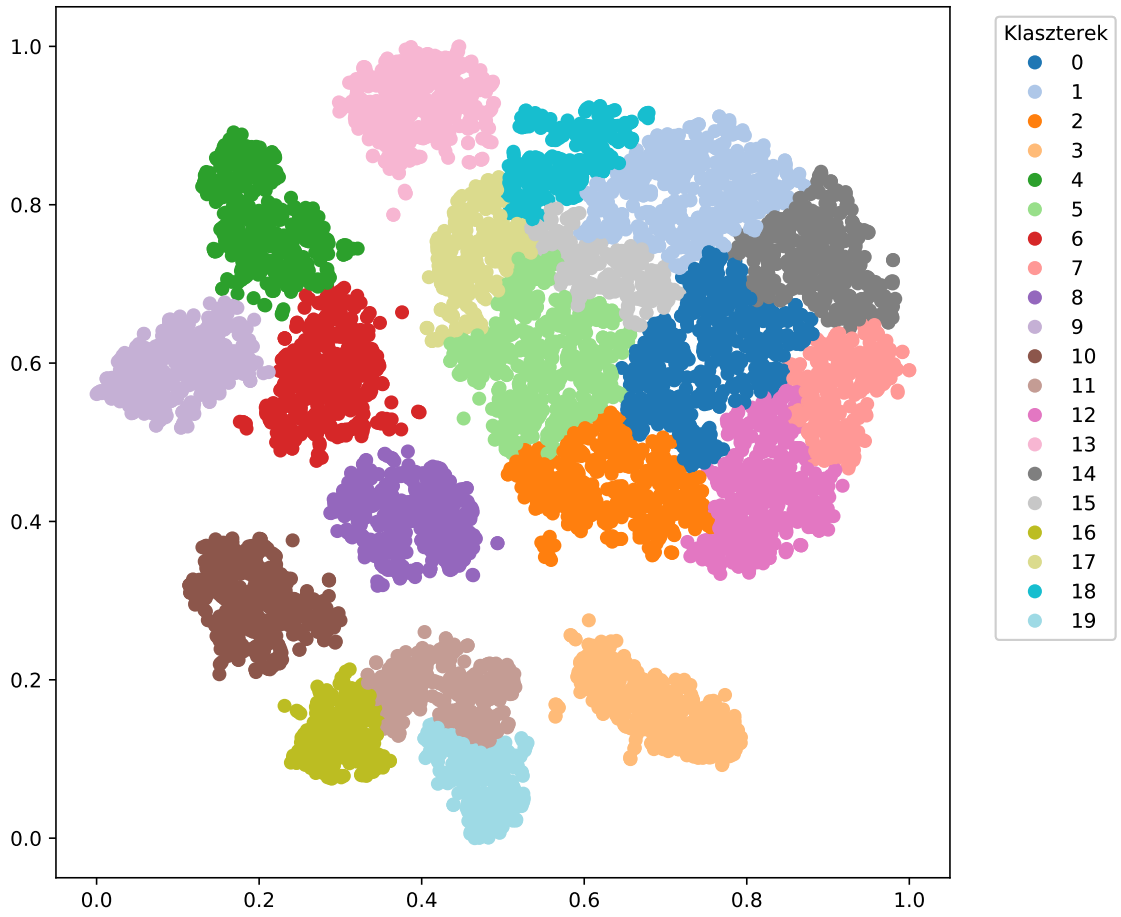


(a) Címkék nélkül **(b)** Tényleges címkékkel

5.9. ábra. Az MNIST adathalmazon tanított autoenkóder t-SNE ábrái címkékkel és azok nélkül

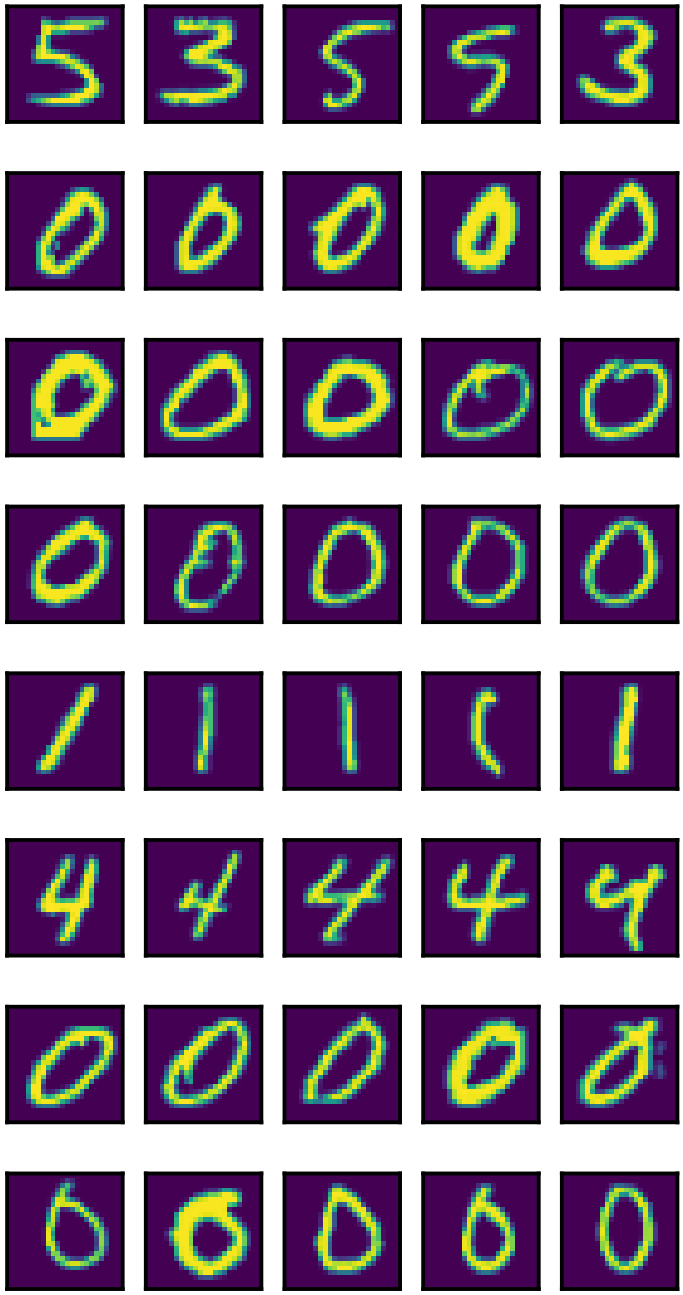
A 5.9a ábrán jól láthatóak az egyes pontklaszterek és mivel előre tudható, hogy hány osztályunk van, ezért könnyen kitalálhatóak az egyes csoportok helyzetei, azonban az algoritmus számára a határvonalak összeérése problémát okozhat: lásd például a 5.9b ábra 9-es és 4-es csoportjait, amelyek esetében az algoritmus egy klaszternek venné őket, míg a legnagyobb 0-s klasztert kettébontaná. Ezt úgy tudjuk kiküszöbölni, hogy szándékosan annyival több csoportot adunk meg bemenetként, hogy ezek a problémás csoportok is biztosan különváljanak egymástól az algoritmus futása során: ez az ábrázolt esetben 15 és 20

közötti klaszterszámot jelent. Így lefuttatva a klaszterező algoritmust a fenti példán a 5.10 ábrához hasonló elrendezést kapunk.



5.10. ábra. A BIRCH algoritmus $N = 15$ klaszterszámmal való lefuttatásának eredménye az MNIST adathalmazon

Ezen az ábrán az látható, hogy sikerült a problémás határokat különválasztanunk, azonban ezzel egyidejűleg más, addig egybefüggő részeket feldaraboltunk több kisebb csoportra (ez legjobban a 0-s halmazon látható). Emiatt a következő lépésként ezeket a „felesleges” klasztereket kell összevonnunk nagyobb csoportokká, ezzel kialakítva a ténylegesen elvárt csoportszámot. Erre egy megoldás a felhasználó bemenete, mely segítségével eldönthetjük, hogy mely csoportok tartalmaznak azonos osztálybeli mintákat. Ez a gyakorlatban úgy tehető meg, hogy felvillantunk példamintákat az egyes klaszterekből a felhasználó számára, aki ezek alapján eldöntheti, hogy valóban mely osztályba tartozhatnak. Az egyes algoritmus által előállított klaszterek példamintáira a 5.11 ábra mutat példát.



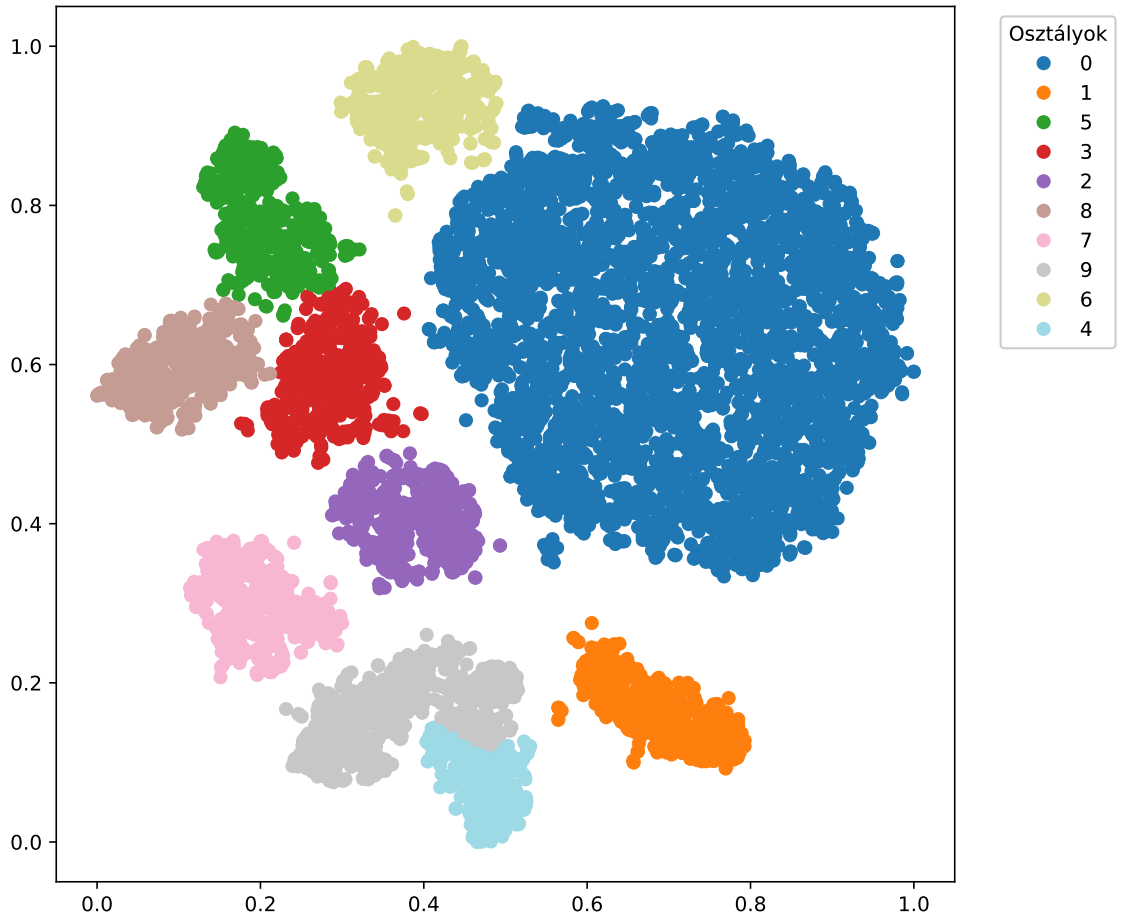
5.11. ábra. Az első néhány klaszterből (sorok) véletlenszerűen kiválasztott minták ábrái

Maga a minták kiválasztása itt egyszerűen a véletlen módszer szerint történik, amely a klaszterek eloszlása szerint reprezentálja azok hovatartozását. Természetesen lehet ennél komplexebb módszereket is használni, attól függően, hogy milyen minőségű a kapott látens tér szerkezete, mégis már ezzel is megfelelő eredmények érhetőek el. Ha például a felhasználói beavatkozás után még mindig nagyon sok egybelogó, összemosódó klasztert kapnánk, akkor ahhoz, hogy minél kiegyensúlyozottabban mintavételezzük a csoportokat, szükséges lenne például elsődlegesen a klaszterek középpontjából válogatni.

Megfigyelhető azonban az, hogy egyes csoportok eltérő kategóriákba tartozó mintákat is tartalmaznak (lásd például a 5-ös sorát, ahová kettő 3-as is bekerült), amely esetén, ha

esetleg csak $1-2$ adatot mintavételeznénk, akkor hibás következtetésre jutnánk a valódi osztályt tekintve. Ahhoz, hogy ezt elkerüljük, célszerű lehetőleg minél több mintát választani az adott csoportból (jelen példában 5-öt), hogy megfelelő képet kapjunk a benne lévő osztályok eloszlásáról. Ha esetleg a látott minták alapján nem dönthető el egyértelműen a megfelelő kategória, akkor vagy növeljük a példaminták számát vagy nagyobb kezdeti klaszterszámot választunk és megismételjük a BIRCH futtatását.

