Debreceni Egyetem
Informatikai Kar
Adattudomány és Vizualizáció Tanszék

DIPLOMAMUNKA

Tanítási adathalmaz dúsítása generatív modellek használatával

Témavezető:
Dr. Bogacsovics Gergő
Adjunktus

Készítette:
Nyika Benedek
PTI MSc

Debrecen
2025

Tartalomjegyzék

1. Bevezetés

A statisztika egy olyan tudományág, amely a valóság számszerű információinak megfigyelésére, összegzésére, elemzésére és modellezésére irányul. Célja, hogy a világunkat alkotó objektumok tulajdonságait pontos és egyértelmű definiálással leírja és számszerűsítse.

A statisztika lehetőséget nyújt a leírt adatokon különböző elemzéseket végezni, ami további nem egyértelmű tulajdonságokat tud feltárni. Például, ha van két oktatási intézmény, aminek a teljesítményét össze akarjuk hasonlítani, akkor vehetjük az intézmények diákjai által elvégzett kompetencia tesztjeit, amelyeket kiértékelve és átlagolva össze is tudjuk hasonlítani, hogy melyik intézmény teljesít jobban az oktatás kérdésében. Így számszerűsítve egy absztrakt kérdésre a választ, mint “Melyik intézmény az eredményesebb?”.

A gépi tanulás a statisztika tudományából vált ki, azzal a céllal, hogy adatokat elemezve egyre komplexebb és absztraktabb mintákat lehessen feltárni a világunkról, kezdve az egyszerű, kvantitatív változók közötti összefüggésektől, egészen a többdimenziós, nem-lineáris rendszerek rejtett kapcsolatának feltárásáig. Ilyen összefüggések lehetnek például egy Descartes-féle koordináta-rendszerben felvett pontokra való egyenesnek az állítása, hogy meggyőződhessünk arról, milyen kapcsolat van az X és az Y tengely értékei között. Esetleg ugyanezt az egyenest felhasználva megmondhatjuk, hogy kiválasztva egy tetszőleges értéket az X tengelyen, melyik érték lenne a legvalószínűbb, ami hozzá tartozna az Y tengelyen, így létrehozva egy kezdetleges prediktáló módszert. Ugyanezt az egyenest felhasználhatjuk arra is, hogy besoroljunk bizonyos pontokat csoportokba, például, ha a vonal alatt van, akkor az egyik csoportba soroljuk, ha a vonal felett van a pont, akkor a másikba. Ezzel megalkotva egy kezdetleges osztályozó algoritmust.

Azonban ezek mind csak lineáris problémák voltak, melyet meg lehetett oldani egy egyenes illesztésével. Ennek ellenére vannak módszerek, amelyek sokkal kifinomultabbak és összetettebb megoldást tudnak a problémákra adni. Ilyen például, ha egy képet mátrixként olvasunk be, és a pixelek elhelyezkedéséből (a kép pontjaiból) meg lehet mondani, hogy milyen kontúrok és alap geometriai alakzatok lelhetőek fel a képen a Hough transzformáció[36] felhasználásával, vagy azon is túl használhatunk még kifinomultabb eszközöket, mint például gépi tanulási algoritmusokat, azon belül a szegmentálást, amely már nem csak behatárolja a körvonalát egy adott objektumnak, hanem még fel is címkézi a kijelölt terület tartalma alapján, hogy milyen objektum található meg benne.

A generatív modellek olyan gépi tanulási algoritmusok, amelyek képesek új, még nem látott adatpontokat előállítani a tanítási adatbázisa alapján. Ilyen modellek például a Generative Adversarial Network (GAN), a Variational Autoencoder (VAE), Generative Pre-trained Transformer (GPT). Ezen modellek meglepően élethű eredményeket tudnak létrehozni, melyek akár hitelesnek is tűnhetnek anélkül, hogy részletesen analizálnánk az adott kimenetüket (ilyen modellek például: DALL-E[37, 38], ChatGPT[39], DeepSeek[40]). Mindegyik modell létezése arról tesz tanúbizonyságot, hogy minél nagyobb számítási kapacitás és jó minőségű adathalmaz áll rendelkezésünkre, akkor egyre jobb minőségű kimenetet tudunk létrehozni a modellekkel.

A generatív modellek működési elve, hogy a komplex és absztrakt összefüggéseket feltárva új, még nem látott kimenetet tudnak létrehozni. Az ilyen módszerek fejlődése mellett fontos, hogy megértsük, milyen feltételek szükségesek ahhoz, hogy egy adott modell képes legyen reális és hihető kimenetet generálni. A generatív modellek elterjedésével és népszerűségének növekedésével érdemes figyelmet fordítani rá, hogy teljesen ki lehessen használni a benne rejlő potenciált.

Azonban felmerülhet bennünk a kérdés, hogy mire lehet a gépi tanulási algoritmusokat felhasználni, ha esetleg olyan területen alkalmaznánk, ahol nem áll rendelkezésre elégséges adat? Erre vannak különböző technikák, amelyek csökkenthetik a szegényes adathalmazból eredő hátrányokat, de ezek tárháza is véges, így érdemes újabb megoldásokat keresni.

Az eddigi kutatások alapján [1] [2] már jól ismert tény, hogy a generatív modellek használata egy tanítási halmazon egy valós adat augmentációs stratégia, mely tovább javítja az új dúsított adathalmazon tanult modell általánosító képességét. A dolgozatom célja, hogy ezen adatbővítési technika határait vizsgálja, vagyis mennyire kis adathalmazon lehet még effektíven alkalmazni hatékonyan.

2. Elméleti háttér

A kutatáshoz először meg kell alapozni, hogy milyen eszközök kellenek a tesztek lebonyolításához. Ezen felül meg kell határozni, hogy milyen egyéb információkra van szükség ahhoz, hogy az eredményeket értelmezni is tudjuk. Ebben a fejezetben ezekre az említett részletekre térek ki.

Gépi tanulási algoritmusok

A gépi tanulási algoritmusok olyan fejlett statisztikai algoritmusok, melyek képesek nem triviális összefüggéseket is feltárni egy adathalmazban. Ilyen összefüggés lehet például egy szöveg érzelmi töltetének a felismerése [3] vagy egy kép pixeleit beolvasva felismerni és szövegesen leírni, hogy az mit tartalmaz [4]. A modell teljesítménye azonban függ az adatok minőségétől, mennyiségétől és a rajta tanuló modell komplexitásától, valamint kapacitásától.

Generatív modellek

A Generatív modellek olyan gépi tanulási algoritmusok, melyek a tanítási folyamat alatt megtanulják az adatbázisban fellelhető statisztikai jellegzetességeket, legyen az nyelvtani, vagy akár képi. Ezt követően ezek felhasználásával új, még nem látott kimenetet tudnak létrehozni, amelyek ugyanazzal a tulajdonságokkal rendelkeznek, mint az eredeti adatbázis elemei.

Különböző típusai vannak, mint például az autoregresszív modellek, amelyek a kimeneti eloszlást sorozatosan, egymásra építve modellezik, így minden egyes kimeneti token, mely lehet szó vagy karakter, feltételes valószínűségét a korábbi tokenek alapján határozzák meg. Erre egy jó példa a GPT - Generative Pre-trained Transformer [33].

Egy másik példa generatív modellre az ún. Variációs Autoencoderek (VAE)[34], mely modellek a szöveget egy látens térre vetítik le, és abból a rejtett térről próbál szöveget generálni. Ezenfelül léteznek diffúziós (diffusion) modellek [35] is. Ezek azt az alapötletet veszik alapul, hogy lényegében bármilyen képet ki lehet élesíteni (még a teljesen random zajt is), ha megadjuk neki mit kell tartalmaznia a kimenetnek.

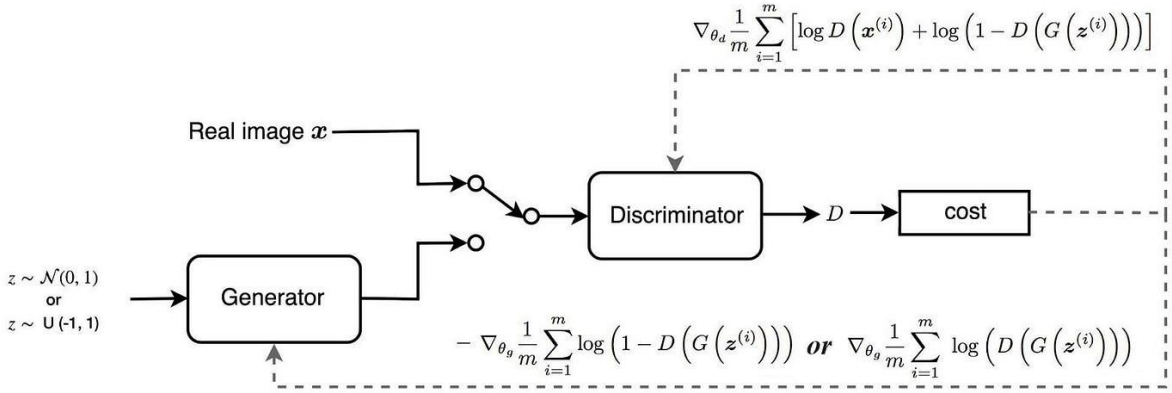
Generative Adversarial Network (GAN)

A Generative Adversarial Network (GAN) a generatív modellek közül talán az egyik legnagyobb potenciállal rendelkezik, hogy élethű és pontos képeket tudjon generálni. Azonban

az Adversarial/Versengő mivolta miatt kifejezetten nehéznek bizonyul a tanítása, ellentétben más, egyszerűbb modellekkel, mint a diffúziós modellek [11].

Mint ahogy a neve is árulkodik róla, a GAN két egymással versengő modellből épül fel, a Generátorból (Generator), illetve a Diszkriminátorból (Discriminator). Ezek szerepei a következők (lásd 1. ábra).

- Diszkriminátor: ez a komponens próbálja kiszűrni a hamis képeket az igaziak közül.
- Generátor: képeket generál és próbál hamis képeket csempészni az igaziak közé anélkül, hogy azt a Diszkriminátor észlelné.



1. ábra GAN tanításának reprezentálása [5]

Működése

A GAN tanítási lépései a következők (lásd: 1. ábra):

1. A Generátor feldolgoz egy vektort (látens vektor), amit véletlen értékekkel töltünk fel. Ez a vektor absztrakt módon reprezentálja, hogy mit is tartalmaz a kép. Például egy kutya oldalról máshogy néz ki, mint szemből, így teljesen más kontúrokkal kell kiidulni.
2. A Generátor a látens vektort felskálázza valamilyen módszer alapján (Conv2DTranspose [21], Upsample [22]), amely fel tudja skálázni a bemeneti mátrix méretét, melyet ha többször használunk egymás után, akkor a végén ugyanazzal a felbontással fog rendelkezni, mint az eredeti kép.
3. A Diszkriminátor keverve olvassa be az eredeti és a hamisított képeket azzal a céllal, hogy kiválassza, szerinte melyik az igazi és melyik a hamisított kép.
4. A Diszkriminátor eredménye alapján számolunk loss értéket.
5. A loss érték alapján optimalizáljuk a Diszkriminátort és a negáltjával a Generátort.

Ezzel a folyamattal azt akarjuk elérni, hogy a Generátor egyre jobb képeket készítsen, ami annyire hasonlít egy igazi képhez, hogy meg tud téveszteni egy külső szemlélőt, mint például a Diszkriminátort. Ez a visszacsatolásos önfelügyelő tanítás sokkal élethűbb képeket tud előállítani mint más modell, de cserében sokkal nehezebb is megtalálni az egyensúlyt a komponensek között, hogy sikeresen menjen végbe a tanítás.

Gyakran felmerülő problémák

A GAN, mivel a saját maga által generált kimeneteire építi az optimalizációs lépését, így akár a legkisebb módosítás, illetve változás a hiperparaméterekben, félreviheti a tanítási folyamatot. A legtöbb hiba a Generátor és a Diszkriminátor közti egyensúly hiányában keresendő, amelyek több, különböző módon jelenhetnek meg.

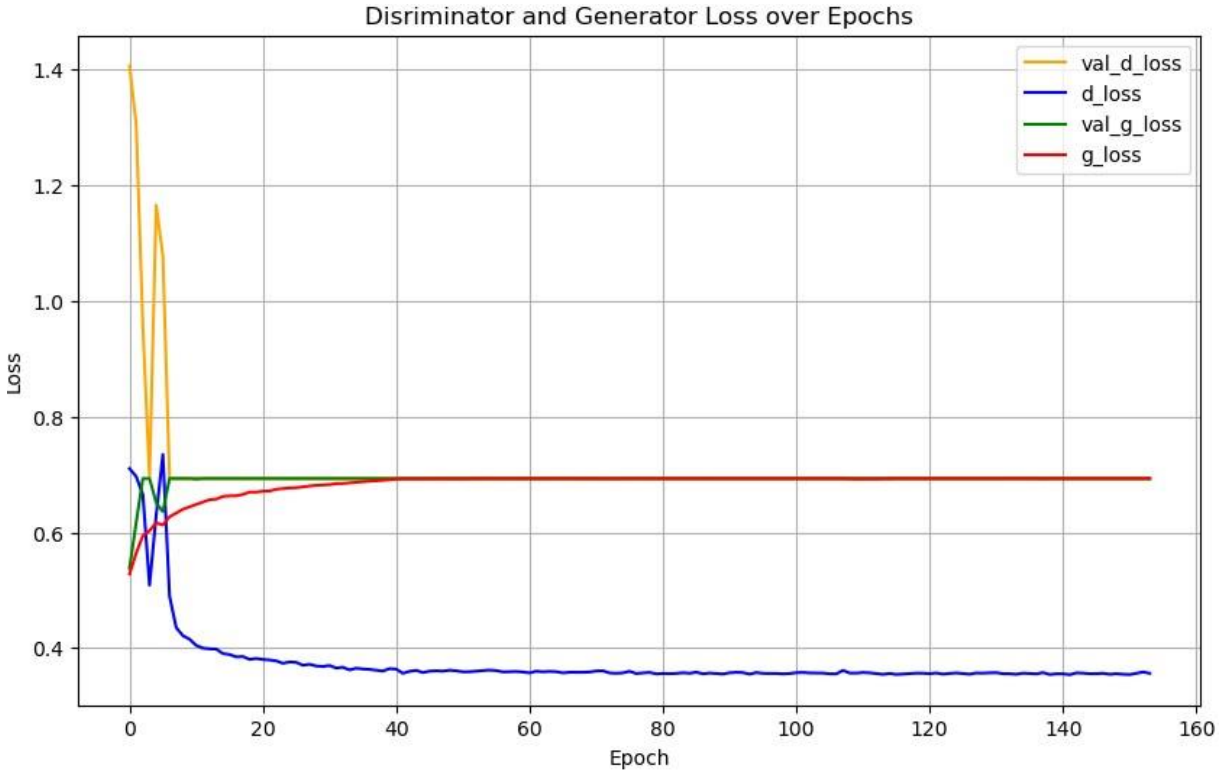
Modellek közti egyensúly hiánya

Az egyik leggyakoribb probléma, hogy az egyik komponens dominálja a másikat a tanítás során. Ha ez bekövetkezik, akkor gyakran szokott mód összeomlásban végződni az optimalizálás, melyben a domináns modell annyira ki tudja játszani a másikat, hogy az optimalizáló lépésnél nem hagy az utóbbinak javulásra lehetőséget. Ennek oka, hogy minden, amit generált kimenet az teljesen rossz és invalidált a másik modell által.