Miután sikerült „felcímkéznünk" az egyes klasztereket és összevontuk az azonos címkét kapott halmazokat, ellenőrizhetjük a módosított csoportosítás szerkezetét, az egyes klaszterek elhelyezkedését. Ezt a 5.12 ábra szemlélteti, ahol már nevezhetjük osztályoknak a kialakult csoportokat, hiszen a felhasználói bemenet révén kapták a nevüket. Érdemes ezen kívül megfigyelni, hogy mely eredeti klaszterekből alakultak ki a végleges csoportok, illetve melyek olvadtak össze, valamint az egyes kategóriák határvonalai hogyan változtak meg.

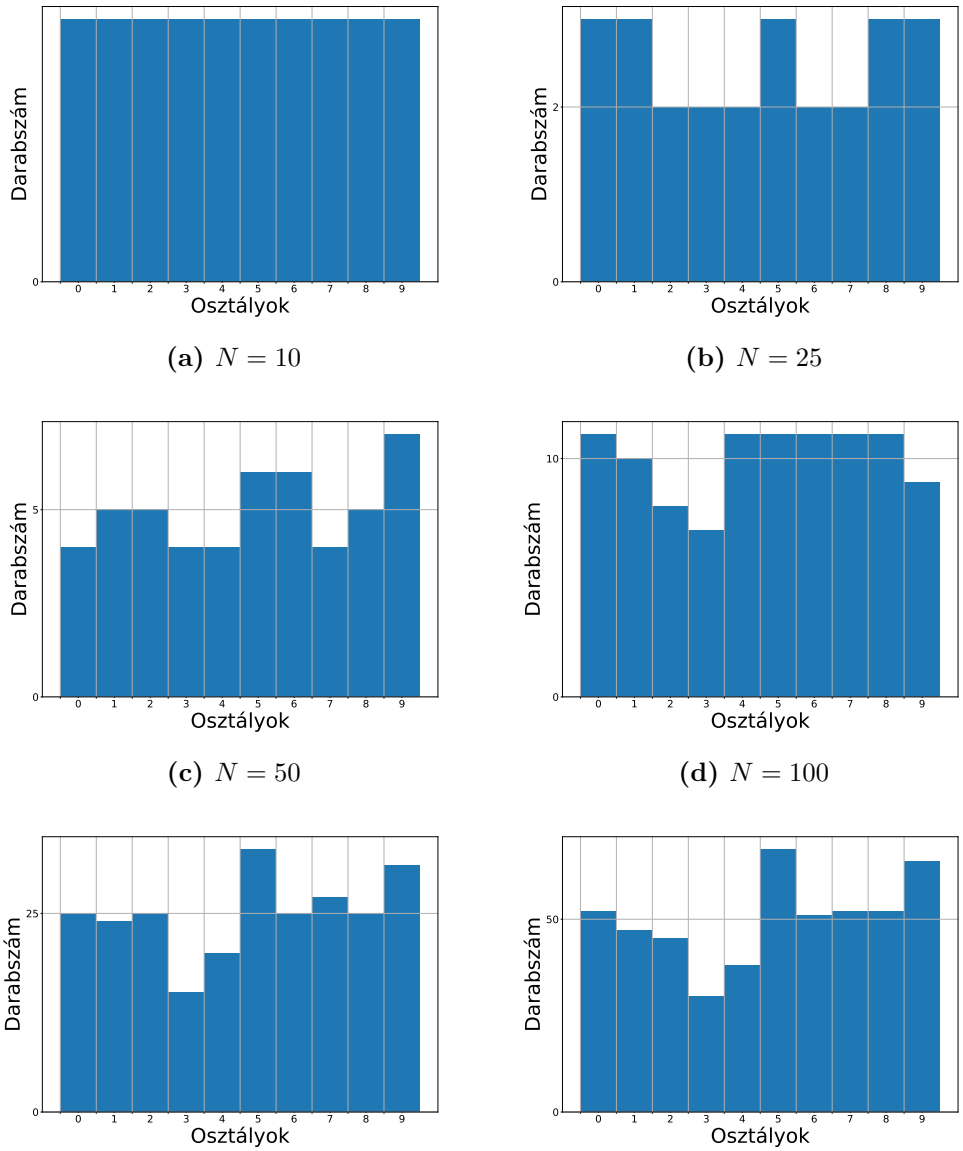


5.12. ábra. A felhasználói beavatkozás eredményének ábrázolása

A kapott ábrát és a tényleges mintaeloszlást (lásd 5.9b ábra) összehasonlítva látható, hogy sikerült egy majdnem tökéletes csoportosítást végrehajtanunk: egyedül a 7-es és a 9-es számjegyek halmazai lógnak egymásba, de ez már a mintavételezés során is előkerült, mint lehetséges probléma. Ezek alapján tehát elmondható, hogy a felhasználói beavatkozással a klaszterezés nagy mértékben javítható viszonylag kis járulékos munka elvégzésével, hiszen jelen esetben mindössze 15 mintasort kellett felcímkéznünk a látott javulás eléréséhez.

5.3.3. A minták kiválasztása a kapott klaszterek alapján

A fentiek elvégzése után következhet az utolsó lépés: a minták kiválasztása és azokból a részadathalmazok létrehozása. Mivel már rendelkezünk a teljes adathalmaz látens terének megfelelő minőségű, a felhasználó által ellenőrzött klaszterezésével, ezért lényegében hagyatkozhatunk a véletlen kiválasztás módszerére. Természetesen, ha nem vagyunk megelégedve a kapott csoportok elhelyezkedésével, szerkezetével, akkor választhatunk összetettebb módszert is, azonban a lényeg, hogy az egyes kialakult klaszterekből külön-külön, egymástól függetlenül válogassunk. Ezzel elérhető, hogy a kapott részadathalmaz osztályeloszlása minél kiegyensúlyozottabb legyen, hiszen feltehető, hogy az egyes klaszterek főként egy-egy adott osztályból állnak. A véletlen kiválasztás ilyenkor lényegében abból áll, hogy a kívánt adathalmazméretet felbontjuk lehetőleg egyenlő részekre a klaszterszámtól függően és minden csoportból ugyanannyi mintát vételezünk. Mivel az eredeti adathalmaz osztályeloszlása egyenlőtlen, így várhatóan a kapott klaszterek mintaszáma is nagyban eltérő lesz, így nagyobb célhalmazméretek esetén a kiválasztott minták nagy része egy csoportból fog származni. Azonban e pont elérése előtt célszerű minél jobban törekedni az egyenletes kiválasztásra az aktuálisan még választható mintát tartalmazó klaszterek között. Egy ilyen véletlenszerű választás eredményeképpen előálló részadathalmazok eloszlásait szemlélteti a 5.13 ábra, ahol az előzőekben kapott MNIST klaszterezés lett felhasználva.



(a) $N = 10$ **(b)** $N = 25$

(c) $N = 50$ **(d)** $N = 100$

(e) $N = 250$ **(f)** $N = 500$

5.13. ábra. N méretű, klaszterező módszerrel generált részadathalmazok eloszlása az MNIST adathalmazból kiindulva

Az eloszlásokból az látható, hogy sikeres volt a klaszterezésen alapuló mintaválasztás, hiszen a naiv módszer eredményeivel összehasonlítva (lásd 5.1 ábra) sokkal egyenletesebb osztályeloszlásokat értünk el már kis mintaszámok esetén is. Legtökéletesebb a 10 minta esete (5.13a ábra), ahol minden egyes osztályból pontosan 1-1 minta került bele a részhalmazba, azonban a többi eloszlás esetében is közel sikerült kerülnünk az ideális egyenletességhez, amit az ábrán bejelölt vízszintes vonal szemléltet. Másik fontos különbség a naiv módszernél tapasztaltakhoz képest, hogy a nagyobb részhalmazok esetében is sikerült megtartani az egyenleteshez közeli reprezentációt (lásd 5.13e és 5.13f ábrák), holott ilyenkor már könnyen kiütközhetne a teljes adathalmaz egyenlőtlensége, ahogy azt korábban láthattuk (lásd 5.1e és 5.1f ábrák), azaz az egyik osztály alkotná a részhalmaz közel 80%-át.

6. fejezet

A kísérletek elvégzése és eredményük értékelése

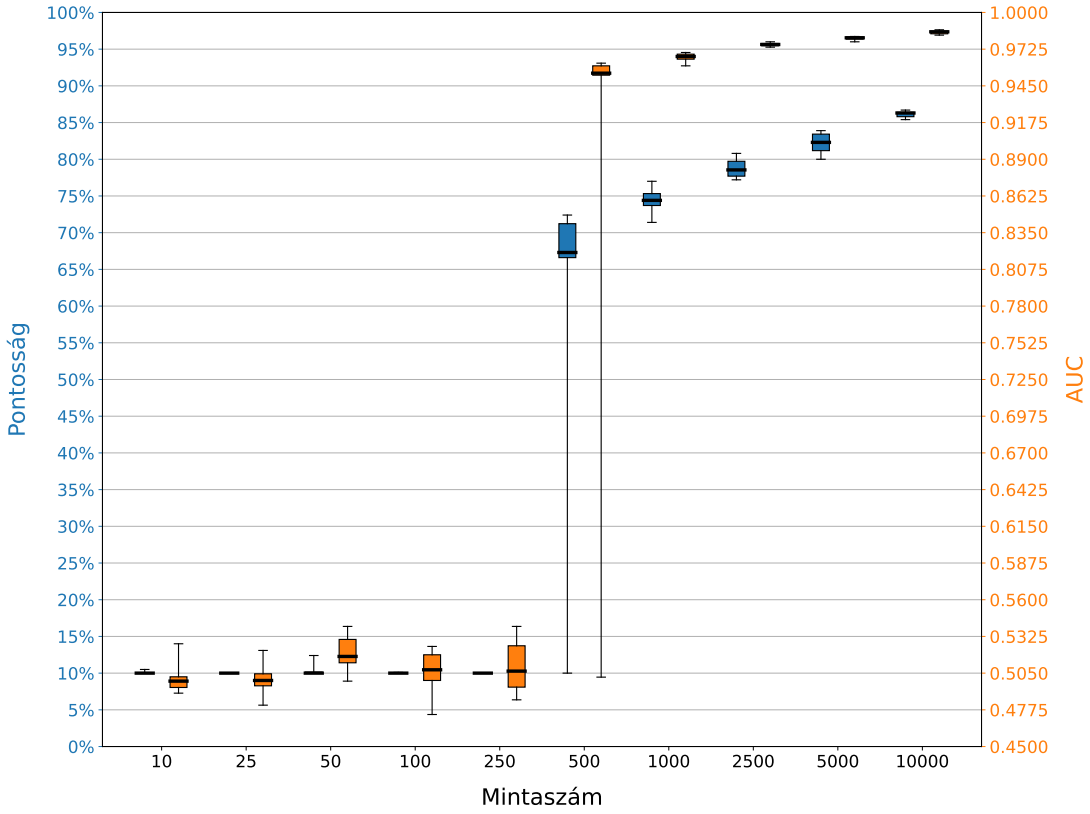
Az eddigiekben ismertetésre kerültek a dolgozat során felhasznált adathalmazok, a kapott eredmények értékeléséhez használt metrikák és a minták kiválasztásához alkalmazott módszerek. Ezeket felhasználva elvégezhetjük az általunk definiált kísérleteket és értékelhetjük, illetve egymással összehasonlíthatjuk. Ebben a fejezetben tematikusan, külön részekre bontva -az adathalmaz, a módszer és a látens tér előállításához használt hálózatarchitektúra alapján- találhatóak az egyes kísérletek és eredményeik, illetve minden esetben összehasonlításra kerülnek az alapszintet képviselő naiv módszerrel. Célszerű itt a fejezet elején megjegyezni, hogy az egyes különböző komplexitású adathalmazok esetén más-más paraméterszámú és összetettségű osztályozó konvolúciós hálózatot használtam fel a megfelelő eredmények eléréséért, azonban ezeken belül az egyes módszerek ugyanazt a hálózat struktúrát használják a kiértékelő lépés elvégzésekor. Minden kísérletet $10 - 10$ különböző inicializálású hálózaton futtattam, majd a kapott eredményekhez mintaszámonként számítottam szórást és mediánt, valamint minimális és maximális értéket. A kapott eredményeket minden kísérlet esetében egy közös ábrán tüntettem fel, így segítve az összehasonlításukat.

6.1. Az MNIST adathalmazon végzett kísérletek

Elsőként az MNIST adathalmazon végeztem el a kísérleteket, hiszen szerkezetét tekintve ez a legegyszerűbb felépítésű adathalmaz a három közül, így könnyebben érhető el rajta viszonylag jó eredmény, mégis alkalmas a különböző módszerek több szempontból történő összehasonlítására.

6.1.1. A naiv módszer eredményei az MNIST adathalmazon

A bemutatott módszerek közül a naiv módszer eredményeinek ismertetésével érdemes kezdeni, hiszen ezekhez képest nyújtanak javulást a látens téren alapuló módszerek. Mivel azonban az MNIST adathalmaz viszonylag egyszerűen tanulható, így már a véletlenszerű kiválasztáson alapuló eljárás is képes nagyobb mintaszámok esetén jó eredményt elérni. A kapott eredmények szemléletesen a 6.1 ábrán láthatóak az egyes előre meghatározott mintaszámú részadathalmazok esetén.



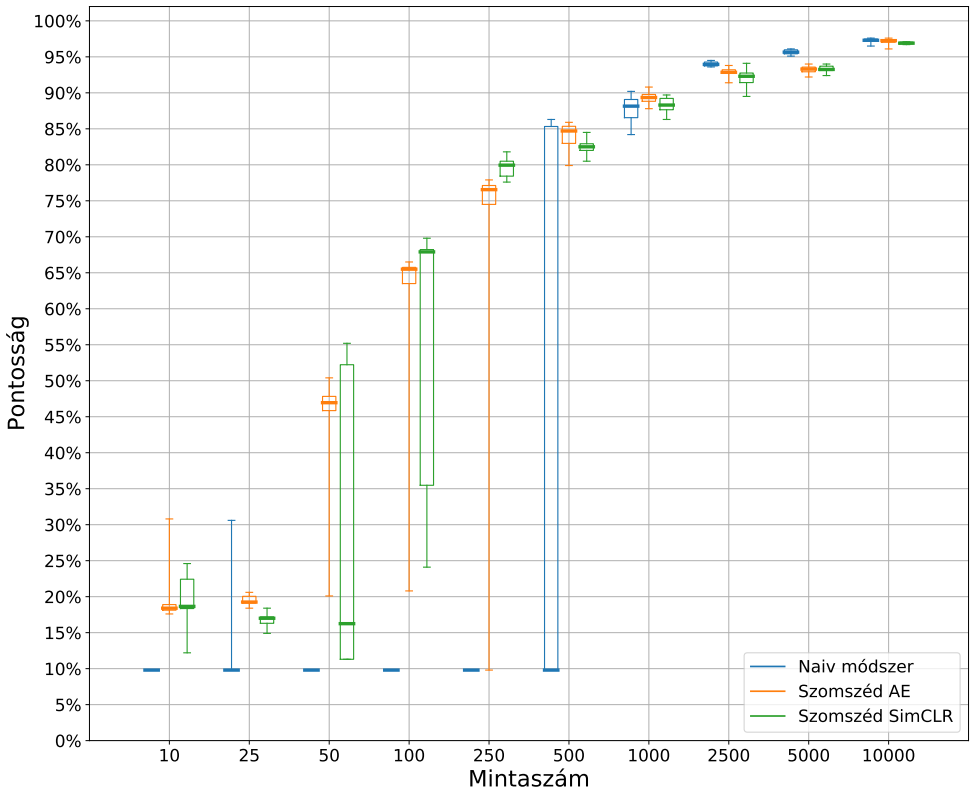
6.1. ábra. A naiv módszer pontosság és AUC eredményei a mintaszámok függvényében az MNIST adathalmaz esetén

Az ábráról szemléletesen látható, hogy egészen 1000-es mintaszámig a hálózat a véletlenszerű minták egyenlőtlen eloszlása miatt nem képes lényeges javulásra, majd hirtelen (500 mintánál) rohamosan megnő a kapott pontosság. A 10% környéki értékek azzal magyarázhatóak, hogy ilyenkor a hálózat a tanuló részadathalmaz alapján csak a leggyakrabban előforduló osztályt tartja érdemesnek megtanulni (az eloszlásokhoz lásd a 5.1 ábracsoportot), hiszen ezzel képes elérni a legjobb eredményt, a többi kategóriáról egyszerűen nem kap elegendő információt, hogy érdemes legyen velük is foglalkoznia. Ez a jelenség nyomon követhető az AUC értékek alakulása esetén is: megfigyelhető, hogy egy bizonyos ponton túl kezd csak el javulni a hálózat teljesítménye, addig nem sokkal jobb a teljesen véletlen szintet képviselő 0.5-ös értéknél.

Mindkét metrika esetében látható, hogy a teljes adathalmaz esetén jónak tekinthető eredményeket kapunk, tehát a hálózat képes a feladat megoldására, ha elegendő adatot lát az egyes osztályokból. A kapott eredmények alapján tehát a látens téren alapuló módszerek célja, hogy az itt tapasztalt kezdeti stagnálást feloldják, és lehetővé tegyék a kisebb részhalmazokon történő hatékony tanítást.

6.1.2. Szomszédsági számon alapuló módszer eredményei az MNIST adathalmazon

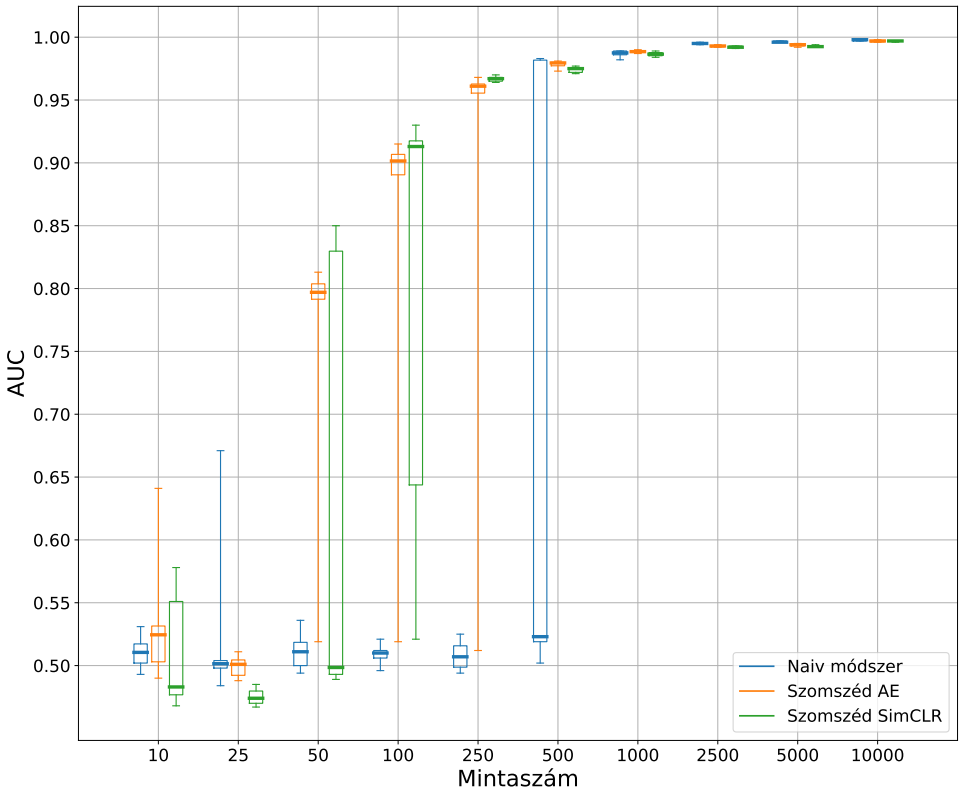
A fentebb kapott, kiindulásként tekinthető eredményeken javítanak a látens téren alapuló módszerek, amelyek közül az első a szomszédsági számon alapuló eljárás, amelyhez először elő kell állítani egy alkalmas kódteret: ezt elsőként autoenkóder hálózattal tesszük meg, majd pedig SimCLR hálózattal. Fontos megjegyezni, hogy a SimCLR-nél alkalmazott enkóder hálózat megegyezik az autoenkódernél használtal, így könnyítve az összehasonlítást a kettő között. A kapott látens térben ezután az egyes mintákat szomszédsági számuk alapján nagyság szerint csökkenő sorrendbe rendezzük és a bemutatott szabály szerint válogatunk belőlük. Ezzel a módszerrel elérhető, hogy a kisebb mintaszámú részadathalmazok esetén is lehetőleg olyan mintákat tanuljon a hálózat, amelyek csökkentik az osztályok eloszlásának egyenlőtlenségét. A kapott eredmények közül a 6.2 ábrán a pontosság értékek láthatóak a különböző mintaszámok szerint.



6.2. ábra. A szomszédságon alapuló módszer pontosság eredményei a mintaszámok függvényében az MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

Az ábrán rögtön szembetűnő a változás az autoenkóder esetében a naiv módszernél kapott értékekhez képest: a hálózat pontossága már 50 mintánál elkezd javulni, sőt a 10-es mintaszámnál kapott érték is majdnem duplája az előzőleg tapasztaltnak. Ezeken túl a mintaszám növekedésével a kapott pontosság értékek is elkezdenek arányosan nőni, lényegében kitöltve azt a „hézagot”, amelyet a naiv módszernél láthattunk 10 és 500

minta között. A SimCLR hálózat esetében kapott eredményeket összevetve az autoenkóder esetében kapottakkal az tapasztalható, hogy a SimCLR hálózatnak sikerült már 10 minta esetén is az alap 10%-ot felülmúlnia. A fentieknek megfelelően változtak a kapott AUC értékek is, amelyek a 6.3 ábrán láthatóak.



6.3. ábra. A szomszédságon alapuló módszer AUC eredményei a mintaszámok függvényében az MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

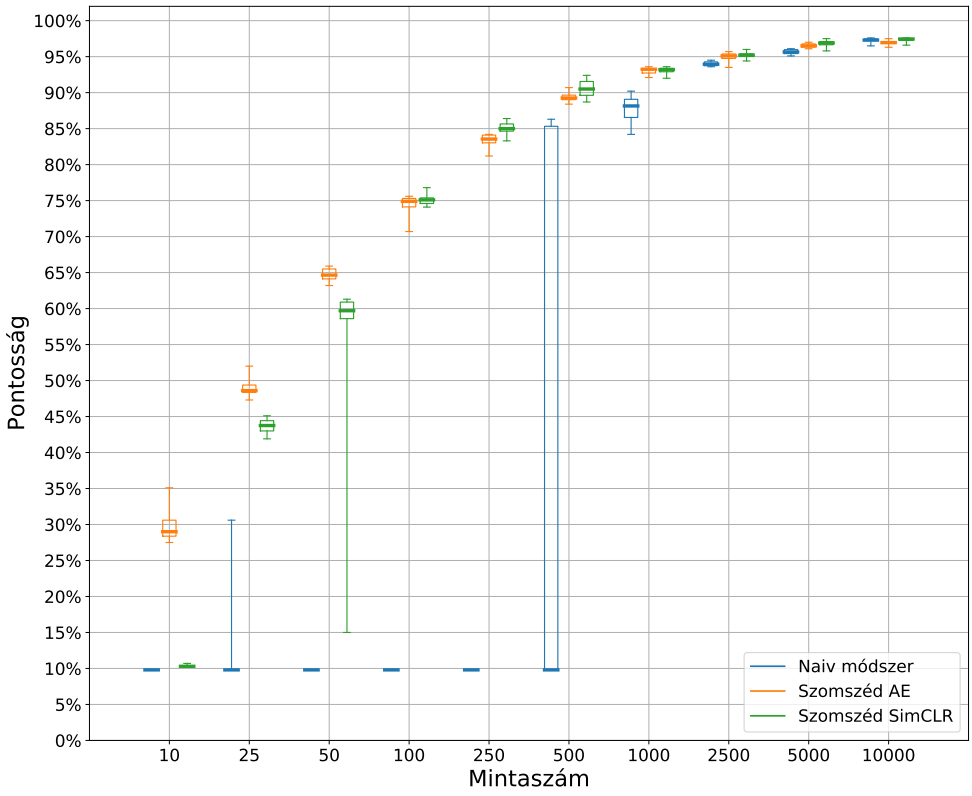
Itt először a naiv esethez hasonlóan (lásd 6.1 ábra) az értékek lényegében a véletlen választás szintjén vannak, majd ugyancsak 50 mintától kezdve látható egy folyamatos javulás, amellyel a 250-es mintaszámot elérve már közel egyenlő értéket kapunk a naiv módszer 500 mintájánál kapottal. Ezen kívül mindkét ábrán megfigyelhető, hogy az egyes mintaszámokhoz tartozó értékek szórása sokkal kisebb, mint az előzőleg tapasztaltak, tehát nagyobb konfidenciával lehet a kapott medián értékekre, mint a hálózat várható teljesítményére hivatkozni. A másik hálózat esetében itt is a fentiekhez hasonló az összehasonlítás tapasztalata: a SimCLR látens terén alkalmazott hálózat kisebb mintaszámok esetén is képes tanulni, illetve az egyes AUC értékek tartománya is sokkal jobban leszűkült az autoenkódernél látottakhoz képest.