Ez gyakran a Diszkriminátor irányába történik meg. Ennek hátterében az áll, hogy a Diszkriminátornak csak egy bináris osztályozást kell elvégeznie arra vonatkozóan, hogy hamisított vagy igazi-e az adott kép. Ezzel szemben a Generátornak egy vektorból kell egy egész képet generálni. Ennek következtében az a kiinduló feltevés, hogy mind a két modell ugyanakkora számítási kapacitással rendelkezik, többnyire nem szokott megfelelő eredményt adni. Ezért szokás például a Diszkriminátor tanulását lekorlátozni valamilyen módon. Egy lehetséges opció, csökkenteni a learning rate-jét. Amennyiben ez nem éri el a kívánt hatást, úgy egy másik lehetőség az architektúra „lebutítása”, azaz a paraméterek csökkentése.

Eltűnő gradiensek

A gradiensek számítása az alapja annak, hogyan tudja az optimalizáló, milyen irányba kell módosítani a jelenlegi modellek súlyait, hogy az közelebb kerüljön a jobb teljesítményhez. Ha ez a visszacsatolás bekerül egy olyan völgybe, ahol ezen gradiens a 0-hoz közeli értéket vesz fel, akkor a tanulás drasztikusan le tud lassulni, vagy esetleg meg is állíthatja a modell konvergálását, effektíve bezárva az akkori állapotába. Az eltűnő gradienst oly módon lehet felismerni a loss értékből, hogy a mért értékek nem változnak semmit, teljesen kilapulnak. Erre mutat egy példát a 2. ábra.



2. ábra Példa az eltűnő gradiensre

Overfitting

A GAN architektúrák eléggé érzékenyek a túltanulásra, vagyis az overfitting-re. Ennek oka, hogy a duális struktúrája miatt több mindenre tud túltanulni a modell. Ezek lehetnek a következők.

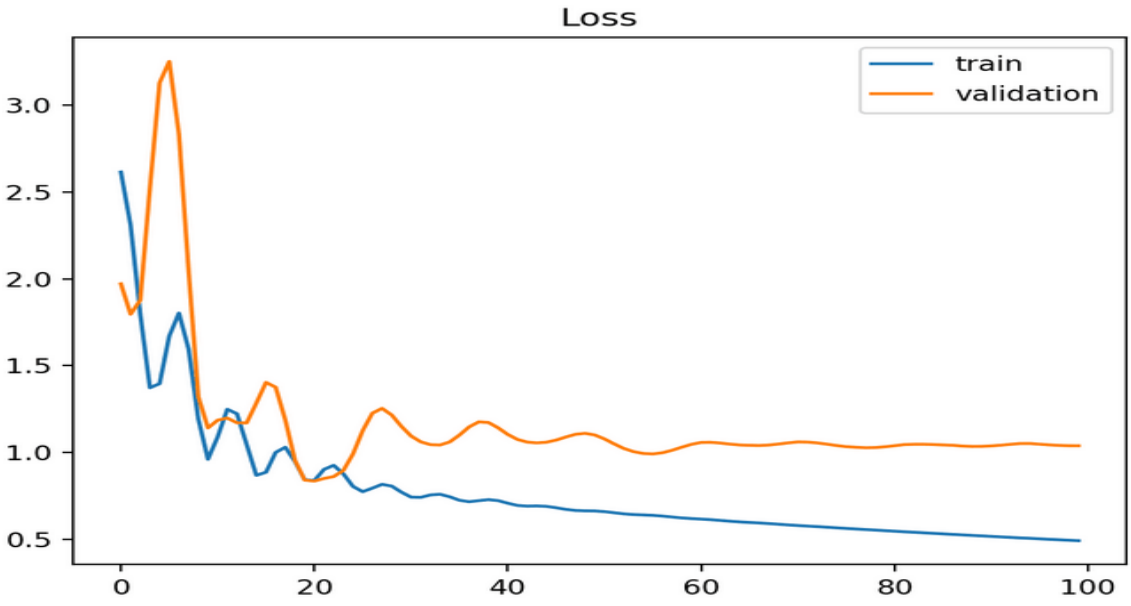
1. Adatbázisra

Amikor a Generátor és a Diszkriminátor elér egy egyensúlyi állapotot, melyben a Diszkriminátor nem tudja jobban megkülönböztetni az eredetit az igazitól és a Generátor sem tud jobb képeket generálni, akkor kialakulhat egy olyan versengés a két modell között, melyben szabadszemmel nem látható mintákkal kezdenek játszani, és nem azzal, hogy mennyire hasonlít a kép az igazira. [6]

2. Egymásra

Mivel a GAN alapja két egymással versengő modell, így az egyik modell képes túltanulni a másik modellre, nem hagyva neki semmi javulásra lehetőséget.

Az eltűnő gradienssel ellentétben az overfitting esetében a modell egyre jobban túltanul a tanítási adatbázisra. Így a tanítási adaton nem látunk szignifikáns változást, addig párhuzamosan a validálási adatbázisra romlást tapasztalunk (lásd 3. ábra).



3. ábra Példa a túlillesztésre

Problémák megelőzése

A következő eszközök merülhetnek fel arra vonatkozóan, hogy hogyan lehet mind a két modellt jó úton tartani a tanítás során.

Learning rate

Érdemes a Diszkriminátornál a kisebb learning rate használata, hogy ne overfitteljen túl hamar a Generátorra.

Optimalizáló lépés arányának állítása

Érdemes lehet több látens vektort is véletlenszerűen mintavételezni és ezáltal több optimalizáló lépést tennie a Generátornak epoch-onként, hogy a látens tér minél nagyobb részén legyen lehetősége végig iterálni. Ugyanakkor ez természetesen függ a tanítási adat mennyiségétől, és a bemeneti zaj vektor diverzitásától is.

R1 penalty

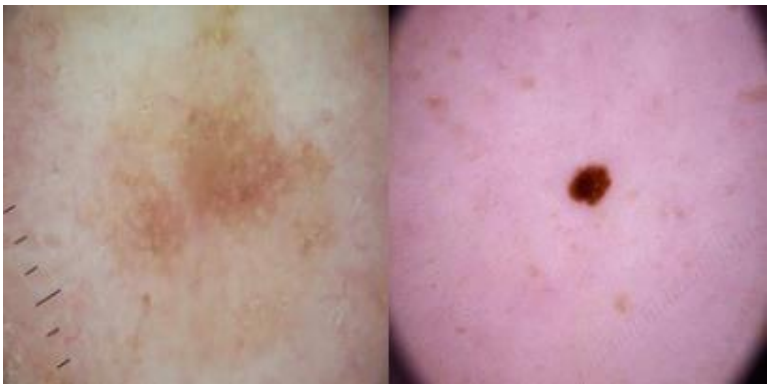
Az R1 penalty egy szabályozási technika, amelyet GAN-ok képzése során alkalmaznak. Célja, hogy a Diszkriminátor modellt megakadályozza abban, hogy elhagyja a Nash-egyensúlyt a valódi adatokon. Ez a szabályozás a Diszkriminátor gradienseinek nagyságát bünteti a valódi adatokon, megakadályozva, hogy a Diszkriminátor túl érzékeny legyen a valódi mintákra. [7]

Normalizálás

Érdemes a rétegek között valamilyen normalizálást beiktatni, hogy az értékek stabil fodrozódását fent tudjuk tartani az egész modellen keresztül.

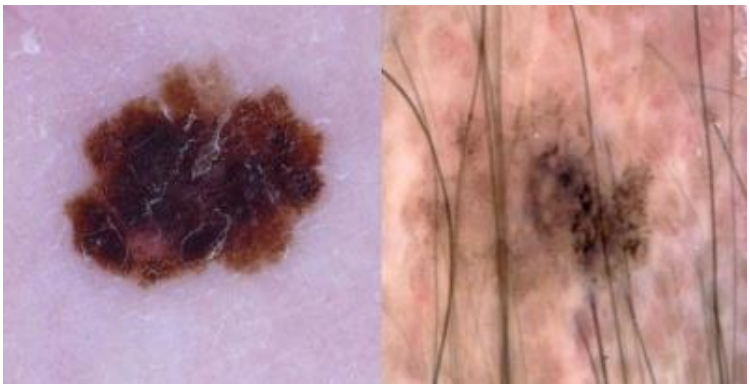
3. Adatbázis

Az adatbázis, amit felhasználtam a Kaggle oldaláról [8] érhető el, azon belül is a Skin Cancer (ISIC Images)[9] adatbázist alkalmaztam. Az adathalmazról röviden az alábbiakat lehet megjegyezni: “This is a balanced dataset of benign (healthy) and malignant (infected) skin spots. Transfer learning and CNNs can be used to classify the images into these two categories.” (Skin Cancer (ISIC Images), 2024). Azonban érdemes és jó szokás az adatbázist is saját magunknak elemezni, hogy tudatában legyünk a rendelkezésre álló adatpontok egyéb tulajdonságaival, esetleges szembetűnő gyengeségeivel.
Megvizsgálva az adatbázist, látható, hogy a képek színes, vizuális bőrminták, besorolva két osztályba benign (jóindulatú) 1800 képpel és malignant (rosszindulató) 1497 képpel, amely egy többnyire kiegyensúlyozott eloszlás a megjelölt osztályok között.
A benign (jóindulatú) osztályból származó mintaképek a 4. ábrán láthatóak.



4. ábra Példa benign osztályból

A malignant (rosszindulatú) osztályból származó mintaképek a 5. ábrán láthatóak.



5. ábra Példa a malignant osztályból

Adatbázis elemzése

Ránézve a képekre, észrevehetünk több mintát is, ami befolyásolhatja a végeredményt. Csak párat említve:

- jelentős része a képeknek tartalmaz szőrzetet,
- bizonyos képek egy lencse hatást mutatnak, melyben egy plusz keretet kaphat a képünk, ami nem hordoz semmilyen információt.

Ezeket érdemes lehet eltávolítani a tanítási adatbázisból, hogy ne zavarja be vagy esetleg ne vigye el rossz irányba a képgenerálást a zajos, ún. outlier képekkel. A kevés adat mennyiségére való tekintettel nem távolítottam el ezen képeket, hiszen ezzel drasztikusan csökkent volna a generálás optimalizálása során felhasználható adatmennyiség. Ugyanakkor a jövőben érdemes lehet letesztelni a képgenerálást ezek eltávolításával is.

Ahhoz, hogy egy GAN modell megfelelően tudja ezeket a képeket beolvasni, meg kell bizonyosodni róla, hogy a bemeneti képek tulajdonságai megegyeznek. Így leellenőriztem, hogy minden kép ugyanazzal a felbontással rendelkezik-e, mint az első, azaz minden képre igaz, hogy:

- színes kép, 3 csatornával,
- a kép mérete 224x224 pixel.

Ha esetleg lenne az adathalmazban szürke-skálás kép, az akkor okozna problémát, amikor a tanítás közepén próbálná betölteni a szkript. Ez hosszas debuguláshoz vezethet, hogy miért is akadt el a tanítás közepén, így ezt feltétlenül érdemes ellenőrizni. Ha azonban a képek nem ugyanabban a felbontásban vannak jelen, akkor az gondot okozhat, amikor a dataloader átméretezi, mert a modell egy torzított képen fog tanulni.

Ehhez végig kell menni az összes adatpontokon, majd megvizsgálni a kép szélességét, magasságát és a csatornái számát, hogy egyezik-e.

Az adatbázis ellenőrzéséhez kapcsolódó kód az alábbi kódrészletben látható.

```
import os
from PIL import Image


healthy_dir = os.path.join(path, 'benign')
infected_dir = os.path.join(path, 'malignant')


# List all files in the directories (original code)

healthy_images = [f for f in os.listdir(healthy_dir)]
infected_images = [f for f in os.listdir(infected_dir)]

print(f"Number of healthy images: {len(healthy_images)}")
print(f"Number of infected images: {len(infected_images)}")



def validate_images(image_dir, category_name):

    invalid = []

    for img_name in os.listdir(image_dir):

        img_path = os.path.join(image_dir, img_name)

        try:

            with Image.open(img_path) as img:

                # Check image dimensions

           if img.size != (224, 224):

                invalid.append(f"{category_name}/{img_name} (Size:
{img.size})")
                # Check if image is RGB

                elif img.mode != 'RGB':
               invalid.append(f"{category_name}/{img_name} (Mode:
{img.mode})")
    except Exception as e:
        invalid.append(f"{category_name}/{img_name} (Error:
{str(e)})")
return invalid


# Validate both directories
```

```
invalid_healthy = validate_images(healthy_dir, 'benign')

invalid_infected = validate_images(infected_dir, 'malignant')

all_invalid = invalid_healthy + invalid_infected

if not all_invalid:
    print("\nAll images are valid RGB images of size 224x224.")
else:
    print("\nValidation issues found:")
    for issue in all_invalid:
        print(f"  - {issue}")
    print(f"\nTotal invalid images: {len(all_invalid)}")
```

Most, hogy megbizonyosodtam róla, hogy az adathalmazban a képek felbontásai és csatornái megegyeznek, így egyértelműen kijelenthető, hogy a bemeneti minták a következő dimenziókkal fognak rendelkezni: [3, 224, 224]. Ahhoz, hogy biztosítsam a bemeneti kép pixelértékeinek megfelelő végigfodrozódását az egész modellen keresztül, érdemes az adatokat normalizálni.

Adatok betöltése

Az adatok betöltése a PyTorch Lightning LightningDataModule [12] osztályból öröklődött, mely elősegíti az adatok egyszerűbb transzformálását és kezelését.

Az adatok dúsítására a következő módszereket használtam fel:

- RandomHorizontalFlip [13]
- RandomVerticalFlip [14]
- RandomRotation(degrees=45) [15]
- RandomResizedCrop(size=(64, 64), scale=(0.8, 1.0)) [16]
- Normalize(átlag, szórás) [17]
- ToTensor [18].