A fenti eredmények alapján tehát elmondható, hogy a szomszédságon alapuló módszerrel sikerült javítani a hálózat teljesítményét a kisebb mintaszámok esetén is, így lehetővé téve, hogy minél kevesebb mintát kelljen annotálni a kívánt értékek eléréséhez. Ezen kívül kijelenthető, hogy a SimCLR látens terén hatékonyabban lehet elvégezni a szom-

szédságon alapuló mintakiválasztási módszert, így a módszert akár 25-ös mintaszámra is kiterjesztve és nagyobb megbízhatóságot eredményezve, mint az autoenkóder esetében.

6.1.3. Klaszterezésen alapuló módszer eredményei az MNIST adathalmazon

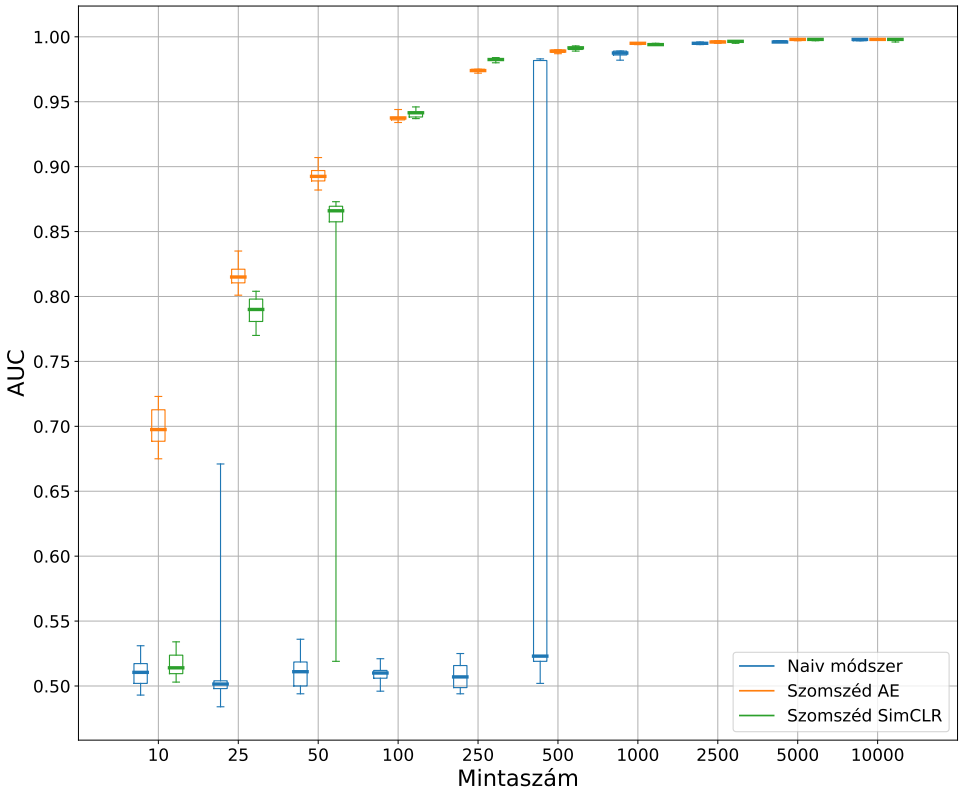
A szomszédságon alapuló módszer vizsgálata után tekintsük a felhasználói beavatkozással kiegészített klaszterezés folyamatát. Ebben az esetben is külön kezeljük a két hálózattípus alapján a kísérleteket, így egyik esetben az autoenkóder hálózat által előállított látens téren vizsgáljuk a kapott eredményeket, majd következik a SimCLR hálózat által előállított kódtéren elért eredmények elemzése. A vizsgált metrikák közül elsőnek a kapott pontosság értékek kerülnek bemutatásra, amelyek változása a 6.4 ábrán látható.



6.4. ábra. A klaszterezésen alapuló módszer pontosság eredményei a mintaszámok függvényében az MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

Itt megfigyelhető, hogy az autoenkóder esetében sikerült már a 10-es mintaszámnál is akár közel 35%-os eredményt elérni, illetve sikerült kiküszöbölni az értékek nagy szórását a 25-ös és a 100-as esetekben, azonban a 100-as mintaszámnál mégis egy visszaesés tapasztalható. Ezzel szemben a SimCLR látens terét használva a kapott értékek a kisebb mintaszámok eseteinek nagy részében szűkebb tartománnyal rendelkeznek, illetve a 10-es és a 100-as mintaszámoknál felül is múlják az autoenkódert. A pontosság javulásában itt a 25-ös méretnél történik egy visszaesés, amikor a naiv módszer szintjére romlik vissza a

hálózat teljesítménye. Ezeket az ismertetett jelenségeket szemlélteti az 6.5 ábra is, ahol az AUC értékek változásának folyamata látható.



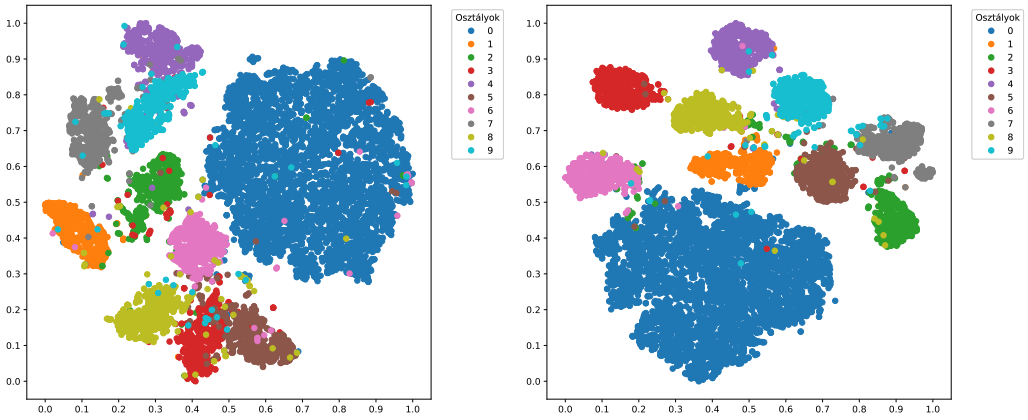
6.5. ábra. A klaszterezésen alapuló módszer AUC eredményei a mintaszámok függvényében az MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

Az AUC értékeket tekintve hasonló tapasztalatok vonhatóak le: az autoenkóder esetében a kapott értékek nagy tartománnyal ugyan, de jól indulnak, majd a 100-as mintaszám esetén hirtelen visszaesnek a véletlen kiválasztás szintjére, ezzel elrontva a javulás folyamatát. A SimCLR esetében azonban a kezdeti magas értéket rögtön egy letörés követi a 25-ös méretnél, de ezután a javulás folyamata visszatér a megkezdett szintre és végig töretlenül fut a mintaszámok növelésével.

A fenti észrevételeket összegezve elmondható, hogy mindkét hálózat látens terének sikerült javítania az előzőleg kapott eredményeken, mégis a SimCLR esetében nagyobb mértékű volt a fejlődés, illetve a tartomány lecsökkenése, mint az autoenkóder esetében, ahol ráadásul a 100-as mintaszámnál bekövetkezett visszaesés megtörte az értékek folyamatos javulását.

6.1.4. Az eredmények viszonya a látens terekhez

A megfogalmazott észrevételek magyarázatához tekintsük a 6.6 ábrát, ahol a kettő hálózat látens terének t-SNE ábrája látható egymás mellett.



(a) Autoenkóder hálózat **(b)** SimCLR hálózat

6.6. ábra. Az MNIST adathalmazon tanított hálózatok t-SNE ábrái

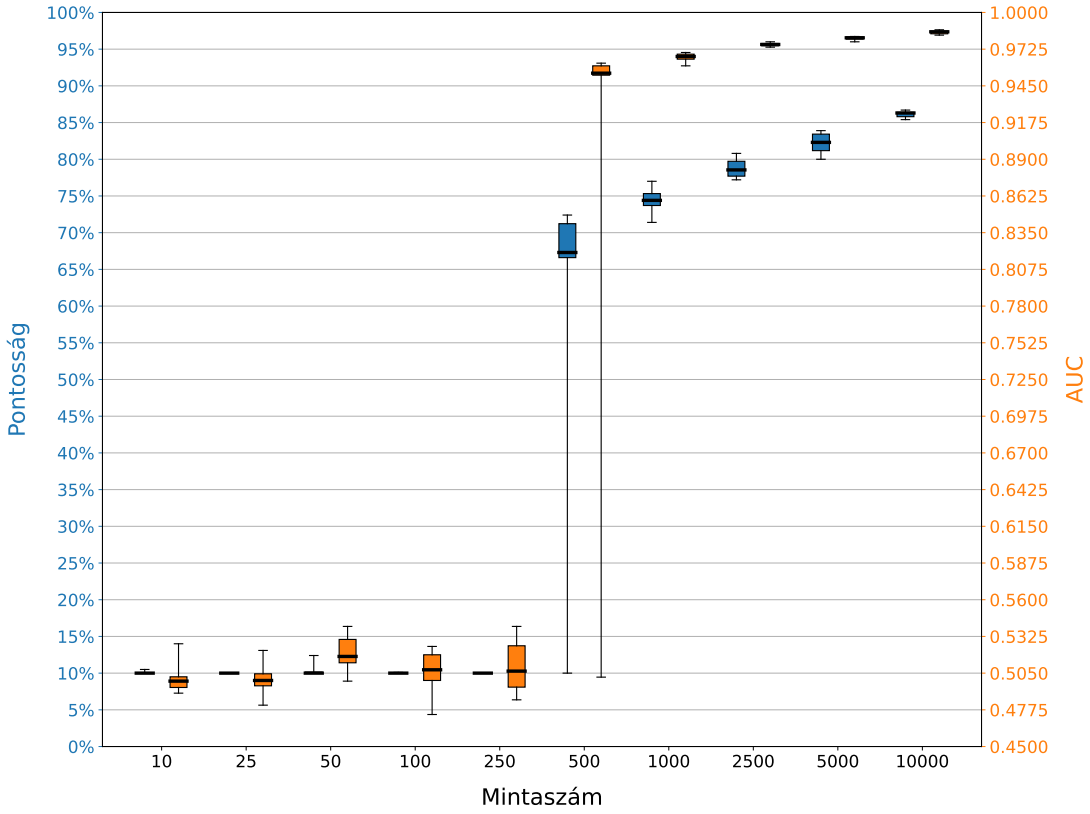
A SimCLR esetében a kapott klaszterek sokkal jobban szétváltak, mint az autoenkóder esetében, ahol egyes csoportok egymásba lógnak, azonban ezen kívül a két látens tér szerkezetének minősége között lényeges eltérés nincsen. Ezek alapján már érthetőek a kapott eredmények: a mintakiválasztó módszereknek sikerült egyenletes eloszlású adathalmazokat generálnia, így a látens terek kis minőségbeli eltérését kompenzálni tudták, ráadásul sok esetben éppen az autoenkóder adott kicsivel jobb eredményt a kettő architektúra közül. Összefoglalva tehát, viszonylag egyszerű adathalmazok esetén (ahol a két hálózat hasonló minőségű látens teret ad) a mintakiválasztó módszerek egyenrangúnak tekinthetőek és így minden kísérleti összeállítás releváns teljesítményt tud nyújtani.

6.2. A Fashion-MNIST adathalmazon végzett kísérletek

A következő vizsgált adathalmaz a Fashion-MNIST halmaz, amely sok tekintetben hasonló az előzőhöz, mégis az egyes ruhadarabok egyedisége miatt nehezebben tanulható. Ennek megfelelően több paraméterű, összetettebb osztályozó konvolúciós hálózatot kell alkalmazni, illetve az autoenkóder és a SimCLR blokkjainak méretet és számát, valamint a kódszóhosszúságot is meg kell növelni a teljesítmény növeléséhez.

6.3. A naiv módszer eredményei

A bevezető után tekintsük először a naiv módszer által adott eredményeket, amelyek a 6.7 ábrán láthatóak.



6.7. ábra. A naiv módszer pontosság és AUC eredményei a mintaszámok függvényében a Fashion-MNIST adathalmaz esetén

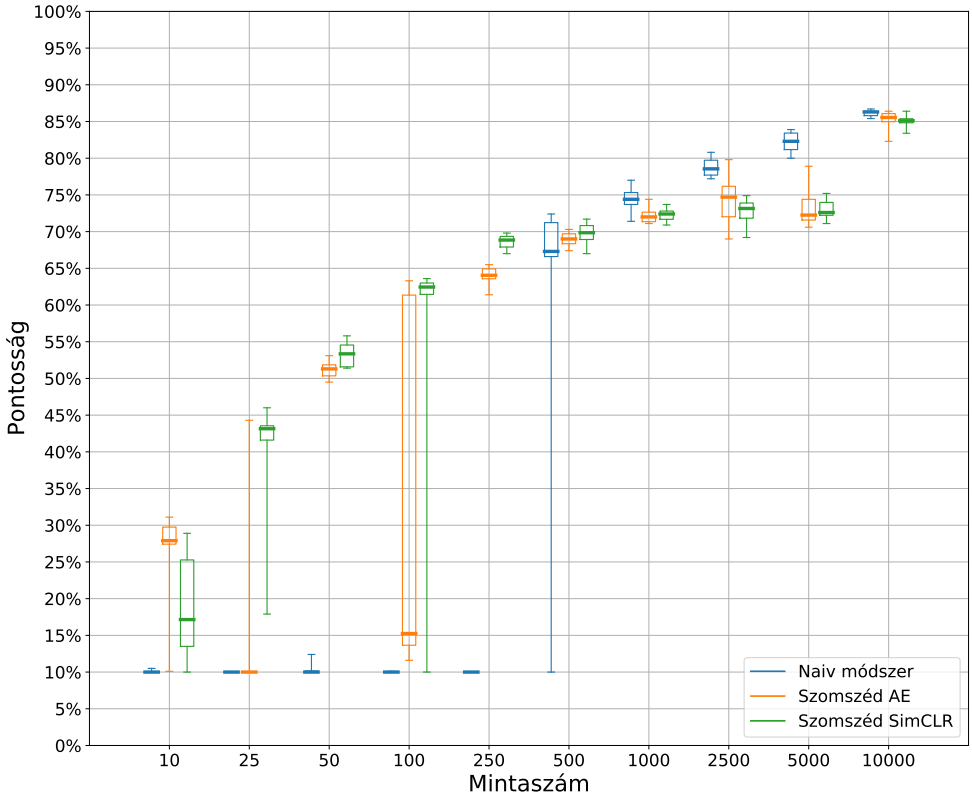
Lényegében ugyanaz figyelhető meg a pontosság értékeknél, mint az MNIST esetében: kezdetben a hálózat nem képes a legnagyobb darabszámú osztályon kívül más kategóriákat megkülönböztetni a részhalmaz egyenlőtlen eloszlása miatt. Ez valamiképp javul 500 minta esetén, azonban igazán csak ezután kezd el a hálózat megfelelően teljesíteni. Hasonlóan alakulnak az AUC értékek, amelyeken ugyancsak jól látható, hogy 500 minta alatt a hálózat alig tudja felülmúlni a véletlen választás szintjét, és csak a nagyobb adathalmaz méretek esetén képes arányosan javulni. Fontos megjegyezni, hogy ennél az adathalmaznál az alkalmazott osztályozó hálózatarchitektúra csak $85-90\%$ közötti maximális pontosság értéket tudott elérni, a paraméterek és a rétegek többszöri ismételt növelése ellenére is. Ez alapján kijelenthető, hogy a Fashion-MNIST mintáiból álló adathalmaz nehezebben tanulható, mint azt elsőre a bevezetőben leírtam.

A fentiek alapján újból adott a feladat, miszerint a látens téren alapuló módszerek segítségével el kell érni, hogy a kisebb mintaszámok esetén is megfelelő teljesítményt nyújtson az osztályozó hálózatunk.

6.3.1. Szomszédsági számon alapuló módszer eredményei

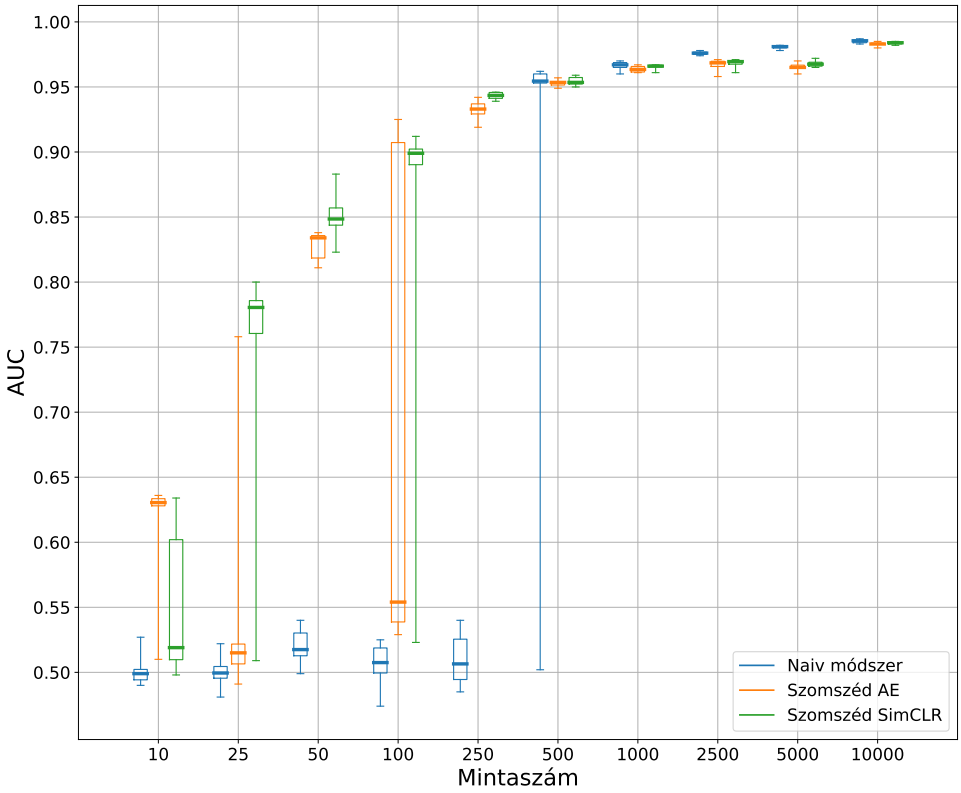
A naiv módszer kiértékelése után a szomszédsági számon alapuló módszerek eredményeinek elemzése következik, külön kísérletet végezve az autoenkóder és a SimCLR háló-

zatok látens terét illetően. A kapott pontosság értékek a naiv módszerrel összehasonlítva a 6.8 ábrán láthatóak.



6.8. ábra. A szomszédságon alapuló módszer pontosság eredményei a mintaszámok függvényében a Fashion-MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

Az autoenkóder értékeit tekintve tapasztalható viszonylag nagy javulás a kisebb mintaszámok esetében (különösen 10-nél), azonban a 25-ös és a 100-as méretek esetén még mindig nagy az értékek tartománya, valamint a medián értéke erősen visszaesik. Ehhez képest a SimCLR esetében a kezdeti érték kisebb a 10-es mintaszámnál, azonban a javulás jobban fennmarad: 25 és 100 darab minta esetében is javul a medián értéke, ugyan a tartomány itt is viszonylag széles. Ezek után tekintsük az AUC értékeket, amelyek a 6.9 ábrán láthatóak.



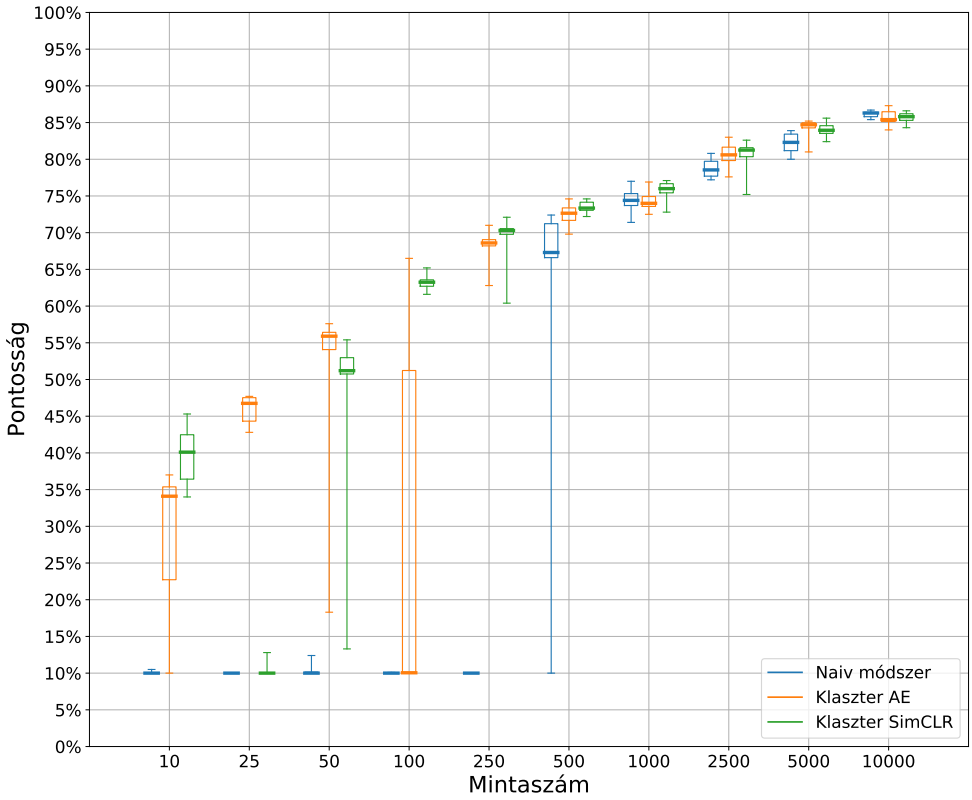
6.9. ábra. A szomszédszágon alapuló módszer AUC értékei a mintaszámok függvényében a Fashion-MNIST adathalmaz esetén a naiv módszerrel összehasonlítva

Jellegét tekintve itt is megjelenik a kisebb mintaszámoknál az autoenkóder esetén számottevő javulás, azonban ugyancsak jelen van a már említett 25-ös és 100-as mintaszámnál tapasztalható visszaesés. Ezzel ellentétben a SimCLR esetében most is az látható, hogy sikerül megtartani a javulás ütemét, ami az AUC értékek esetén is széles tartománnyal társul a problémás mintaszámoknál.

A fentiek alapján elmondható tehát, hogy a szomszédsági számon alapuló módszer sikeresen javított a naiv eljáráshoz képest a kisebb mintaszámok esetében. A javulás a SimCLR hálózat látens terénél volt stabilabb, az autoenkódernél a medián értékek egyes pontokban visszaestek a véletlen választás szintjére.

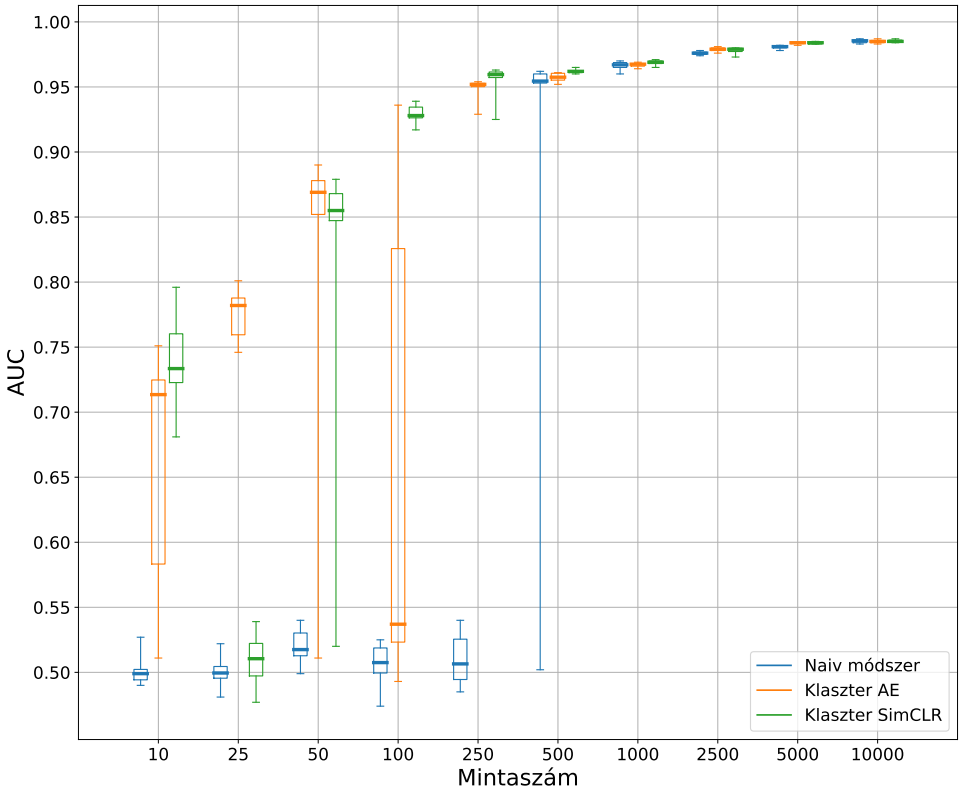
6.3.2. Klaszterezésen alapuló módszer értékelése

A fentebbi 2 módszer után utolsóként tekintsük a felhasználói beavatkozással kiegészített klaszterezésen alapuló mintakiválasztási módszer eredményeit, ugyancsak külön az autoenkóder és a SimCLR hálózat látens tere esetében. A naiv módszerrel való összehasonlításhoz a 6.10 ábra nyújt segítséget, ahol a két módszer pontosság értékei szerepelnek.