Az adatok betöltéséhez tartozó kód a következő kódrészletben látható.

```
class DataModule(pl.LightningDataModule): def init(self,
data_dir=path, batch_size=BATCH_SIZE, num_workers=NUM_WORKERS):
super().init() self.data_dir = data_dir self.batch_size =
batch_size self.num_workers = num_workers

    self.transform = transforms.Compose(
        [
            # Random horizontal flip (50% chance)
            transforms.RandomHorizontalFlip(),

            # Random rotation in the range of -45 to +45 degrees
            transforms.RandomRotation(degrees=45),

            # Random cropping followed by resizing to the
original size
            transforms.RandomResizedCrop(size=(64, 64),
scale=(0.8, 1.0)),

            # Color jittering (brightness, contrast, saturation,
hue)
```

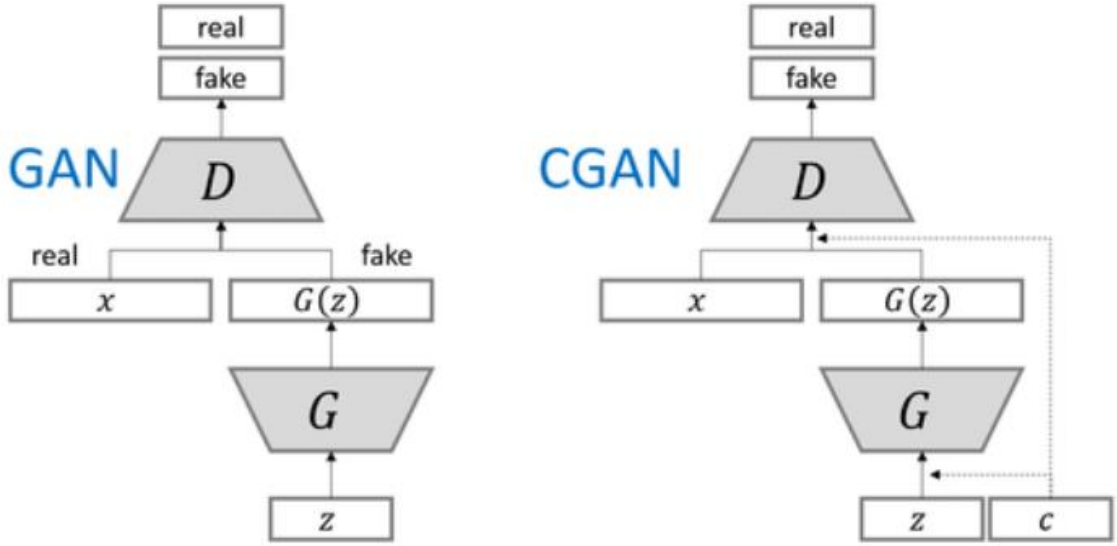
```
            transforms.ColorJitter(brightness=0.2, contrast=0.2,
saturation=0.2, hue=0.1),

            # Convert image to tensor
            transforms.ToTensor(),

            # Normalize using precomputed mean and std
            transforms.Normalize((0.7591, 0.5373, 0.5387),
(0.1530, 0.1611, 0.1772)),
        ]
    )

def setup(self, stage=None):
    if stage == "fit" or stage is None:
        # Load the full dataset
        dataset_full = datasets.ImageFolder(root=self.data_dir,
transform=self.transform)

        # Extract labels for stratified splitting
        targets = [sample[1] for sample in dataset_full.samples]

        # Split into train+val (50%) and test (50%)
        train_val_indices, test_indices = train_test_split(
            range(len(dataset_full)),
            test_size=0.01,
            stratify=targets,
            random_state=42
        )

        # Split train+val (50%) into train (25%) and val (25%)
        train_indices, val_indices = train_test_split(
            train_val_indices,
            test_size=0.3,
            stratify=[targets[i] for i in train_val_indices],
            random_state=42
        )

        # Create subsets for train, val, and test
        self.dataset_train = Subset(dataset_full, train_indices)
        self.dataset_val = Subset(dataset_full, val_indices)
        self.dataset_test = Subset(dataset_full, test_indices)

def train_dataloader(self):
    return DataLoader(self.dataset_train,
batch_size=self.batch_size, num_workers=self.num_workers,
shuffle=True)

def val_dataloader(self):
    return DataLoader(self.dataset_val,
batch_size=self.batch_size, num_workers=self.num_workers,
shuffle=False)
```

```
def test_dataloader(self):
    return DataLoader(self.dataset_test,
batch_size=self.batch_size, num_workers=self.num_workers,
shuffle=False)
```

4. Modell

A képek generálásához egy GAN alapú modellből indultam ki, amely működését már kifejtettem korábban. Azonban, hogy lehetséges legyen egy másik modell eredményét is tesztelni, úgy kell képeket generálni, hogy a címkék kihatással legyenek a képek tartalmára, és meg is tudjuk tartani ezt a címkét a későbbi kiértékelésekhez.

Ehhez azonban a GAN-t módosítani kell, hogy be tudjon fogadni címkét is mint bemeneti paramétert. Erre ad megoldást a cGAN - Conditional Generative Adversarial Network[23]. Ez utóbbi egy plusz bemenetet biztosít a címkének, melyet be tud ágyazni és integrálni a modell felépítésébe. A hagyományos GAN architektúra összehasonlítása a cGAN modellel a 6. ábrán látható.



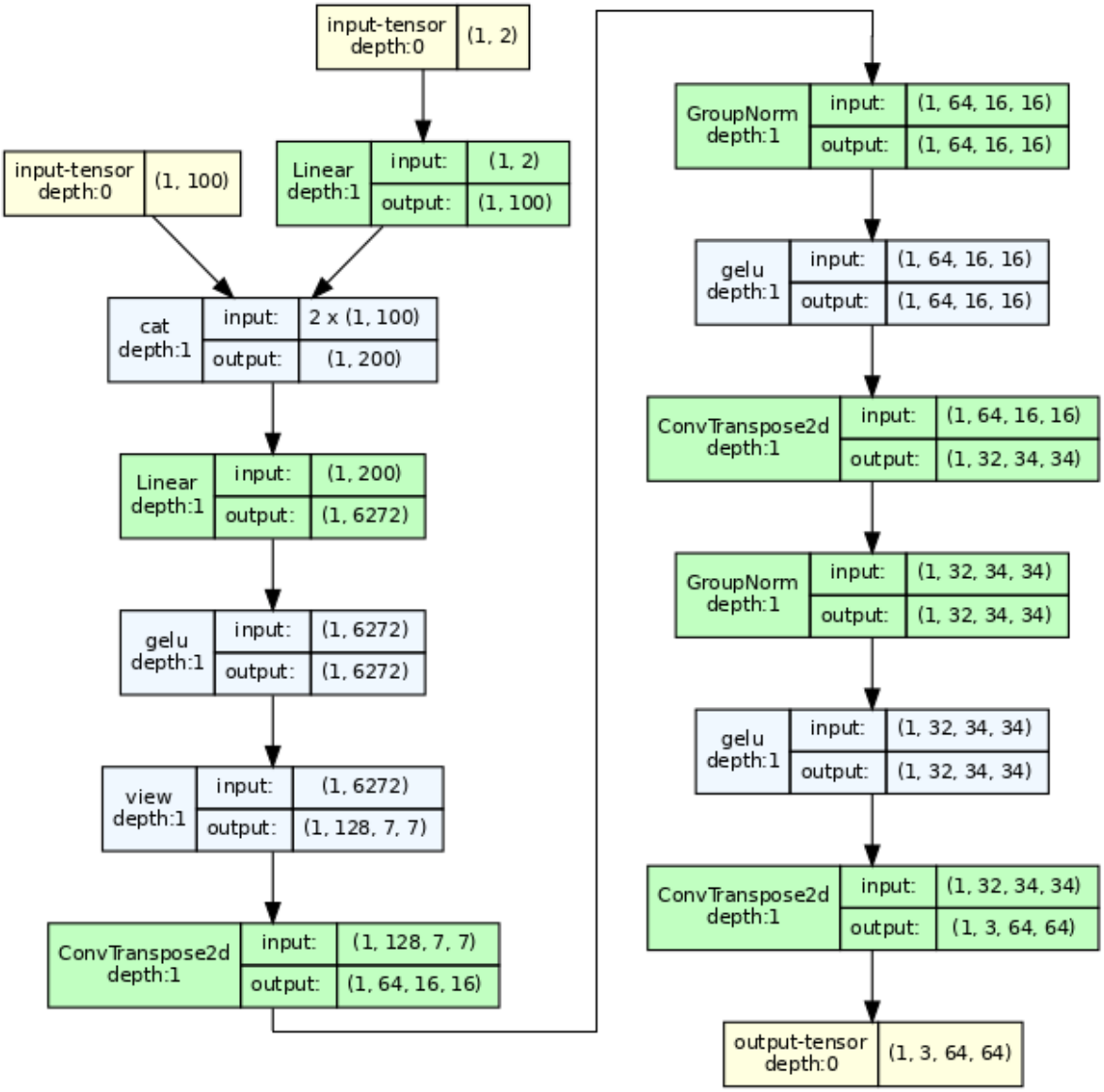
6. ábra GAN és cGAN felépítése[10]

Mint ahogy az ábrából is látszik, a modell két fő részből áll:

- Generátor (G),
- Diszkriminátor (D).

Generátor

A Generátorhoz használt architektúra a 7. ábrán látható. A költséges számítások miatt a RandomResizedCrop átméretezte a bemeneti képeket 3 x 64 x 64-es felbontásúra, csökkentve ezzel a már így is eléggé költséges számítási igényét a modellnek.



7. ábra a Generátor felépítése

Mint ahogyan az látható, a modellnek két bemenete is van. Ezek közül [n, 100] az a látens vektor, ami reprezentálja a kép tartalmát, ahol n a mini-batch számossága. A másik bemenet [n, 2] pedig egy one-hot kódolt vektor, ami reprezentálja, hogy a bemeneti kép az benign (jóindulatú), vagy pedig malignant (rosszindulatú).

A címke vektort egy lineáris réteggel beágyazom, hogy hasonló arányban legyen kihatással a modell kimenetére, mint maga a látens vektor. Ezután összekonkatenálom és egy lineáris réteggel felskálázom a megfelelő méretre, azt pedig egy torch.tensor.view fügvénnyel átformálom egy [n, 128, 7, 7]-es mátrixra. Erre meghívva többször is ConvTranspose2d és GroupNorm majd a GELU aktivációs rétegeket alkalmazom, amíg nem éri el a kívánt [n, 3, 64, 64] -es kimenetet.

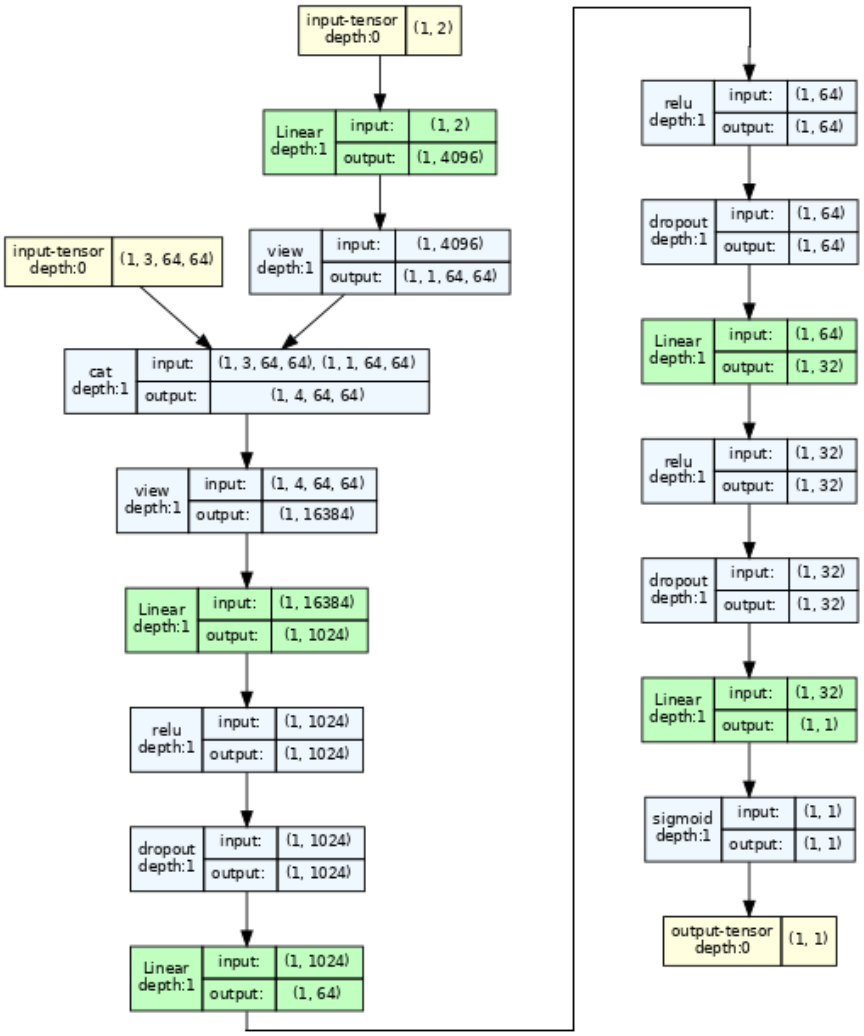
Fontos megjegyezni, hogy a tanítási adatbázisban lévő képekkel megegyező méretet kell generálnia a modellnek. Ez azért szükséges, hogy a Diszkriminátor tudjon vele dolgozni.

A tanítás során az alábbi hiperparamétereket alkalmaztam:

- Learning rate: 0.0005,
- Optimizer: Adam[24],
- Batch size: 64.

Diszkriminátor

A Diszkriminátorhoz használt architektúra a 8. ábrán látható. A Diszkriminátor feladata, hogy a 3 x 64 x 64-es képeket beolvasva rájöjjön, hogy az az igazi adatbázisból származik vagy esetleg a generátor által készített hamisított kép. Így a kimenete egy bináris osztályozás.



8. ábra a Diszkriminátor felépítése

A címkék beágyazása teljesen megegyezik a Generátoréval, csak annyi a különbség, hogy a beágyazás folyamán a címkét egy plusz kép csatornává bővítjük ki, így a Diszkriminátor beágyazás utáni bemenete a 4 x 64 x 64, ahol a 4. csatorna a címke. Ezt csak végig vezettem számos rétegen, amely a következő módon ismétlődött: Linear, ReLU, Dropout. Végül egy Linear réteg lecsökkenti egy skalár étékre és az egy Sigmoid aktivációs réteggel zárul, hogy valószínűségi értéket adjon vissza.
A tanítás során az alábbi hiperparamétereket alkalmaztam:

- Learning rate: 0.001,
- Optimizer: Adam[24],
- Batch size: 64.

Osztályozó modell felépítése

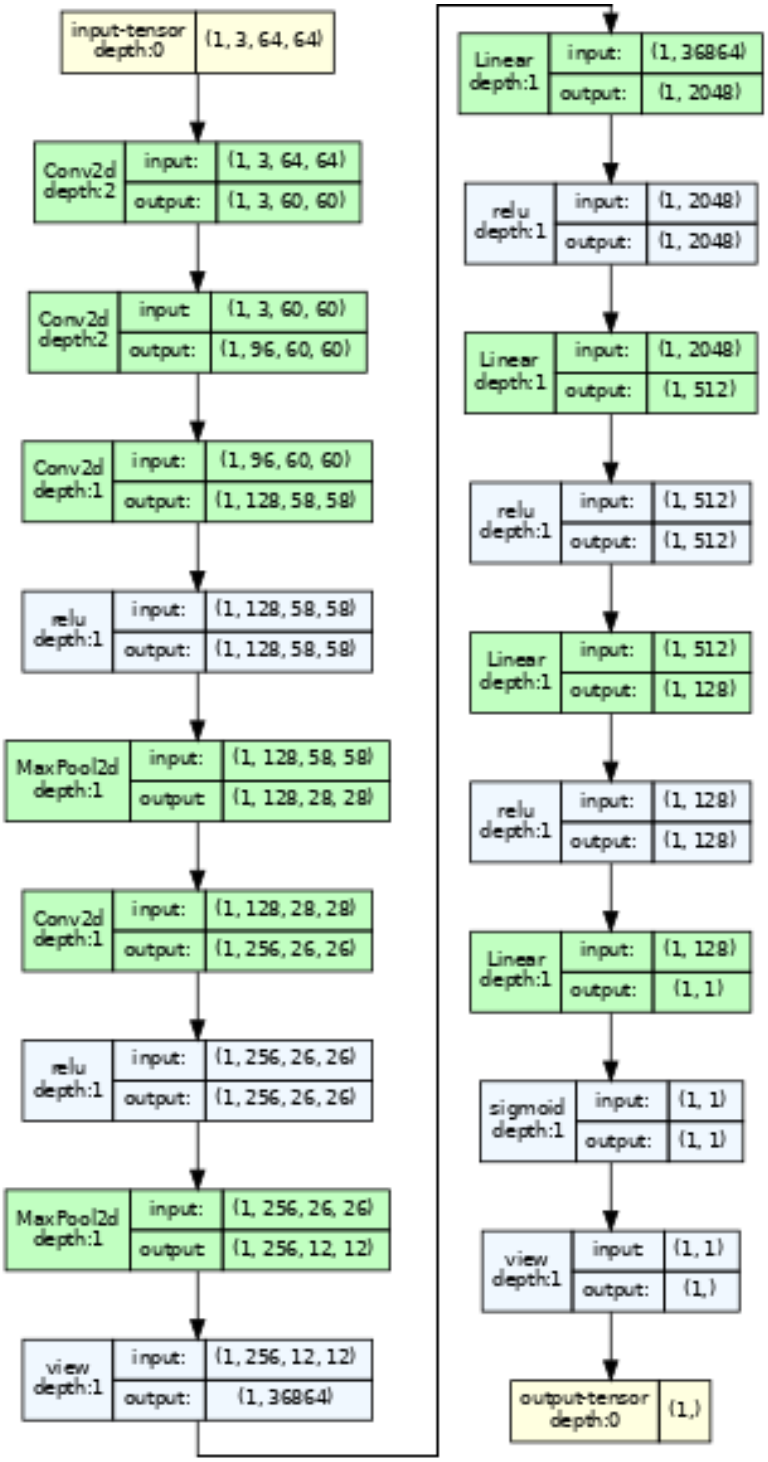
Ahhoz, hogy mérhető is legyen a modell pontossága, kelleni fog egy kiértékelési metódus. Úgy határoztam, hogy egy egyszerűbb bináris Osztályozó kiértékelése jól demonstrálhatja, hogy milyen kihatással lesz a teljesítményre, ha a tanítási adatbázist felcserélném szintetikus adatpontokkal. Az implementált architektúra az 5. ábrán látható.

A bemenet egy separable convolution-be megy be [19], ami nem csökkenti lényegesen a felbontást, csak dúsítja a csatornák számát. Utána felváltva egyszerű conv2D, Relu, MaxPool2D rétegeken megy át. A végén egy Lineáris réteg kiad egy skalár értéket az eredménnyel.

A tanítás során az alábbi hiperparamétereket alkalmaztam:

- Learning rate: 0.0005,
- Optimizer: Adam [24]
- Batch size: 64.

Az osztályozó modellhez használt architektúra a 9. ábrán látható.



9. ábra Az osztályozó modell felépítése

5. A tesztelés menete

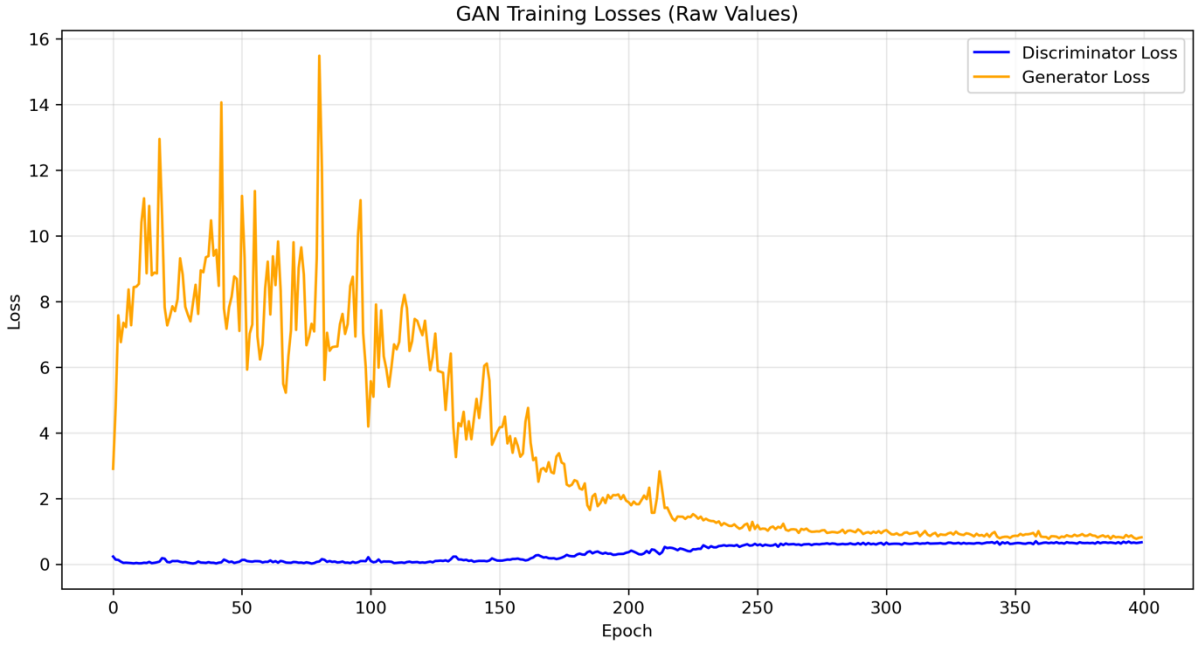
A kísérlet azt vizsgálja, hogy az eredeti adatbázis hány százalékát tudom kipótolni szintetikusan generált adatpontokkal. Ennek megvalósításához előálltam a következő tesztelési metódussal. Mindegyik GAN tanítása ugyanazon a modellel megy végbe módosítás nélkül. Ennek oka, hogy ezáltal csakis az adat minőségének változása a változó faktor. A GAN számítási és bonyolultsági igényei miatt csakis 400 epoch-on át kerül tanításra, majd ezt követően kerül felhasználásra a képek generálására.

Hasonló módon a bináris osztályozó is változatlan marad futtatások között. A korábban említettekhez hasonlóan ennek is az volt az oka, hogy csak az adat minősége legyen a változó faktor. Az osztályozó könnyed tanítása miatt csak 50 epoch-ig lett tanítva. Ezt követően a betanított modell kerül tesztelésre az eredeti adatbázison a hamisított képek nélkül.

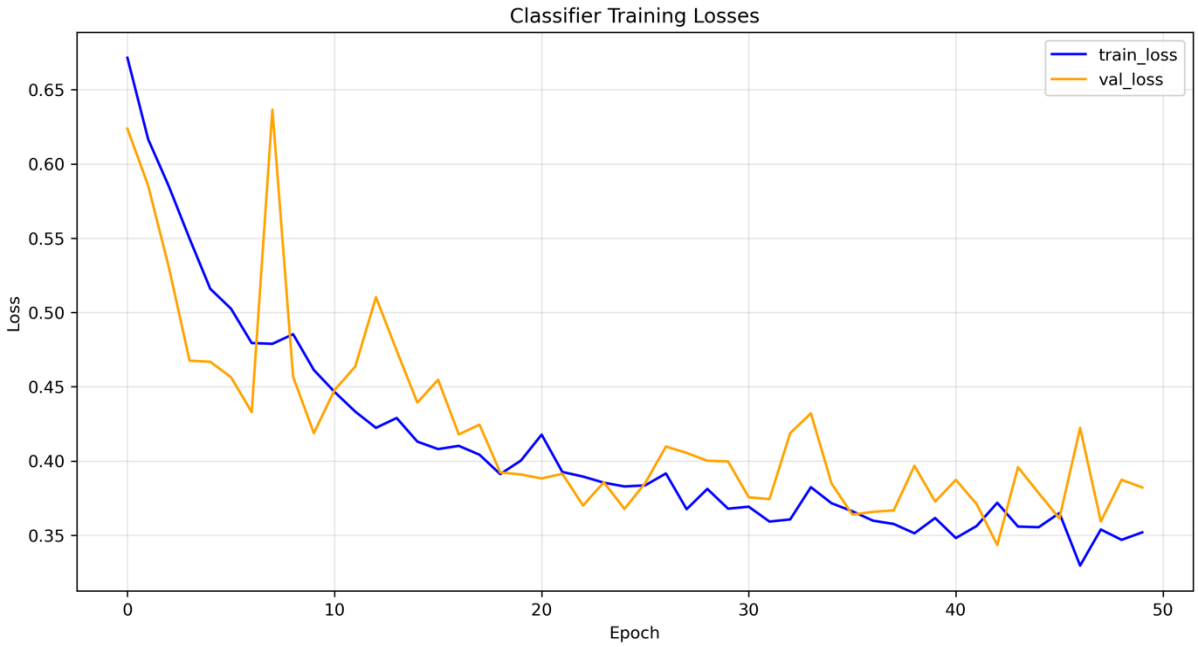
Az eredményeket az osztályozó metrikái alapján értékelem ki, figyelembe véve a következő metrikákat: Accuracy, Precision, Recall.

1. Kontrollcsoport kialakítása

Elsőnek tesztelni kell, hogy milyen az alap teljesítménye a modelleknek. Erre azért van szükség, hogy lehessen mihez viszonyítani a szintetikus adatok használatával betanított modellek teljesítményét. A kontrollcsoport ennek megfelelően a rendelkezésre álló adatok egészén, azaz 100%-án tanult. Az ilyen módon betanított modell eredményei a 10. és 11. ábrákon láthatóak.



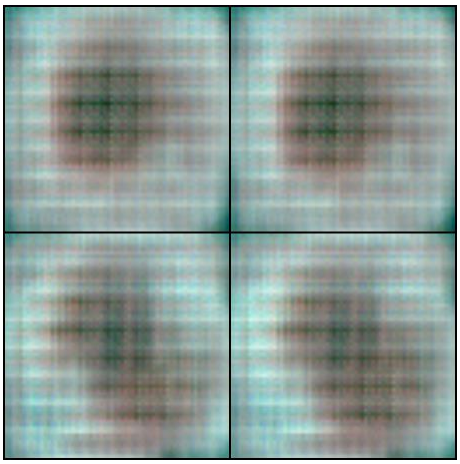
10. ábra Kontrollcsoporton tanult Generátor és Diszkriminátor loss értékei



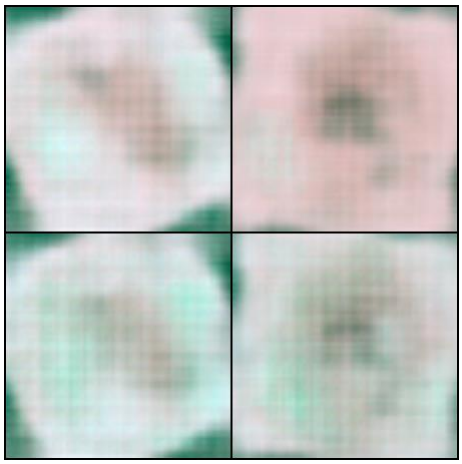
11. ábra Kontrollcsoporton tanult osztályozó train_loss és val_loss értékei

Mint ahogy ez látható, a tanítás a kontrollcsoportra lett igazítva, mivel mind a GAN (lásd 10. ábra) és az osztályozó modell (lásd 11. ábra) is a max epochnál eljutott az ún. „plateau” fázisba (plateau stage) [25], amikor már nem látható további lényeges javulás.

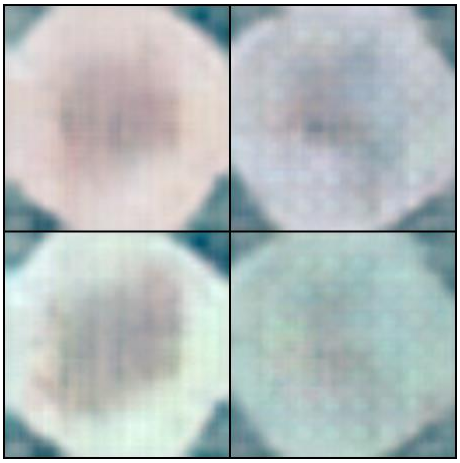
A generált képek minősége 12-15. ábrákon látható, 100 epoch-onként.



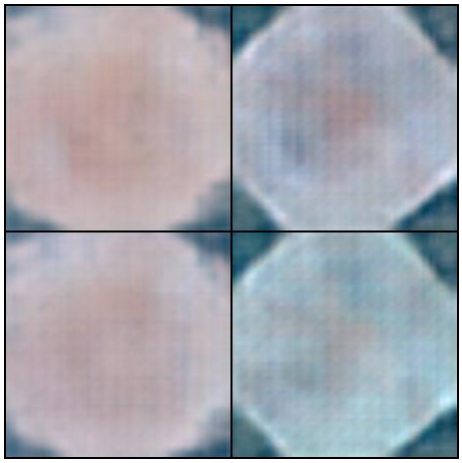
12. ábra Minták 100. epochnál



13. ábra Minták a 200. epochnál



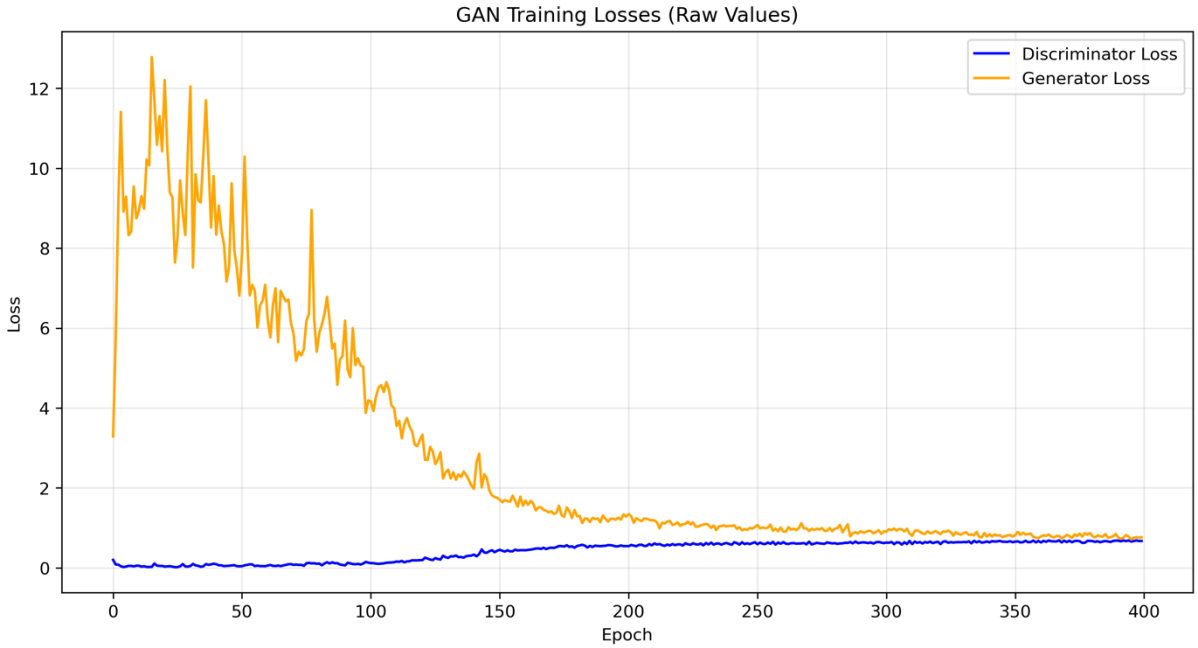
14. ábra Minták a 300. epochnál



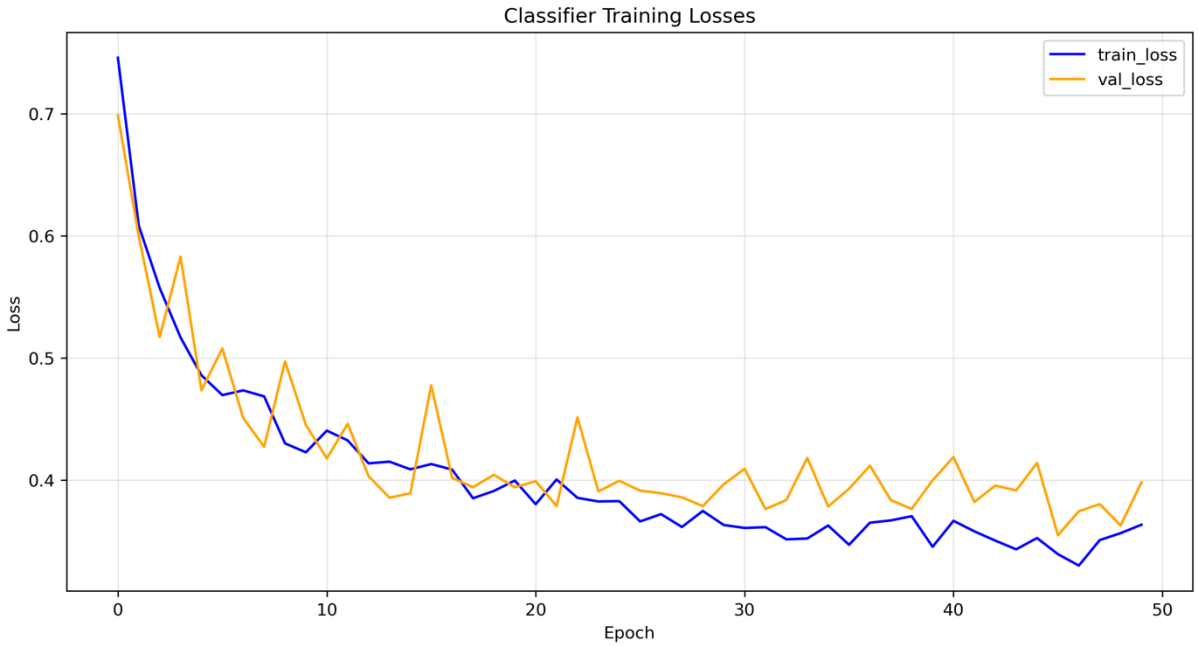
15. ábra Minták a 400. epochnál

2. Data99 csoport

Az ún. Data99 csoport adathalmaza 99 %-ban valós, eredeti képekből áll, a hiányzó 1 %-ot pedig ezen a csökkentett halmazon tanult GAN által generált, szintetikus képekkel van kiegészítve, hogy az adathalmaz mérete ne változzon. A tanítási eredmények a 16. és 17. ábrákon láthatóak.