6.10. ábra. A klaszterezésen alapuló módszer pontosság eredményei a mintaszámok függvényében a Fashion-MNIST adathalmazon a naiv módszerrel összehasonlítva

A klaszterező módszerek pontosság értékeit megfigyelve ugyancsak elmondható az autoenkóder látens tere alapján képzett adathalmazok esetén a szomszédsági számon alapuló módszereknél tapasztaltak. Itt is először javulás látható a naiv módszerhez képest, azonban 100-as mintaszám esetében megint visszaesik a pontosság a 10%-os szintre és csak ezután kezd el újra javulni. A SimCLR esetében is hasonló jelenség figyelhető meg, azonban itt más mintaszámnál: itt az 25-es méretnél következik be egy leromlás, míg a többi kis mintaszám esetén hasonlóan teljesít az autoenkóderrel, sőt több esetben felül is múlja azt. Ezeket kiegészítve tekintsük a 6.11 ábrát, ahol az AUC értékek összehasonlítása látható.



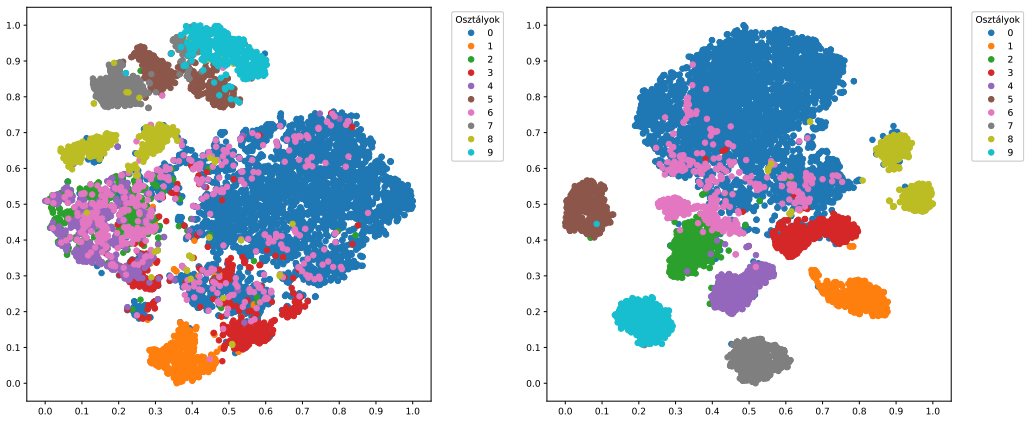
6.11. ábra. A klaszterezésen alapuló módszer AUC értékei a mintaszámok függvényében a Fashion-MNIST adathalmazon a naiv módszerrel összehasonlítva

Itt megint csak egyezés van a pontosság értékeknél látott jelenségekkel, miszerint az autoenkóder esetében a 100-as mintaszámnál, míg a SimCLR hálózatnál a 25-ösnél tapasztalható jelentős romlás, a többi kis mintaszámnál látható a javulás folyamata.

A fentiek alapján összegezve a tapasztaltakat az mondható el, hogy mind az autoenkóder, mind a SimCLR látens terén alapuló klaszterezési módszer képes volt a kisebb mintaszám esetében számottevő javulás elérésére a szomszédsági számon alapuló mintakiválasztási módszerhez képest. Mindkettő esetben azonban megjelentek visszaesési pontok más-más helyen, ahol a módszer lényegében a naiv módszer szintjével egyezett, így bár a SimCLR több esetben valamivel magasabb értékeket adott, mint az autoenkóder, illetve kevesebb helyen romlott le, mégsem lehet egyértelműen az összes vizsgált mintaszámra megadni egy legjobban teljesítő módszert.

6.3.3. Az eredmények viszonya a látens terekhez

Az észrevételek összefoglalása után tekintsük a 6.12 ábrát, ahol a kettő hálózat látens terének t-SNE ábrája látható egymás mellett.



(a) Autoenkóder hálózat **(b)** SimCLR hálózat

6.12. ábra. A Fashion-MNIST adathalmazon tanított hálózatok t-SNE ábrái

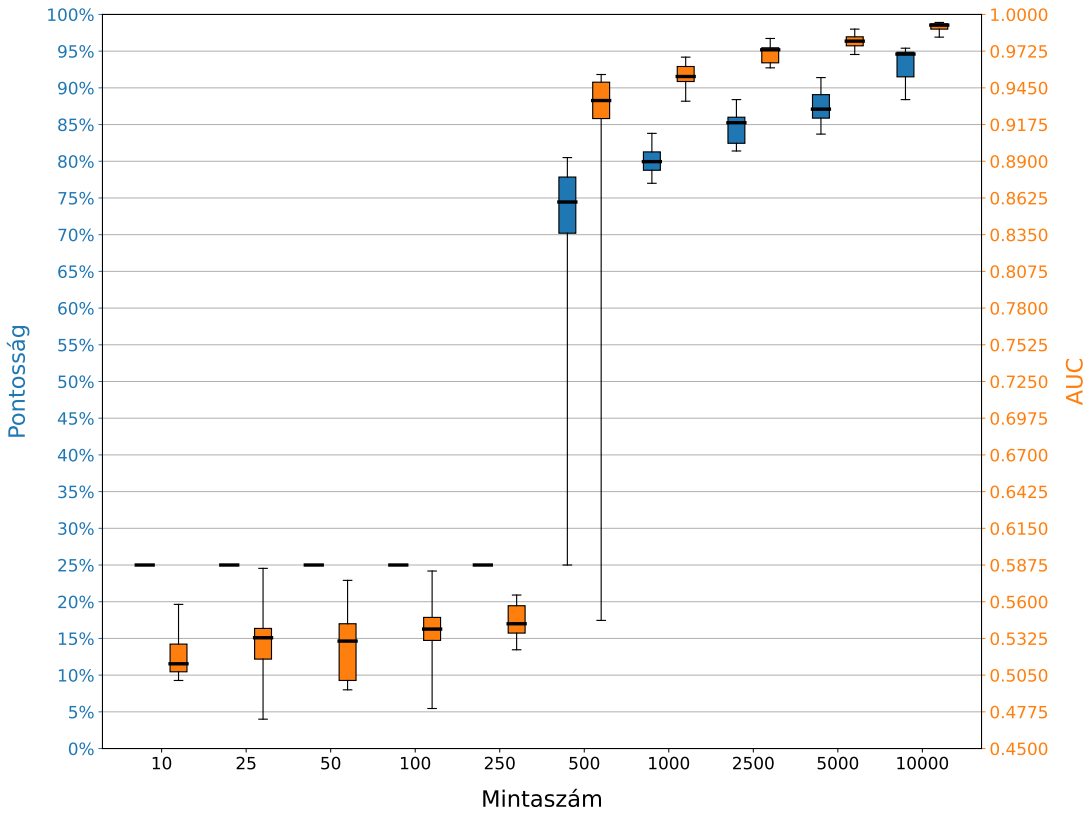
A fenti ábrák alapján rögtön egyértelművé válnak a kisméretű adathalmazoknál tapasztalt problémák: a kapott látens tér egyes klaszterei teljesen egymásba lógnak, ezzel ellehetetlenítve a pontos kiválasztást. Ez a jelenség az autoenkódernél jelentkezik igazán, míg a SimCLR látens terének esetében már sokkal kevesebb mintát érint az egyes csoportokból. Ezek okozzák tehát a fentebb látott visszaeséseket, hiszen egy kicsit is megnövelve a vizsgált mintaszámot, könnyen beleválaszthatunk a problémás klaszterhatárok területére, így elrontva az amúgy egyenletesnek gondolt eloszlást. Emellett magyarázatot ad a SimCLR valamivel jobb teljesítményére is, hiszen ott ez a hátrányos klaszterhelyzet nem annyira szembetűnő, mint az autoenkóder esetében. Összességében elmondható tehát, hogy a Fashion-MNIST sokkal nehezebb adathalmaznak bizonyult (ez már a 85%-os maximális pontosságból is látszik), mint az előző, így a látens téren alapuló módszerek ugyan javítottak a naiv módszeren, mégis az általuk adott eredmények felemásak lettek. Az architektúrák közül azért valamivel kiemelkedik a SimCLR, a jobb minőségű látens terével, így az összetettebb adathalmazok esetén az arra épülő módszerek használata javasolt.

6.4. A forrasztási képek adathalmazon végzett kísérletek

Ebben a részfejezetben, a vizsgált adathalmazok közül utolsóként, a forrasztási képek adathalmazán végzett kísérletek eredményei kerülnek bemutatásra. Mivel a 3 halmaz közül ez az egy valódi ipari adathalmaz, így az itt levont következtetések vélhetően jól jellemzik az egyes módszerek képességeit más ipari alkalmazásokban is. Maguk a forrasztási képek ugyan nagyobb méretűek, mint a számjegyek vagy ruhadarabok mintái, mégis az azonos beállítás és elrendezés miatt a képek nagy része azonos. Emellett a lényeges részek pedig a minőségi osztályt határozzák meg, így elegendő egy konvolúciós blokkal kiegészíteni az osztályozó hálózatot és az enkóder struktúrát.

6.4.1. A naiv módszer eredményei

A különböző módszerek közül most is elsőként a naiv mintakiválasztási eljárást vizsgáljuk először, hiszen az itt megfigyelt jelenségekre, esetleges problémákra kell a látens téren alapuló módszereknek megoldást adniuk. A kapott értékek a 6.13 ábrán láthatóak.



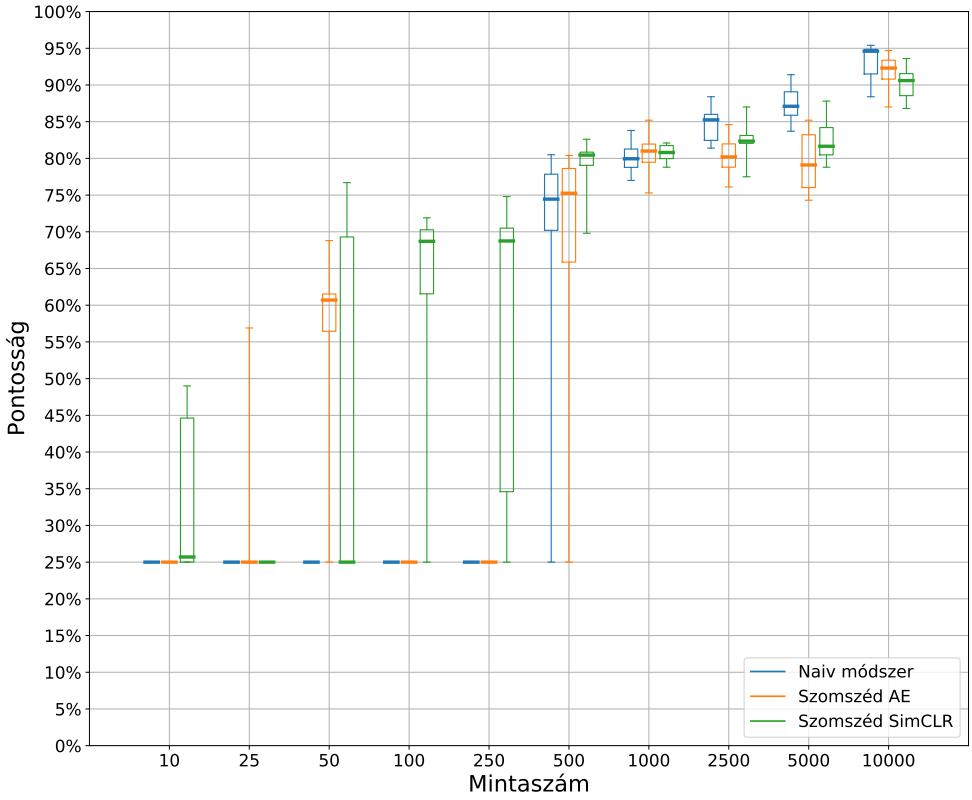
6.13. ábra. A naiv módszer pontosság és AUC eredményei a mintaszámok függvényében a forrasztási képek adathalmazán

Rögtön szembetűnő a pontosság értékek alakulását tekintve, hogy itt is csak 500 minta után kezdenek el igazán tanulni a hálózatok, azelőtt lényegében a véletlen kiválasztás szintjén, 25%-on maradnak. Ez az érték azzal indokolható, hogy mivel ebben az adathalmazban csak 4 osztály van, így amikor a legnagyobb eloszlású kategória (a megfelelő forrasztási minták) dominálják a részhalmazt, akkor a hálózat mindent abba az osztályba sorol, hiszen számára ez jelenti a legkisebb költséget, tehát csak az esetek 1/4-ében dönt jól. Az 500-as méretet elérve és meghaladva azonban elkezd a pontosság az elvárható mértékben javulni, mígnem eléri a jónak tekinthető 95%-os értéket. Ugyanez a tendencia figyelhető meg az AUC értékek esetében is: az 500 alatti mintaszámok esetén a hálózat kicsivel van a véletlen kiválasztáshoz tartozó 0.5-ös szint felett. Ezen a határon túl a mintaszámot növelve azonban a pontossághoz hasonlóan ugyancsak hirtelen megugrik a teljesítmény és a jónak számító $[0.95, 1]$ tartományba kerülünk bele a nagyobb méretek esetén.