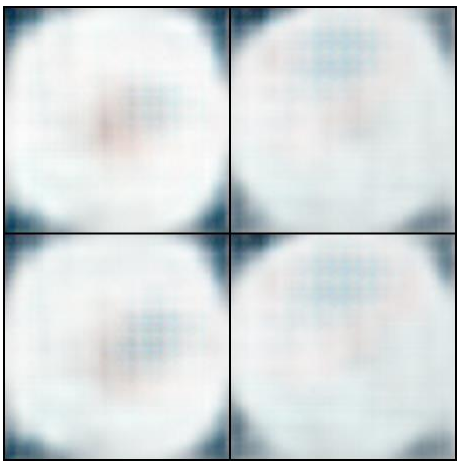
16. ábra Data99 csoporton tanult Generátor és Diszkriminátor loss értékei



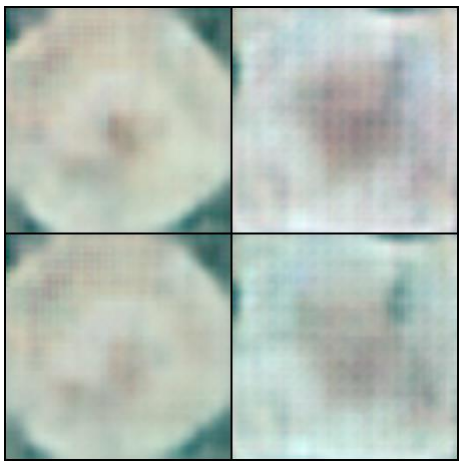
17. ábra Data99 csoporton tanult osztályozó train_loss és val_loss értékei

Látszik, hogy a „plateau” fázis [25] már hamarabb elkezdődött, mint a kontrollcsoportban. Ez a jelenség leginkább a GAN-on vehető észre (lásd 16. ábra).

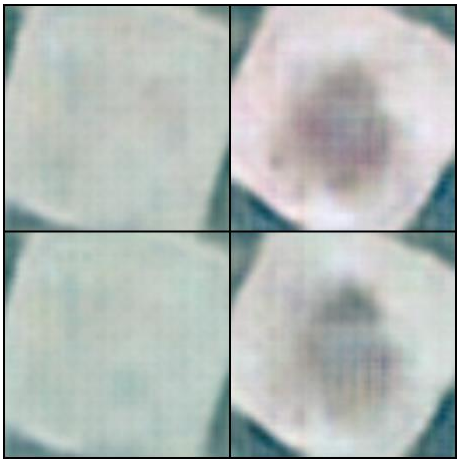
A generált képek minősége a 18-21. ábrákon látható, 100 epoch-onként.



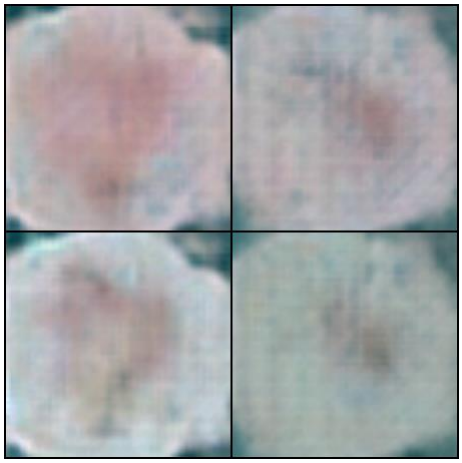
18. ábra Minták a 100. epochnál



19. ábra Minták a 200. epochnál



20. ábra Minták a 300. epochnál

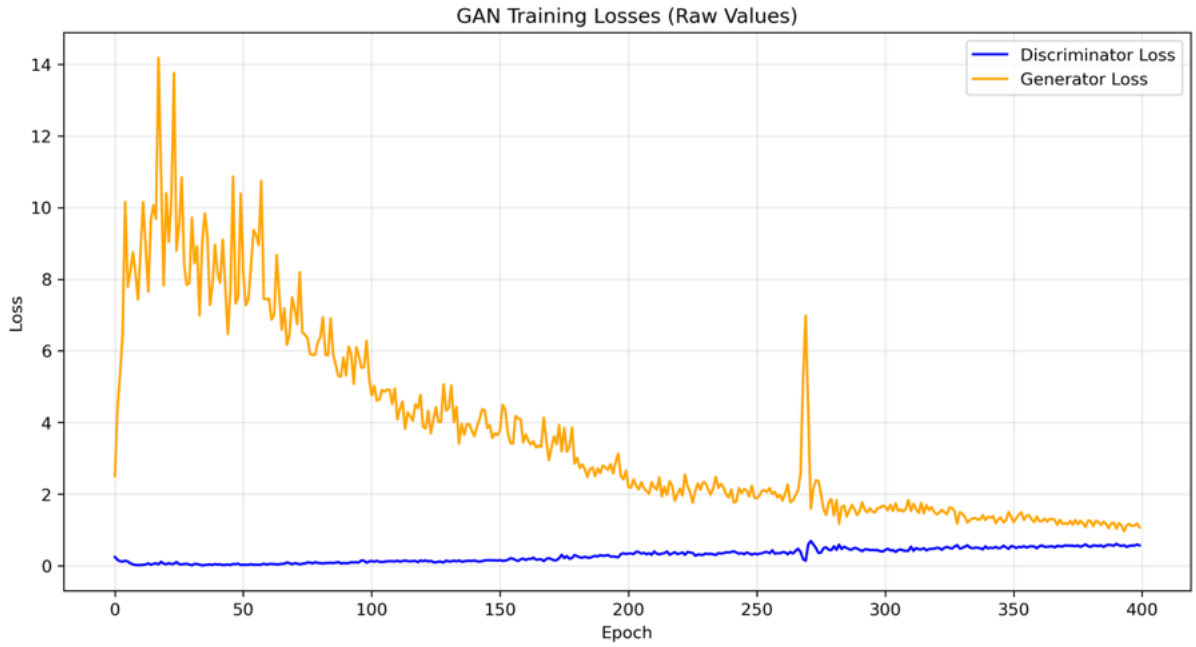


21. ábra minták a 400. epochnál

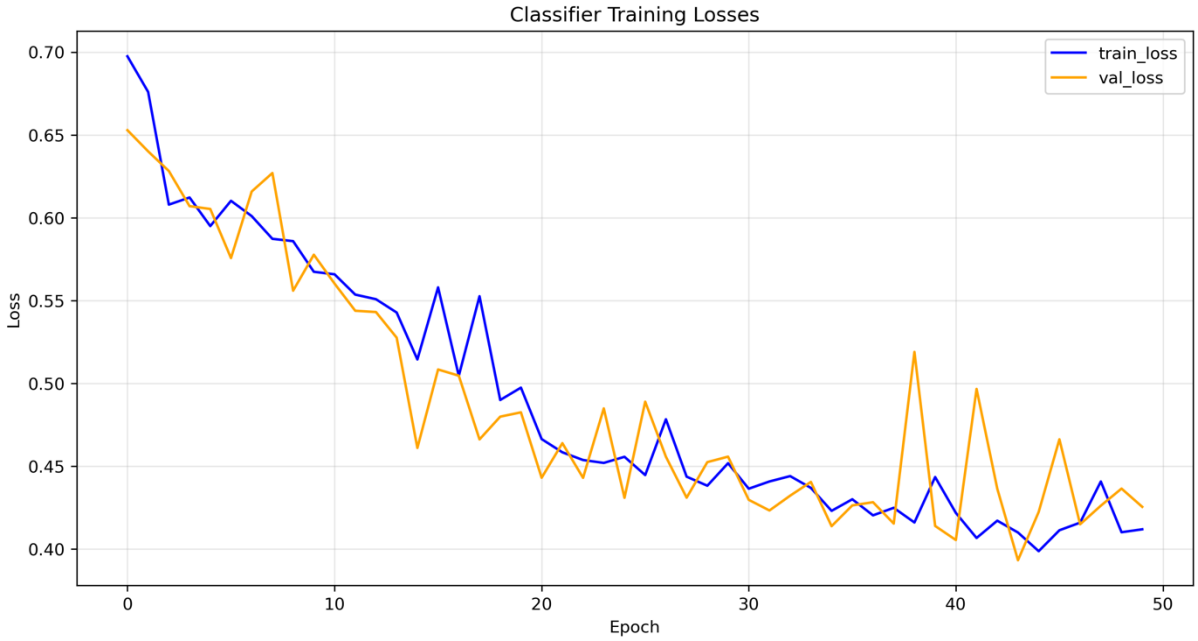
3. Data50 csoport

Az ún. data50 csoport adathalmaza 50%-ban valós, eredeti képekből áll, a hiányzó 50 % pedig ezen a csökkentett halmazon tanult GAN által generált, szintetikus képekkel van kiegészítve, hogy az adathalmaz mérete ne változzon.

A tanítási eredmények a 22. és 23. ábrákon láthatóak.



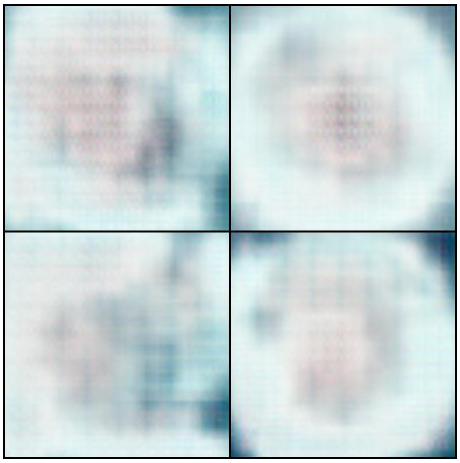
22. ábra Data50 csoporton tanult Generátor és Diszkriminátor loss értékei



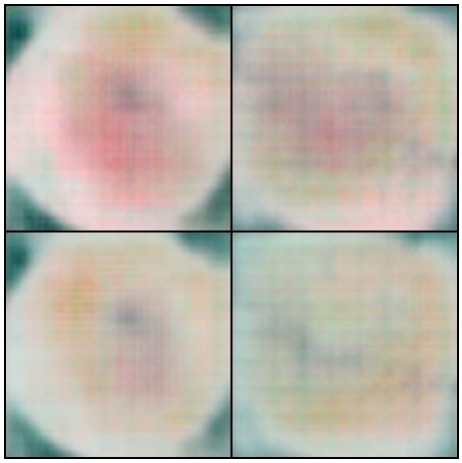
23. ábra Data50 csoporton tanult osztályozó train_loss és val_loss értékei

Ránézve a GAN (lásd 22. ábra) és az osztályozó (lásd 23. ábra) loss értékeire, úgy tűnik, hogy nehezen tudja megtanulni a képekben rejlő mintákat. A GAN-nak 400 epoch nem volt elég, hogy elérje a „plateau” fázisát, míg az osztályozón is látszódik, hogy 50 epoch alatt éppen csak megközelítette a 0,4-es loss értéket, sokkal később, mint más csoportok osztályozói eddig.

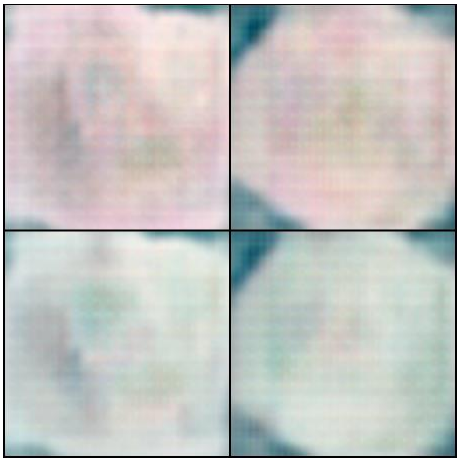
A generált képek minősége a 24-27. ábrákon látható, 100 epoch-onként.



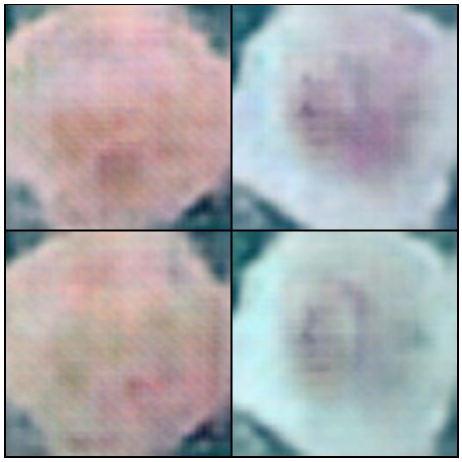
24. ábra Minták 100. epochnál



25. ábra Minták a 200. epochnál



26. ábra Minták a 300. epochnál



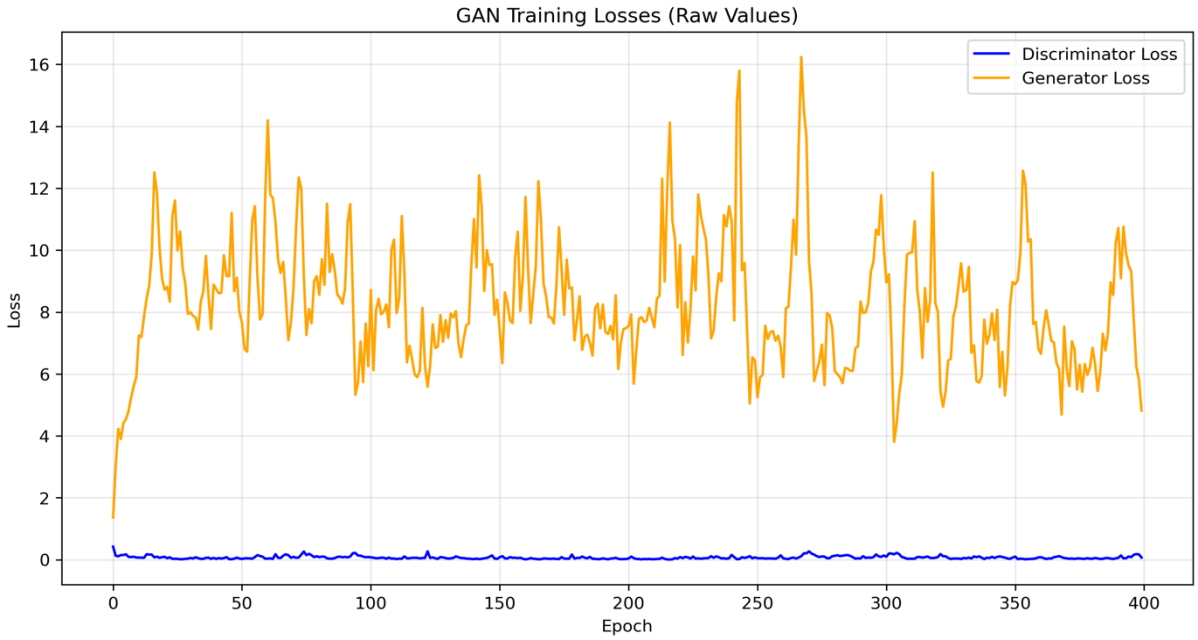
27. ábra Minták a 400. Epochnál

A generált képek alapján úgy tűnik, hogy maga a generátor nem vesztett lényegesen a generálásának pontosságából.

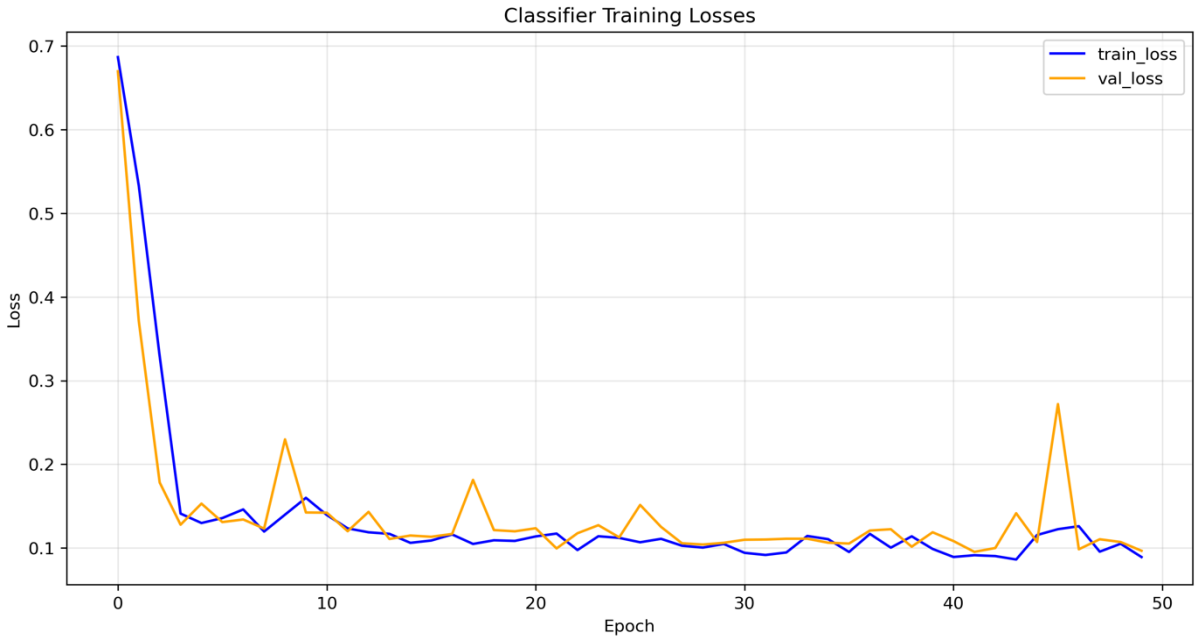
4. Data15 csoport

A Data15 csoport adathalmaza 15 %-ban valós, eredeti képekből áll, a hiányzó 85 % pedig ezen a csökkentett halmazon tanult GAN által generált, szintetikus képekkel van kiegészítve, hogy az adathalmaz mérete ne változzon.

A tanítási eredmények a 28. és 29. ábrákon láthatóak.



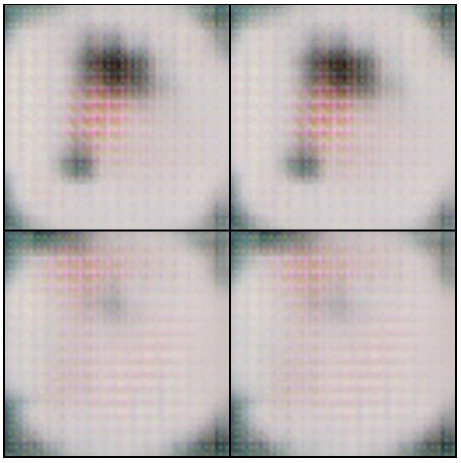
28. ábra Data15 csoporton tanult Generátor és Diszkriminátor loss értékei



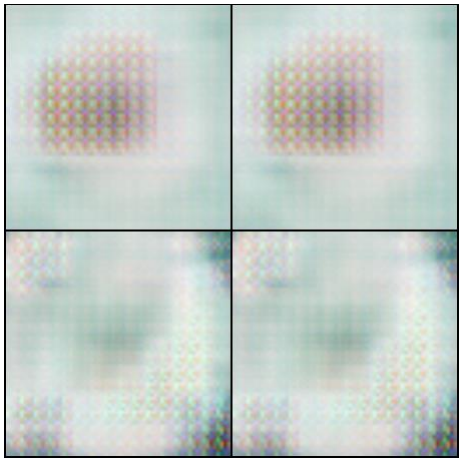
29. ábra Data15 csoporton tanult osztályozó train_loss és val_loss értékei

Az eredeti adatok 15%-ára csökkentése (~230 kép/osztály) már közel vitte a GAN-t (lásd 28. ábra) a mode collapse felé, melyben a modell nem tudna lényeges információkat megtanulni, de még így is felismerhető egy csökkenő trend a Generátor loss értékében, minimális tanulásra való képességre utalva. Utóbbi látszódik is a generált képeken (lásd 30-33. ábrák), ahol a kontúrok már szabad szemmel is kivehetőek. Ez a modell a jelenlegi beállítások mellet már az underfit jeleit mutatja.

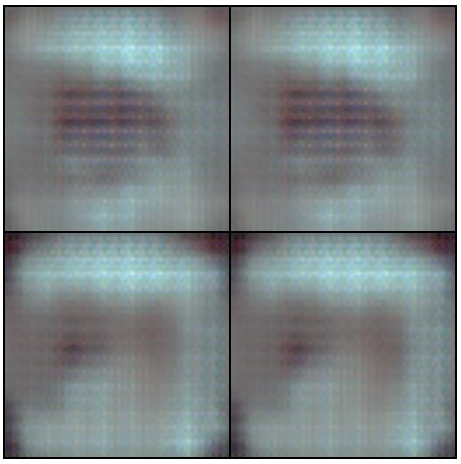
A generált képek minősége a 30-33. ábrákon látható, 100 epoch-onként.



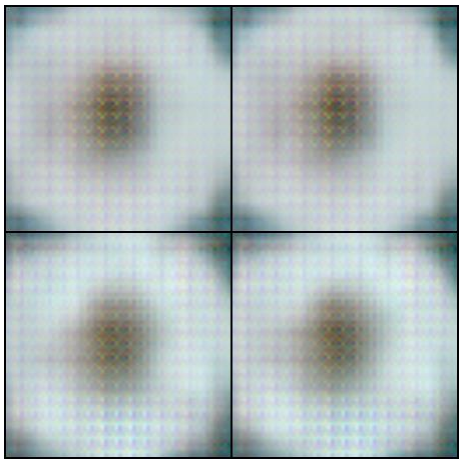
30. ábra Minták 100. epochnál



31. ábra Minták a 200. epochnál



32. ábra Minták 300. epochnál



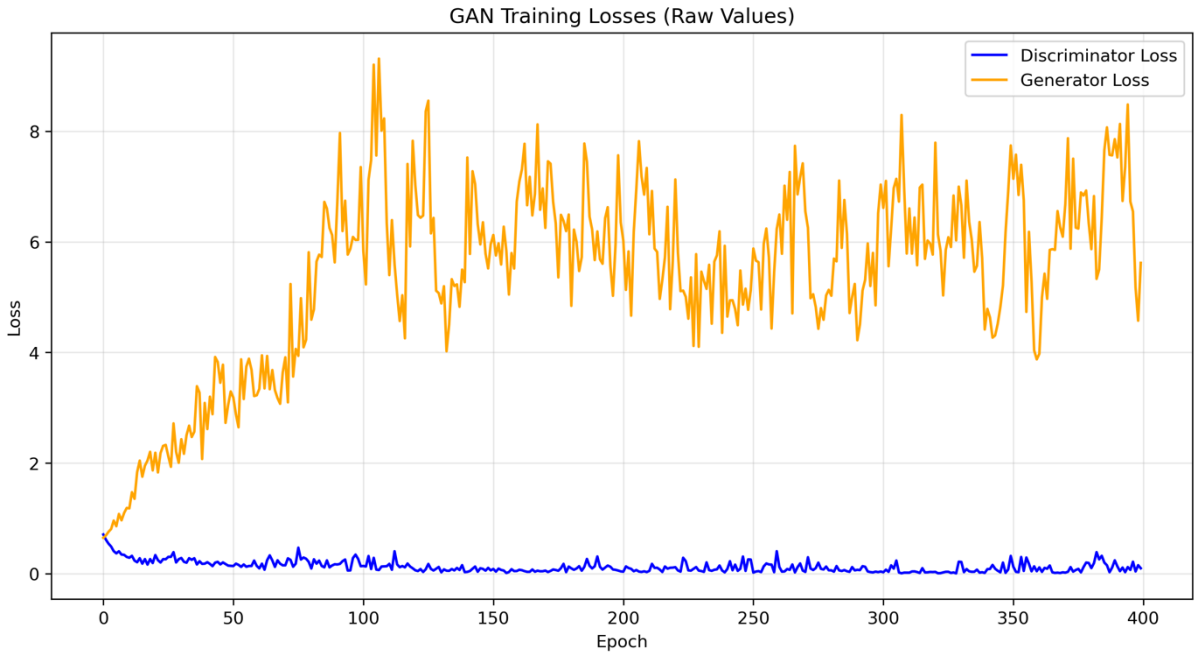
33. ábra Minták a 400. epochnál

Ahogy ez észrevehető, a 400. epochra a GAN már elkezdett mintákat felismerni. Az osztályozó továbbra is underfittelt a tanítási halmazra.

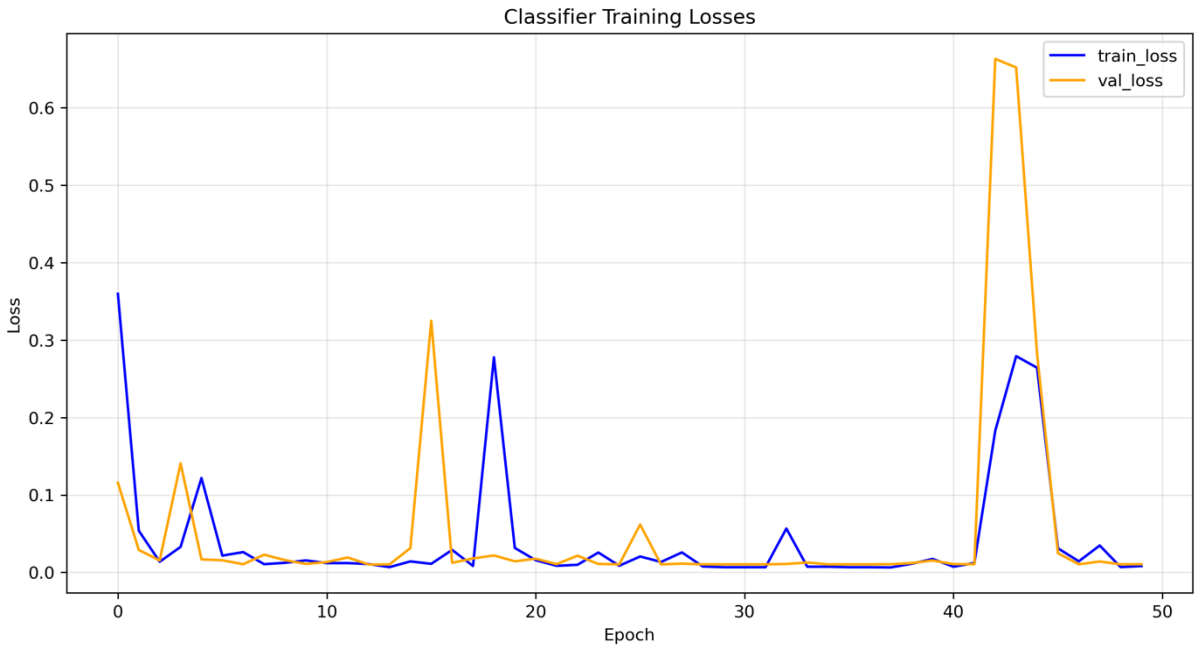
4. Data1 csoport

A Data1 csoport adathalmaza 1 %-ban valós, eredeti képekből áll, a hiányzó 99 % pedig ezen a csökkentett halmazon tanult GAN által generált, szintetikus képekkel van kiegészítve, hogy az adathalmaz mérete ne változzon.

A tanítási eredmények a 34. és 35.ábrákon láthatóak.



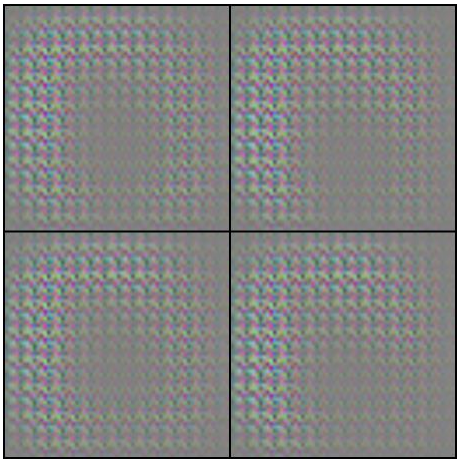
34. ábra Data1 csoporton tanult Generátor és Diszkriminátor loss értékei



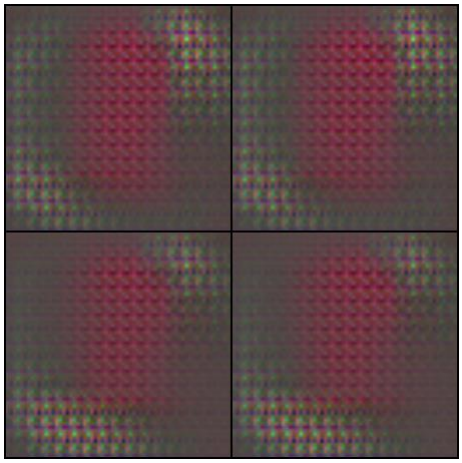
35. ábra Data1 csoporton tanult osztályozó train_loss és val_loss értékei

Ahogy ez várható volt, az eredeti adatbázis 1%-a (~15 kép/osztály) már nem elégséges, hogy a GAN (lásd 34. ábra) mintákat tudjon felfedezni, és javuló tendenciát mutatni 400 epoch alatt. Ez a modell a jelenlegi beállítások mellett jelentős underfitet mutat.