A fenti tapasztalatok alapján tehát a javulás igazán a kis mintaszámok esetében érhető el ezen adathalmaz esetén is, amelyet a látens téren alapuló módszerek segítségével lehetünk képesek elérni.

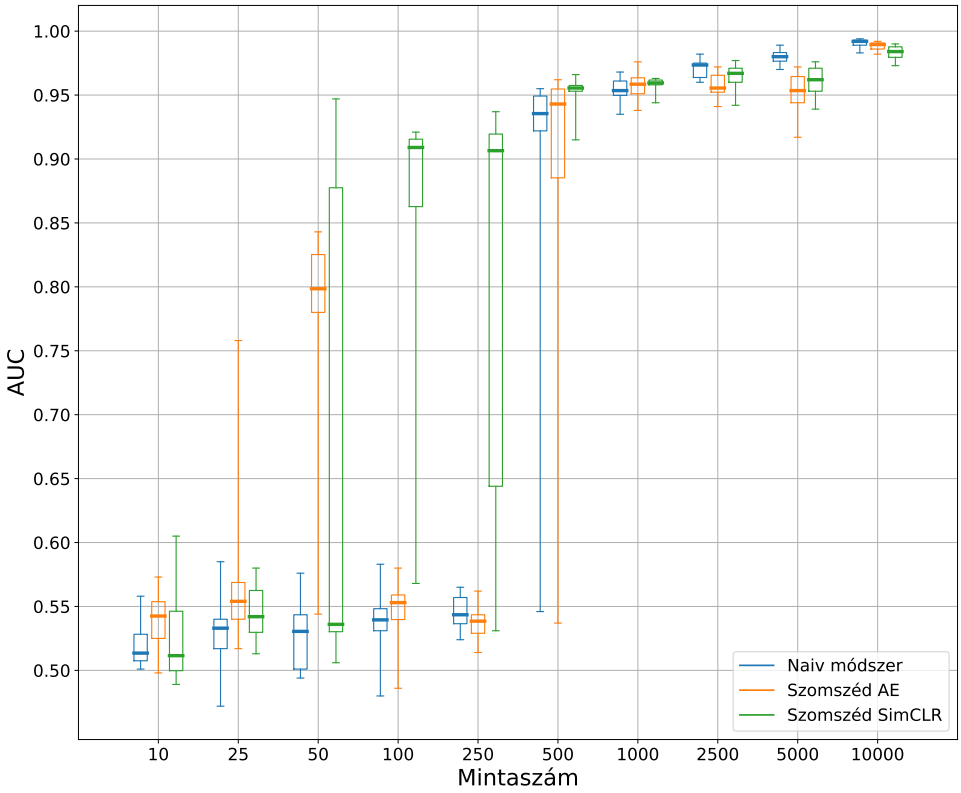
6.4.2. Szomszédsági számon alapuló módszer eredményei

A naiv módszer eredményeinek taglalása után következhet a látens téren alapuló módszerek esetén kapott értékek elemzése, azok közül is elsőként a szomszédsági szám alapján történő mintakiválasztási módszer eredményei. Itt is, mint az előző részekben, külön esetként kezeljük az autoenkóder és a SimCLR hálózat által előállított látens tereken történő kiválasztást, így összehasonlítva azok minőségét. A vizsgált metrikák közül először tekintsük a kapott pontosság értékeket, amelyek a 6.14 ábrán láthatóak.



6.14. ábra. A szomszédságon alapuló módszer pontosság eredményei a mintaszámok függvényében a forrasztási képek adathalmaza esetén a naiv módszerrel összehasonlítva

Az ábra alapján látható, hogy sikerült a kisebb mintaszámok esetében a naiv módszert néhány esetben felülmúlni, azonban a különböző hálózatok egyes méretek esetén visszaestek a várt javulás folyamata közben. Ilyen visszaesés tapasztalható az autoenkóder esetében a 10-es, 100-as és a 250-es mintaszámoknál, míg a SimCLR hálózat javuló görbéje kevesebb pontban (25-ös és 50-es méret esetén) szenvedett törést. Ezeket egészíti az AUC értékek alakulása, amely a 6.15 ábrán követhető nyomon.



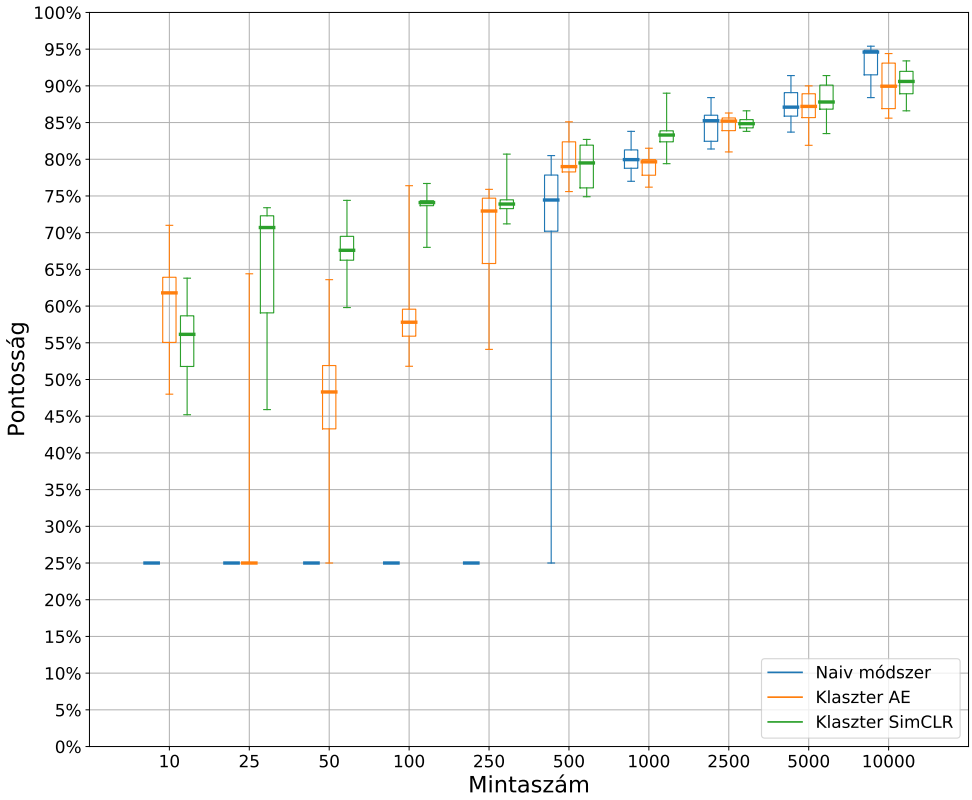
6.15. ábra. A szomszédságon alapuló módszer AUC értékei a mintaszámok függvényében a forrasztási képek adathalmaza esetén a naiv módszerrel összehasonlítva

Az AUC értékek tekintetében itt már folyamatosabbnak tekinthető a változás menete: az autoenkóder esetében az 50-es mintaszámnál tapasztalható egy kiugrás, azonban a többi kis méretnél alig lépi túl a 0.5-ös szintet, míg a SimCLR modell már az 50-es mérettől kezdve képes akár 0.9 fölötti AUC érték elérésére is.

A fentebb látott jelenségek alapján elmondható tehát, hogy ugyan mindkét hálózat látens tere alapján végzett szomszédság alapú mintakiválasztás valamelyest javított a naiv módszer eredményein, mégis inkább a SimCLR hálózat esetén volt látható igazán jelentős, stabilabb javulási folyamat, míg az autoenkóder esetében inkább csak egy-egy kiugrás volt tapasztalható.

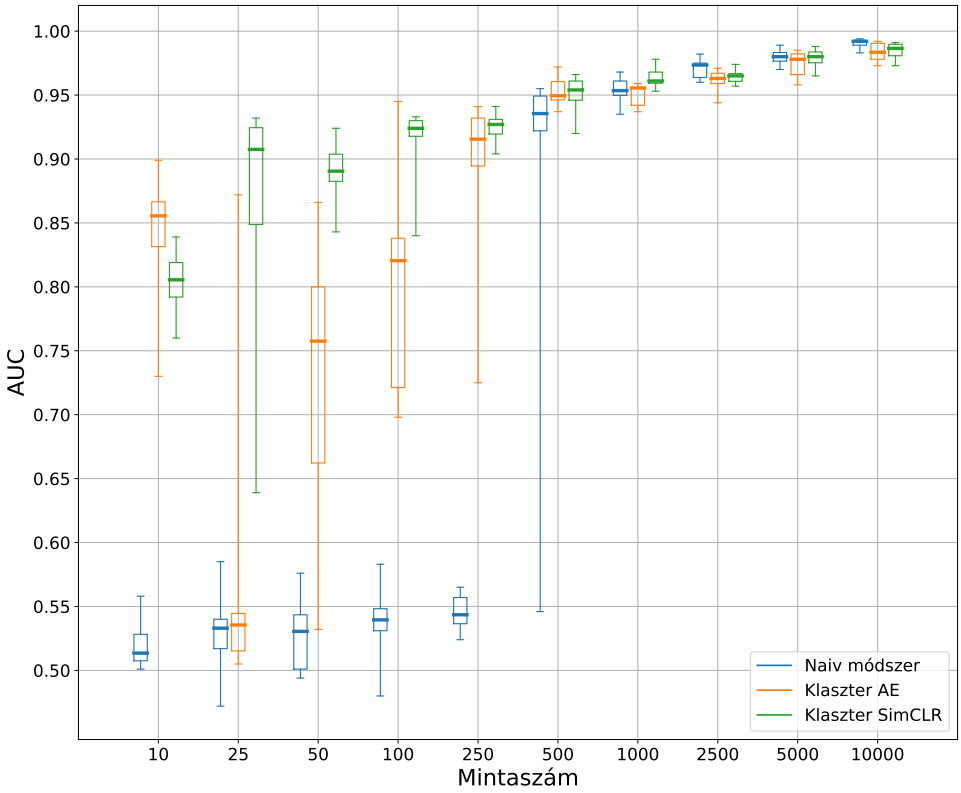
6.4.3. Klaszterezésen alapuló módszer értékelése

A látens téren alapuló módszerek közül utolsóként vizsgáljuk meg a felhasználói beavatkozással kiegészített klaszterezésen alapuló mintakiválasztási módszert, mellyel a feladat az előzőekben tapasztalt visszaesések kiküszöbölése. Ennek sikerességét a 6.16 ábra alapján lehet eldönteni, amelyen a pontosság értékek változása követhető nyomon.



6.16. ábra. A klaszterezésen alapuló módszer pontosság eredményei a mintaszámok függvényében a forrasztási képek adathalmaza esetén a naiv módszerrel összehasonlítva

Először az autonekóder esetében kapott értékeket vizsgálva az látható, hogy ugyan 25-ös mintaszámnál még mindig van egy letörés, mégis a többi kis mintaszám esetén sikerült a 25%-os szintet jóval felülmúlni. A SimCLR látens terének esetében még jobb a javulás az előző eredményekhez képest: itt egyszer sem tapasztalható visszaesés a pontosság értékek alakulásában, illetve kisebb tartománnyal is rendelkeznek, mint az autoenkóder esetében. Az előbb levont következtetéseket hivatott kiegészíteni, illetve megerősíteni a 6.17 ábra, ahol az AUC értékek kerülnek bemutatásra.



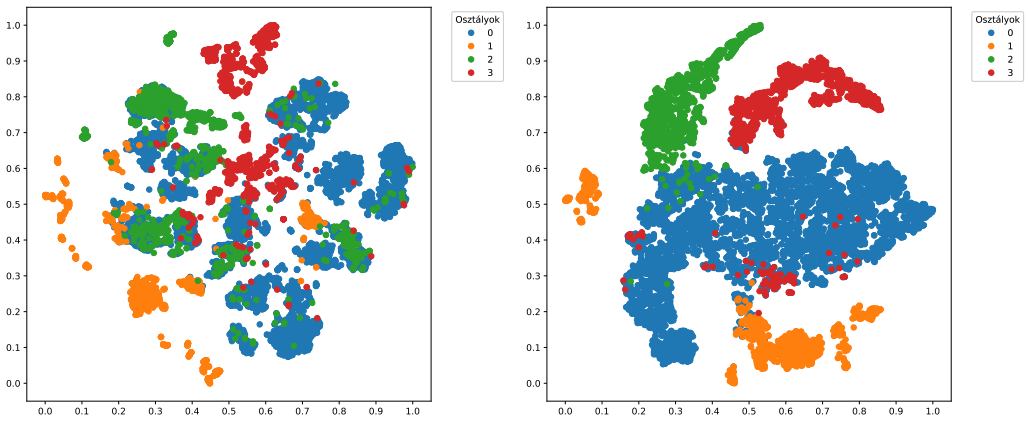
6.17. ábra. A klaszterezésen alapuló módszer AUC értékei a mintaszámok függvényében a forrasztási képek adathalmaza esetén a naiv módszerrel összehasonlítva

Az AUC értékek elemzése során hasonló jelenséget láthatunk, mint a pontosság értékek esetében: az autoenkóder magas értékről indul, majd pedig a 25-ös mintaszámnál visszaesik a naiv módszer szintjére és onnan kezd el csak javulni a halmaz méretének növelésével. Ezzel szemben a SimCLR látens terén alapuló kiválasztás végig 0.8-as AUC érték felett marad és már a 25-ös méretnél eléri a 0.9-es szintet, amit meg is tud tartani a mintaszám növelése során.