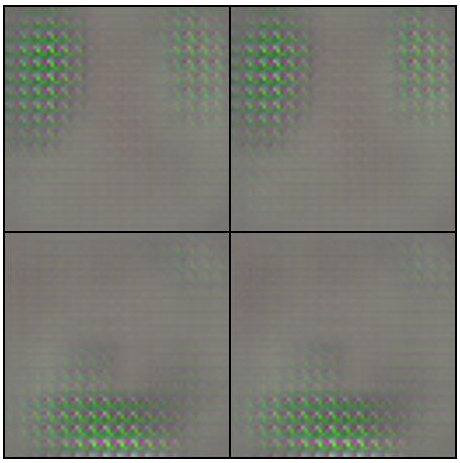
A generált képek minősége a 36-39. ábrákon látható, 100 epoch-onként.



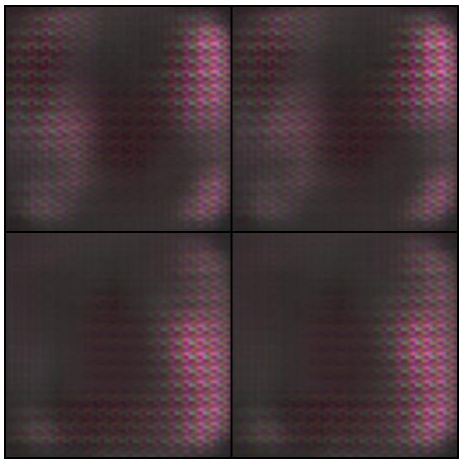
36. ábra Minták 100. epochnál



37. ábra Minták a 200. epochnál



38. ábra Minták 300. epochnál



39. ábra Minták a 400. epochnál

Az eredmények alapján úgy tűnik, hogy ennyi bemeneti adatból nem tudott jelentős mintákat feltárni sem a Generátor, sem pedig az osztályozó.

6. Teszt eredményének kiértékelése

A kiértékeléshez a kevert eredeti és szintetikus adathalmazon betanított osztályozót futtattam az eredeti tiszta adathalmazra.

Egy dataN csoport kialakítása véletlenszerű (random) mintavételezéssel történt, melyben az eredeti adathalmazból kimásoltam N% adatpontot, majd azon az N%-on betanítottam a GAN modellt, és azzal generáltam képeket, melyből visszamásoltam (100-N) % szintetikus adatot, visszatöltve az eredeti adatmennyiségre.

Így tehát a data99 az 99% eredeti és 1% szintetikus kép, a data50 az 50% eredeti és 50% szintetikuskép, míg a data15 az 15% eredeti és 85 % szintetikus kép, végezetül pedig a data1 az 1% eredeti és 99% szintetikus kép. Emellett a kontrollcsoport a data100-zal ekvivalens, mivel az 100% eredeti és 0% szintetikus képből áll

A könnyeb olvashatóságra való tekintettel az ábrázolt grafikonok a következő megjelenítést követik:

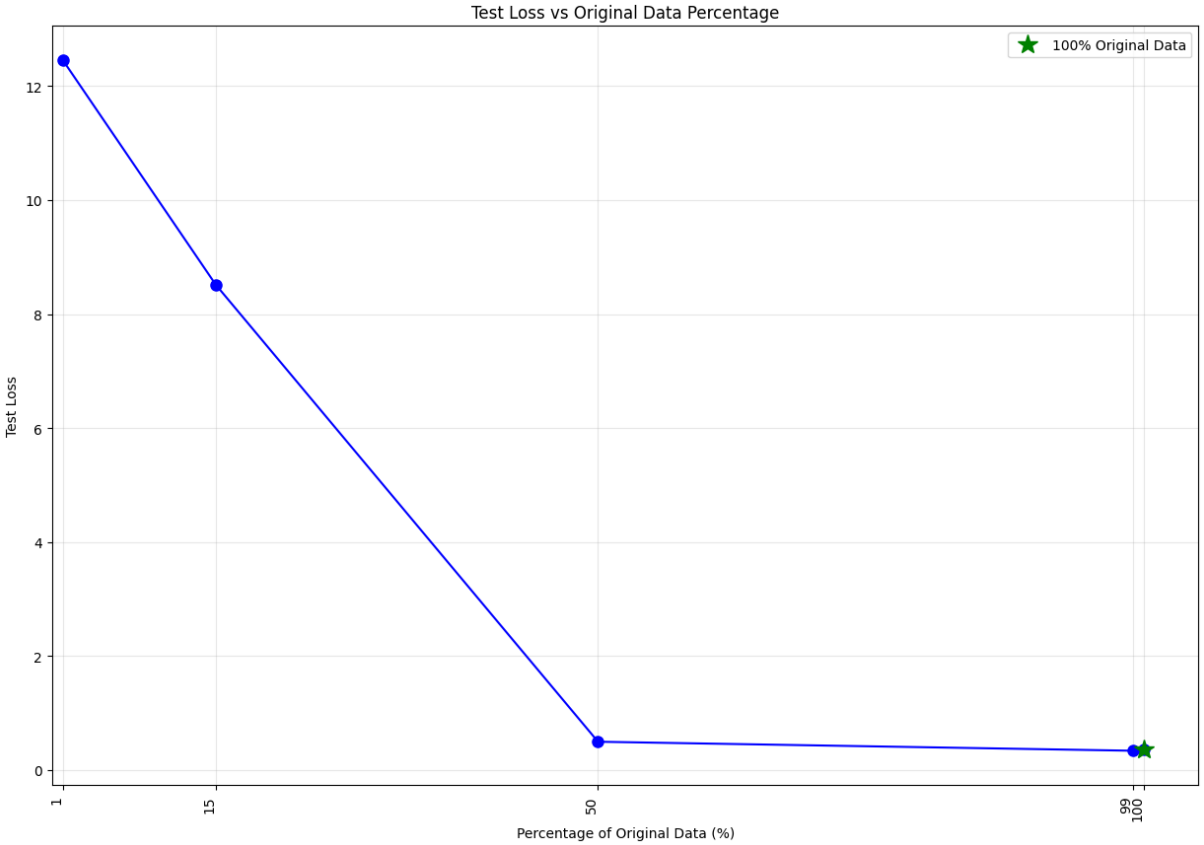
- Az X tengely mindig azt mutatja, hogy az osztályozó tanítási adathalmaza hány százalékban tartalmazott eredeti képeket. Lásd csoportok felépítése az 5. fejezetben.
- Az Y tengely mindig az aktuális metrika értékét mutatja, hogy az adott csoporton tanult osztályozó, hogyan teljesített a teszt adathalmazon (data100-as csoporton).

Loss

A következő részben megvizsgálom a test loss értékeket annak függvényében, hogy az adatok hány százaléka származott az eredeti adatbázisból. A loss érték reprezentálja a modell által adott eredmény (predikció, y'-al jelölve) és a tényleges címke (y-nal jelölve) közötti távolságot. A loss függvény ezen adathalmaz esetében egy egyszerű bináris kereszt-entrópia, mely az alábbi formula alapján kerül kiszámításra:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(y_i') + (1 - y_i) \cdot \log(1 - y_i')].$$

Az 5 adatcsoporton elvégzett kiértékelés eredménye a 40. ábrán látható.



40. ábra BCE loss értékek az eredeti adatok %-os arányának függvényében

Ahogy várható, több eredeti kép többnyire jobb eredménnyel jár. A durva törések alapján úgy tűnik, hogy a mintavételezés részletességével kapcsolatban még érdemes további vizsgálatokat végezni a jövőben, kipróbálva különböző egyéb arányokat is.

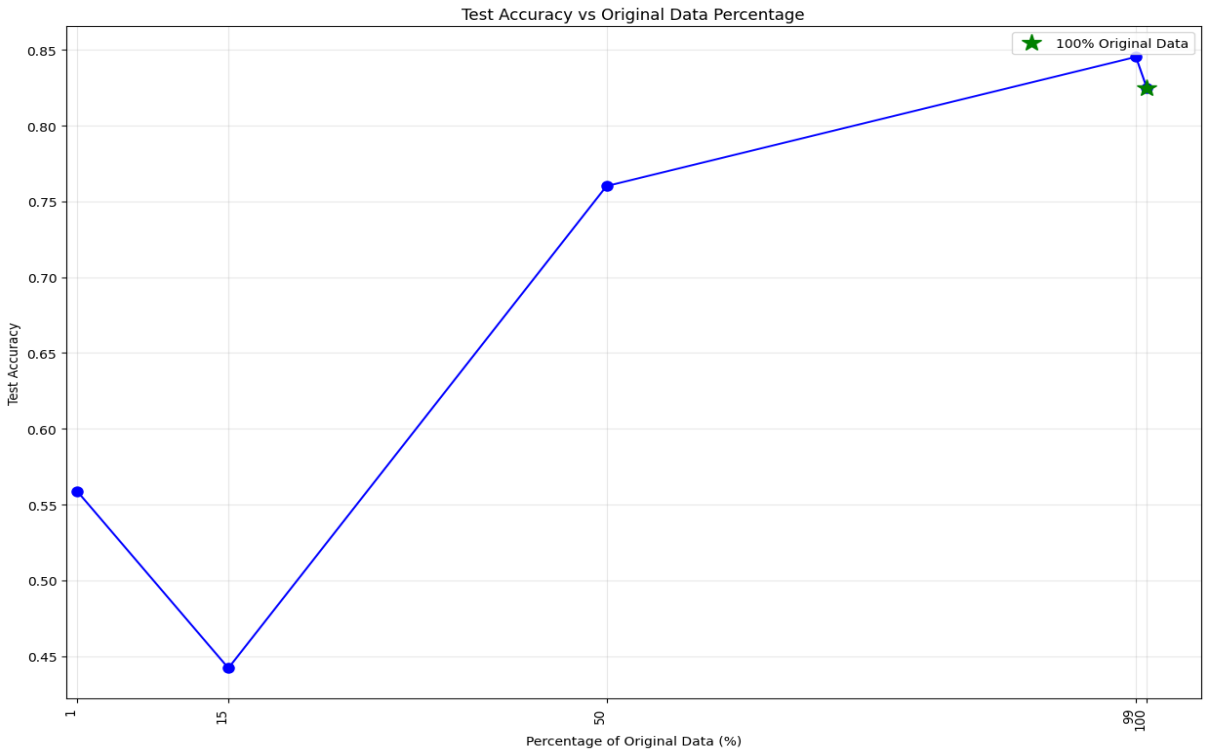
Accuracy

A következőkben a teszt adathalmaz pontosság (Accuracy) értékeit vizsgálom annak függvényében, hogy annak hány százaléka származott az eredeti adathalmazból. Az Accuracy a modell teljesítményének értékelésére használt mérőszám a helyes prediktálások (mind a valódi pozitív, mind a valódi negatív) mérésével. A metrika az alábbi képlet alapján kerül kiszámításra

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN},$$

ahol TP jelöli a True Positive (helyesen meghatározott rosszindulatú esetek száma), FP a False Positive (helytelenül rosszindulatúnak tekintett jóindulatú esetek száma), TN a True Negative (helyesen meghatározott jóindulatú esetek száma), valamint FN a False Negative (helytelenül jóindulatúnak címkézett rosszindulatú esetek száma) eredményeket.

Az 5 adatcsoporton elvégzett kiértékelés eredménye a 41. ábrán látható.



41. ábra Accuracy értékek az eredeti adatok %-os arányának függvényében

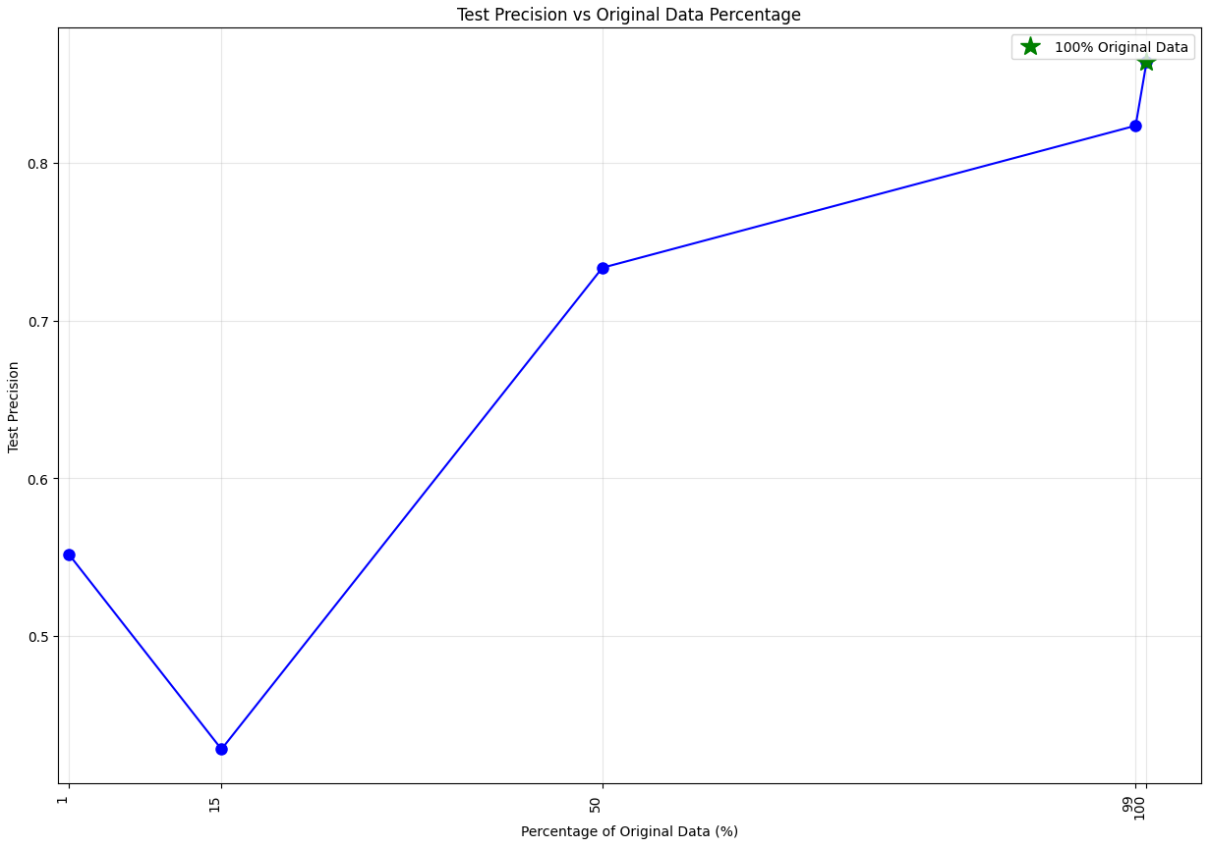
Meglepő módon az 1% eredeti adat használata nemhogy rontott, hanem még javított az accuracy-t figyelembe véve. Ez azonban az alacsony mintaszám miatt nem feltétlenül reprezentatív eredmény. Ezenfelül érdemes lehetne kivizsgálni sűrűbb mintavételezésekkel 90%-tól 100%-ig bezárólag, hogy milyen pontossági értékeket eredményeznének a betanítások. Egy ilyen, sűrűbb mintavételezéssel remélhetőleg pontosabb értékeket kaphatunk. Ez kiváltképp fontos, figyelembe véve, hogy egy jelentős törés látható az 50% környékén, mely eléggé ígéretes kutatási iránynak bizonyul.

Precision

A teszt adathalmazon kiértékelt modellek precision értékeit is vizsgáltam annak függvényében, hogy az adatok hány százaléka származott az eredeti adathalmazból. Ennek oka, hogy a Precision egy meglehetősen fontos és elterjedten használt pontossági metrika, amely értékeli a modell pozitív prediktálásának minőségét. Formulája a következő:

$$\text{Pr}\,e\,cision\ =\ \frac{TP}{TP\ +\ FP}.$$

Az 5 adatcsoporton elvégzett kiértékelés eredménye a 42. ábrán látható.