Összegezve a tapasztaltakat az állítható, hogy az autoenkóder ugyan a 10-es méretnél kifejezetten magas metrika értékekről indul, mégis a többi kis mintaszám esetén először visszaesik a naiv módszer szintjére és onnan kezd el csak javulni. Ezzel egyidejűleg habár a SimCLR kisebb kezdeti értékekről indul, mégis sokkal stabilabban képes tartani a javulás ütemét, így a 10-es méretet követően egészen a 250-es mintaszámig kimondottan jobb teljesítményt tud nyújtani.

6.4.4. Az eredmények viszonya a látens terekhez

A tapasztalatok összegzése után tekintsük a 6.18 ábrát, ahol a kettő hálózat látens terének t-SNE ábrája látható egymás mellett.



(a) Autoenkóder hálózat **(b)** SimCLR hálózat

6.18. ábra. A forrasztási képek adathalmazán tanított hálózatok t-SNE ábrái

Az autoenkóder látesn terének elemzésével kezdve rögtön az látható, hogy szinte semmilyen szerkezetet nem lehet találni a pontfelhők elhelyezkedésében: a jó mintákhoz egyszerűen sok helyen hozzákapcsolódnak a hibás kategóriák adatpontjai, így -főleg az ábra közepén- nagy zavart okozva. Ezzel szemben a SimCLR hálózatnak sikerült megtanulnia az egyes minták főbb tulajdonságait és ezáltal bár néhol összeérő, mégis jól elkülöníthető klasztereket kaptunk. Ezek fényében már érthető, hogy az átlapolódó csoportok miatt voltak gyakori visszaesések az autoenkóder látens terét használó módszerek esetében, míg a SimCLR esetében a sokkal jobb minőségű látens térből adódóan stabilabb javulási görbét sikerült kapnunk. Ezen felül a két ábra összehasonlítása rávilágít arra is, hogy a valóságos ipari adathalmazok esetén, ahol már sokszor kifejezetten nehéz a mintákat osztályozni, a SimCLR hálózat által használt módszereket érdemes felhasználni. A két mintakiválasztó módszer közül a klaszterezésen alapuló algoritmus bizonyult a hatékonyabbnak, mivel a látens tér (még ha az rossz minőségű is) klaszterezését a felhasználó közvetlenül befolyásolhatja, így sokszor jobb eredményt elérve a szomszédságon alapuló eljárásnál.

7. fejezet

Összefoglalás

A dolgozat során bemutatásra került 2, a látens téren alapuló mintakiválasztási módszer, amelyek eltérő megközelítéssel ugyan, de azonosan az annotáláshoz szükséges időt voltak hivatottak csökkenteni a hálózat teljesítményének minél jobb megőrzése mellett. A látens teret 2 hálózatarchitektúra alapján állítottam elő, melyek az autoenkóder és a SimCLR struktúra voltak. Az így kapott összeállításokat 3, különböző felépítésű, így eltérő nehézségű adathalmazon értékeltem ki: a számjegyeket tartalmazó MNIST adathalmazon, az ehhez hasonló, ruhákat tartalmazó Fashion-MNIST adathalmazon és a forrasztási képek ipari adathalmazán. A kapott eredményeket a pontosság és az AUC metrika segítségével értelmeztem, elemeztem, azok értékeit grafikusan is szemléltetve. Ezekből a lehetőségekből összesen 15 kísérleti összeállítást készítettem (a naiv módszert is beleszámítva) és minden így előálló hálózatot 10 alkalommal tanítottam a véletlen inicializálás hatásának kizárása érdekében 10 különböző méretű részhalmazon. Összesítve tehát a dolgozatban közölt eredmények legalább 1500 különböző hálózat tanítása alapján adódtak, így valóban egy átfogó képet adva a módszerek teljesítményéről.

A kísérleti összeállításban az egyenlőtlen eloszlású adathalmazokat bontottam szét különböző méretű részadathalmazokra, majd az ezeken tanított hálózatokat a már egyenlő osztályeloszlású validációs halmazon értékeltem ki. Az adott adathalmazhoz tartozó látens tereket az előállító hálózattól függően szétválasztottam, majd ezeket a mintakiválasztó módszerek bemenetére adtam. A hálózatok validációja során a pontosság és az AUC metrika értékeket vizsgáltam, ezek mediánját és tartományát, illetve szórását grafikusan ábrázoltam. A kapott értékek magyarázatához felhasználtam az egyes hálózatok által előállított látens terek szerkezeti jellemzőit, azon belül is az egyes klaszterek elhelyezkedését.

A kapott eredmények alapján elmondható, hogy mindkét szempontból sikeresnek bizonyultak a bemutatott eljárások, hiszen mindhárom adathalmaz esetén sikerült nagy mértékben javítani a naiv módszer által jelentett alapszintű teljesítményen és mélyebb rálátást nyertünk ezáltal a vizsgált adathalmazok struktúrájára. Ezen felül a 2 különböző hálózattípusnak köszönhetően egy-egy módszert többféle látens téren is ki lehetett próbálni és a kapott eredmények alapján a legjobbat kiválasztani.

Ez a kettőség, miszerint szabadon párosíthatóak a módszerek a különböző látens terekkel, az egyedi szerkezetű és emiatt eltérő nehézségű adathalmazok esetében mindig más-más optimális kombinációt eredményezett, ezáltal minden eshetőségre egy lehetséges megoldást adva. Természetesen az itt közölt eredményeken és metrika értékeken az osztályozó és a látens tereket előállító hálózatok módosításával, esetleg más architektúra választásával lehet javítani, a folyamatot optimalizálni. Például mind a szomszédsági számon, mind a klaszterezésen alapuló módszernél a minták kiválasztásához alkalmazott eljárás nem a lehető legoptimálisabb megközelítés, így azt javítva, tökéletesítve meg lehet növelni a módszerek hatékonyságát.

Ezzel együtt a kapott értékek alapján levont tapasztalatokból kiindulva elmondható, hogy sikerült a már meglévő módszerek mellé egy lehetséges alternatívát biztosítani az ipari adathalmazok hatékony annotálásra, amely számos körülmény és tényező esetén is hatékony és megbízható eredményt képes nyújtani. A bemutatott módszerek egyik nagy előnye, hogy lényegében az adathalmaz szerkezetétől (osztályok száma, minták mérete) függően egységesen paraméterezhetőek, nem szükséges magának az eljárás lépéseinek a változtatása.

Érdemes végül megjegyezni, hogy az előzőek mellett számos lehetőség van a továbbfejlesztésre és a kiegészítésre is, hiszen a bemutatott algoritmusok még tovább javíthatóak egy-egy adott problémához igazodva, illetve megadható más olyan eljárás, amely a látens tereken alapuló mintakiválasztást valósítja meg. Jó lehetőség a fejlesztésre például a szomszédsági szám meghatározásának árnyalása az egyes pontok közötti kapcsolat nyilvántartásával (szomszédsági gráf) vagy a látens tér klaszterezésének másfajta csoportosító módszerrel történő megvalósítása.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani Karz Gergelynek (Artillence Kft.) a szakmai segítségért, útmutatásért és a számítógépes erőforrások biztosításáért.

Irodalomjegyzék

[1] Raffaele Cioffi – Marta Travaglioni – Giuseppina Piscitelli – Antonella Petrillo – Fabio De Felice: Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability*, 12. évf. (2020) 2. sz. ISSN 2071-1050. URL https://www.mdpi.com/2071-1050/12/2/492.

[2] Xuebing Yang – Qiuming Kuang – Wensheng Zhang – Guoping Zhang: Amdo: An oversampling technique for multi-class imbalanced problems. *IEEE Transactions on Knowledge and Data Engineering*, 30. évf. (2018) 9. sz., 1672–1685. p.

[3] Zahra Shamsi – Kevin J. Cheng – Diwakar Shukla: Reinforcement learning based adaptive sampling: Reaping rewards by exploring protein conformational landscapes. *The Journal of Physical Chemistry B*, 122. évf. (2018) 35. sz., 8386–8395. p.
URL https://doi.org/10.1021/acs.jpcb.8b06521. PMID: 30126271.

[4] Davoud Davoudi Moghaddam – Omid Rahmati – Mahdi Panahi – John Tiefenbacher – Hamid Darabi – Ali Haghizadeh – Ali Torabi Haghighi – Omid Asadi Nalivan – Dieu Tien Bui: The effect of sample size on different machine learning models for groundwater potential mapping in mountain bedrock aquifers. *CATENA*, 187. évf. (2020), 104421. p. ISSN 0341-8162. URL https://www.sciencedirect.com/science/article/pii/S0341816219305636.

[5] Ting Chen – Simon Kornblith – Mohammad Norouzi – Geoffrey E. Hinton: A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709. évf. (2020). URL https://arxiv.org/abs/2002.05709.

[6] F. Rosenblatt: The perceptron - a perceiving and recognizing automaton. 85-460-1. Jelentés, Ithaca, New York, 1957. January, Cornell Aeronautical Laboratory.

[7] Olaf Ronneberger – Philipp Fischer – Thomas Brox: U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597. évf. (2015).
URL http://arxiv.org/abs/1505.04597.

[8] L.J.P. van der Maaten – G.E. Hinton: Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9. évf. (2008), 2579–2605. p.

[9] Martin Wattenberg – Fernanda Viégas – Ian Johnson: How to use t-sne effectively. *Distill*, 2016. URL http://distill.pub/2016/misread-tsne.

[10] Mayra Z. Rodriguez – Cesar H. Comin – Dalcimar Casanova – Odemir M. Bruno – Diego R. Amancio – Luciano da F. Costa – Francisco A. Rodrigues: Clustering algorithms: A comparative approach. *PLOS ONE*, 14. évf. (2019. 01) 1. sz., 1–34. p.
URL https://doi.org/10.1371/journal.pone.0210236.

[11] Y. LeCun – L. Bottou – Y. Bengio – P. Haffner: Gradient-based learning applied to document recognition. In *Intelligent Signal Processing* (konferenciaanyag). 2001, IEEE Press, 306–351. p.

[12] Han Xiao – Kashif Rasul – Roland Vollgraf: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747. évf. (2017). URL http://arxiv.org/abs/1708.07747.

[13] Guy S. Handelman – Hong Kuan Kok – Ronil V. Chandra – Amir H. Razavi – Shiwei Huang – Mark Brooks – Michael J. Lee – Hamed Asadi: Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods. *American Journal of Roentgenology*, 212. évf. (2019) 1. sz., 38–43. p.
URL https://doi.org/10.2214/AJR.18.20224. PMID: 30332290.

[14] Hieu Pham – Qizhe Xie – Zihang Dai – Quoc V. Le: Meta pseudo labels. *CoRR*, abs/2003.10580. évf. (2020). URL https://arxiv.org/abs/2003.10580.

[15] Tom Fawcett: An introduction to roc analysis. *Pattern Recognition Letters*, 27. évf. (2006) 8. sz., 861–874. p. ISSN 0167-8655. URL https://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition.

[16] Siu Kwan Lam – Antoine Pitrou – Stanley Seibert: Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15 konferenciasorozat. New York, NY, USA, 2015, Association for Computing Machinery. ISBN 9781450340052.
URL https://doi.org/10.1145/2833157.2833162. 6 p.

[17] David M. Chan – Roshan Rao – Forrest Huang – John F. Canny: t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data. *CoRR*, abs/1807.11824. évf. (2018). URL http://arxiv.org/abs/1807.11824.

[18] Tian Zhang – Raghu Ramakrishnan – Miron Livny: Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96 konferenciasorozat. New York, NY, USA, 1996, Association for Computing Machinery, 103–114. p. ISBN 0897917944. URL https://doi.org/10.1145/233269.233324. 12 p.

[19] Boris Lorbeer – Ana Kosareva – Bersant Deva – Dženan Softić – Peter Ruppel – Axel Küpper: Variations on the clustering algorithm birch. *Big Data Research*, 11. évf. (2018), 44–53. p. ISSN 2214-5796. URL https://www.sciencedirect.com/science/article/pii/S2214579617300151. Selected papers from the 2nd INNS Conference on Big Data: Big Data Neural Networks.