42. ábra Precision értékek az eredeti adatok %-os arányának függvényében

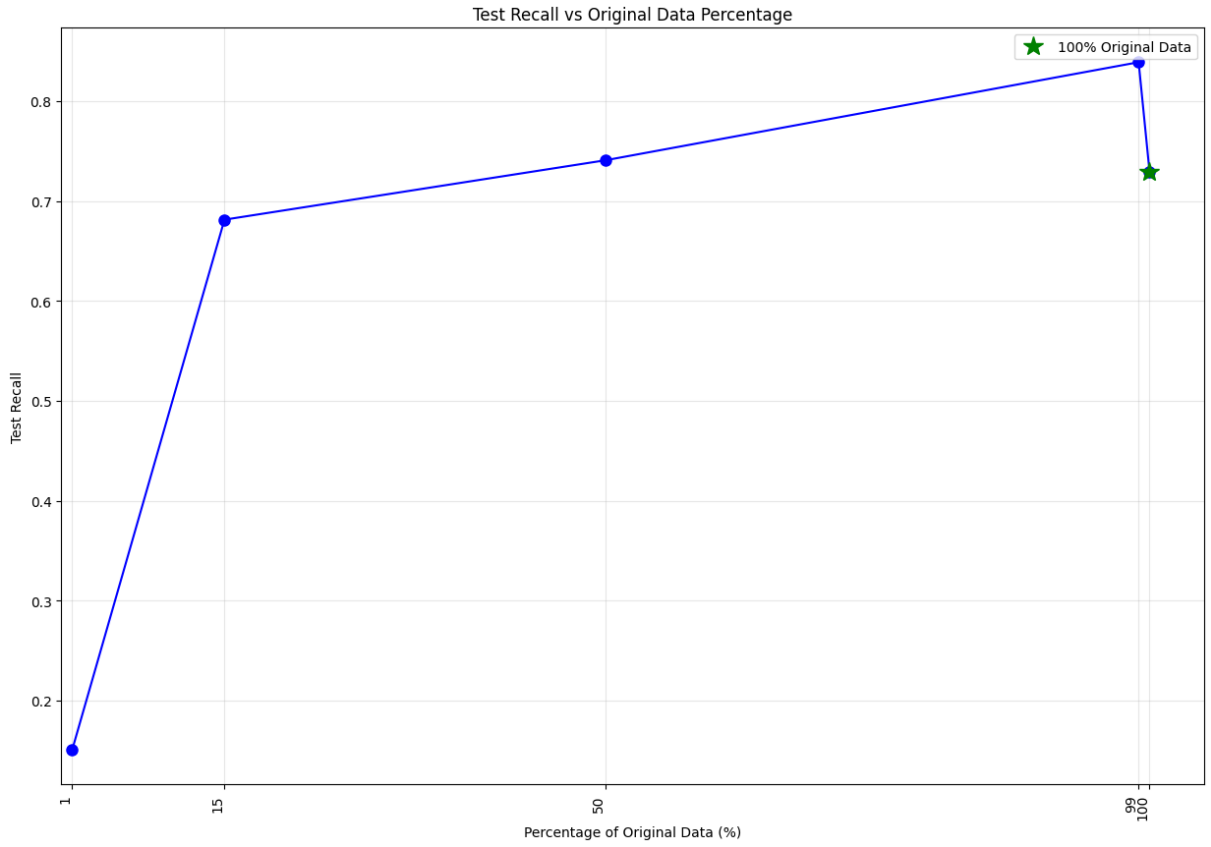
Érdekes módon a Precision-ben az 1% szintetikus adat már jelentős romlást eredményezett.

Recall

A teszt adathalmazon mért Recall értékek alakulását is vizsgáltam, annak függvényében, hogy az adatok hány százaléka származott az eredeti adathalmazból. Erre azért volt szükség, mert a Recall, mint metrika, méri a modell azon képességét, hogy helyesen azonosítsa az összes releváns pozitív példányt egy adatkészletben. Ezáltal ez szintén egy kulcsfontosságú metrika a mélytanuló modell kiértékelésére az általam választott adathalmaz esetében. A Recall-t bizonyos esetekben érzékenységnek vagy valódi pozitív aránynak (TPR) is nevezik. Formulája a következő:

$$Recall = \frac{TP}{TP + FN}.$$

Az 5 adatcsoporton elvégzett kiértékelés eredménye a 43. ábrán látható.



43. ábra Recall értékek az eredeti adatok %-os arányának függvényében

Az eredmények alapján meglepően a Recall-t figyelembe véve javított a 1% szintetikus adat hozzáadása az adathalmazhoz, hiszen a 100%-ot jelölő mérési eredmény alacsonyabb, mint a 99%-nyi adatot tartalmazó verzióé.

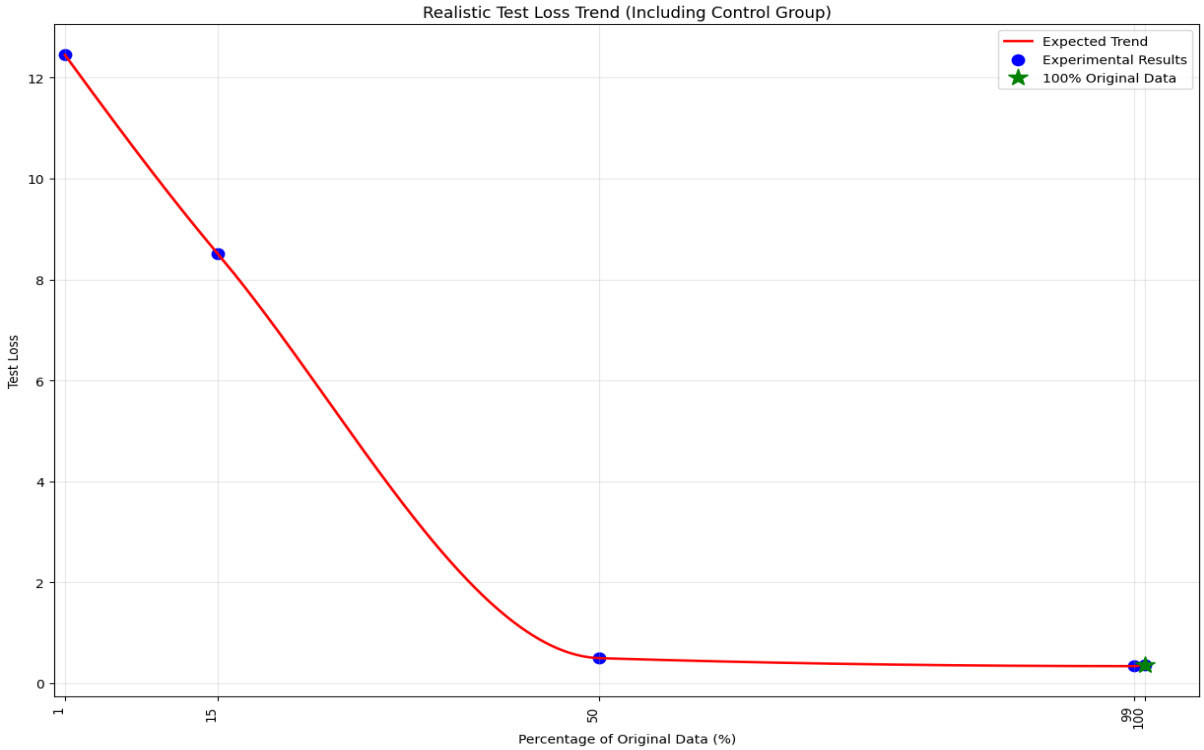
Eredmények finomítása

Mivel elég sok helyen éles törésvonalak jelentek meg a kis mintavételezés miatt, ezért használhatunk interpolálási eszközöket, hogy egy kicsit kisimítsuk a mérésekre illeszkedő görbéket. Ennek a módszernek az előnye, hogy így közelebb kerülhetünk a valódi eredményhez anélkül, hogy további költséges számítást kellene végezni.

Ilyen módszer a PCHIP - Cubic Hermite spline [20] közelítés, mely gyakran monotonitás tartó. Ennek révén nem fog a tapasztaltaknak ellentmondó görbét eredményezni, ellentétben például egy B-spline-al. A továbbiakban az ezen módszer használatával kisimított eredményeket mutatom be.

Loss PCHIP közelítése

Az 5 adatcsoporton a loss közelítés eredménye a 44. ábrán látható.



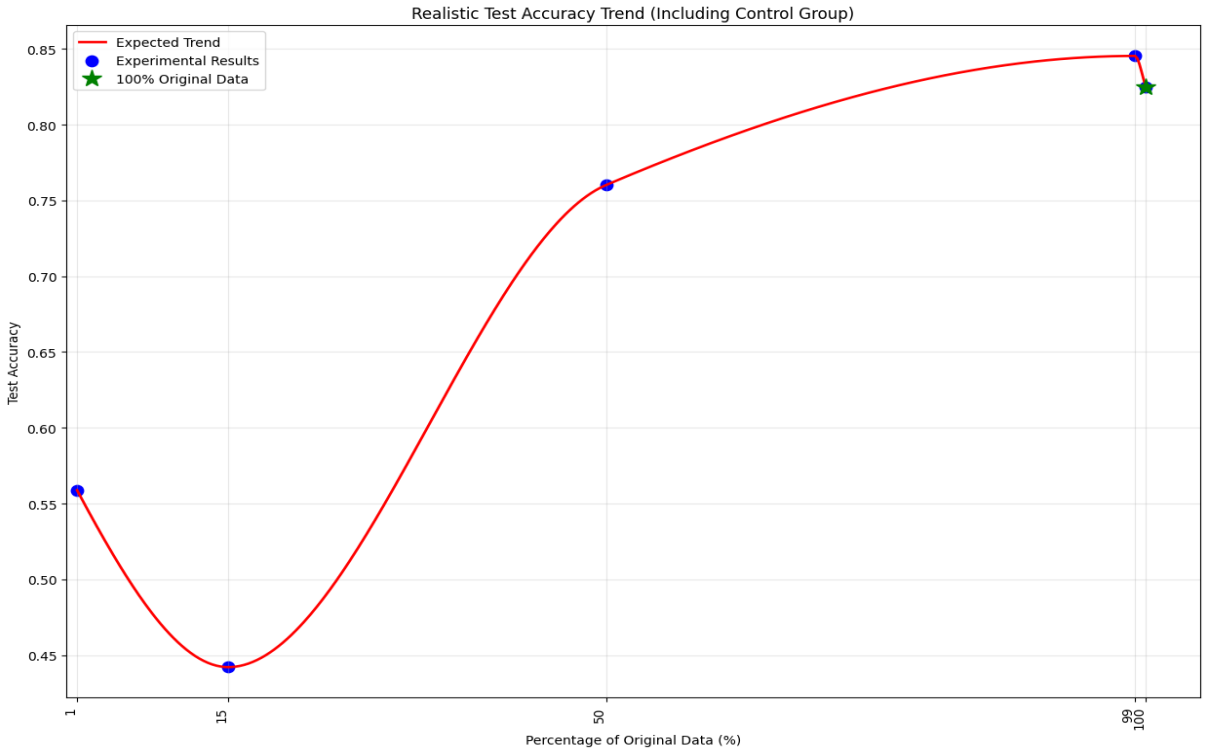
44. ábra BCE loss PCHIP közelített értékek az eredeti adatok %-os arányának függvényében

Ahogy látható, a PCHIP kisimította valamelyest az eredményeket, de még mindig elég nagy törés látszódik a 40%-50%-os intervallumon. Ez rámutat arra, hogy feltehetően érdemes lehet

ott is további mintavételezéseket készíteni, a korábban említett 90%-100%-os intervallumon felül is.

Accuracy PCHIP közelítése

Az 5 adatcsoporton az Accuracy közelítés eredménye a 45. ábrán látható.

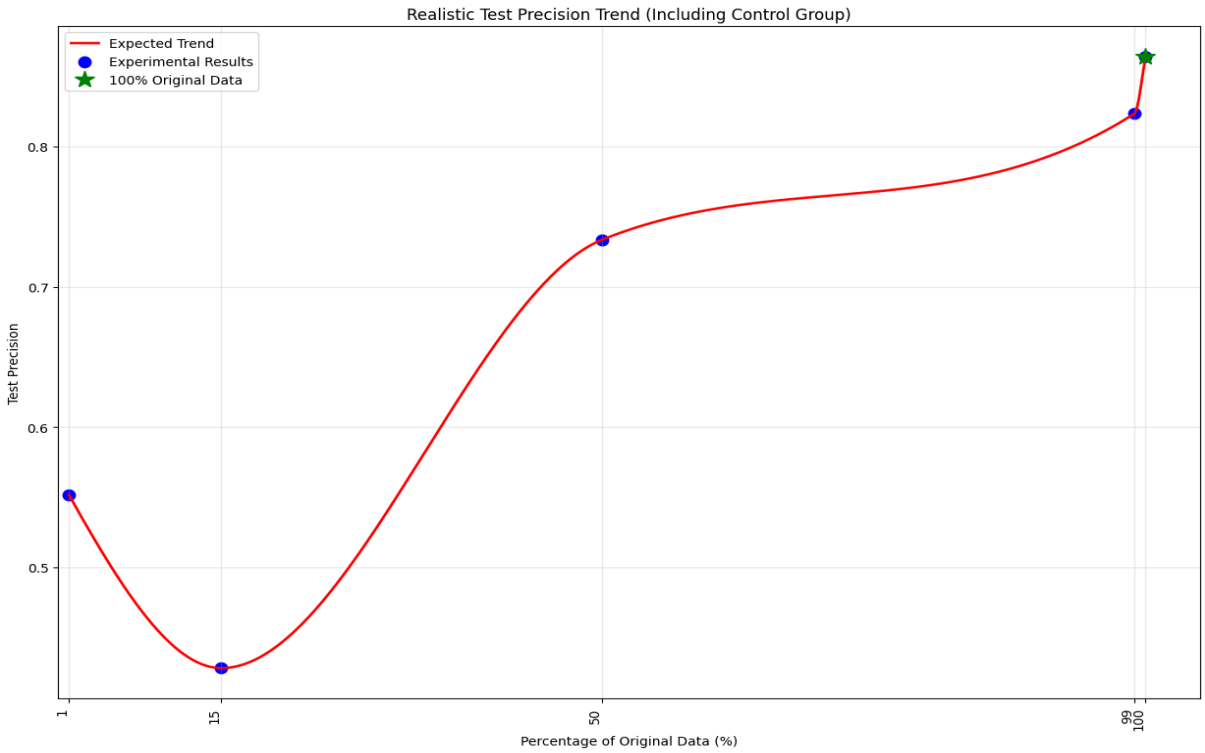


45. ábra Accuracy PCHIP közelített értékek az eredeti adatok %-os arányának függvényében

Elnézve a közelített eredményeket, feltétlenül kellene további mintavételezés a 90%-tól 100%-os intervallumra, mivel még a PCHIP-s közelítéssel is a 99%-nál éles törésvonal figyelhető meg. Ugyan máshol nem észlelhető éles törés a kisimított mérési eredményeken, azonban érdemesnek tűnik további mintavételezési pontokat felvenni a 50% alatti területen is, hogy minél pontosabb képet kapjunk a tényleges teljesítményről.

Precision PCHIP közelítése

Az 5 adatcsoporton a precision közelítés eredménye a 46. ábrán látható.



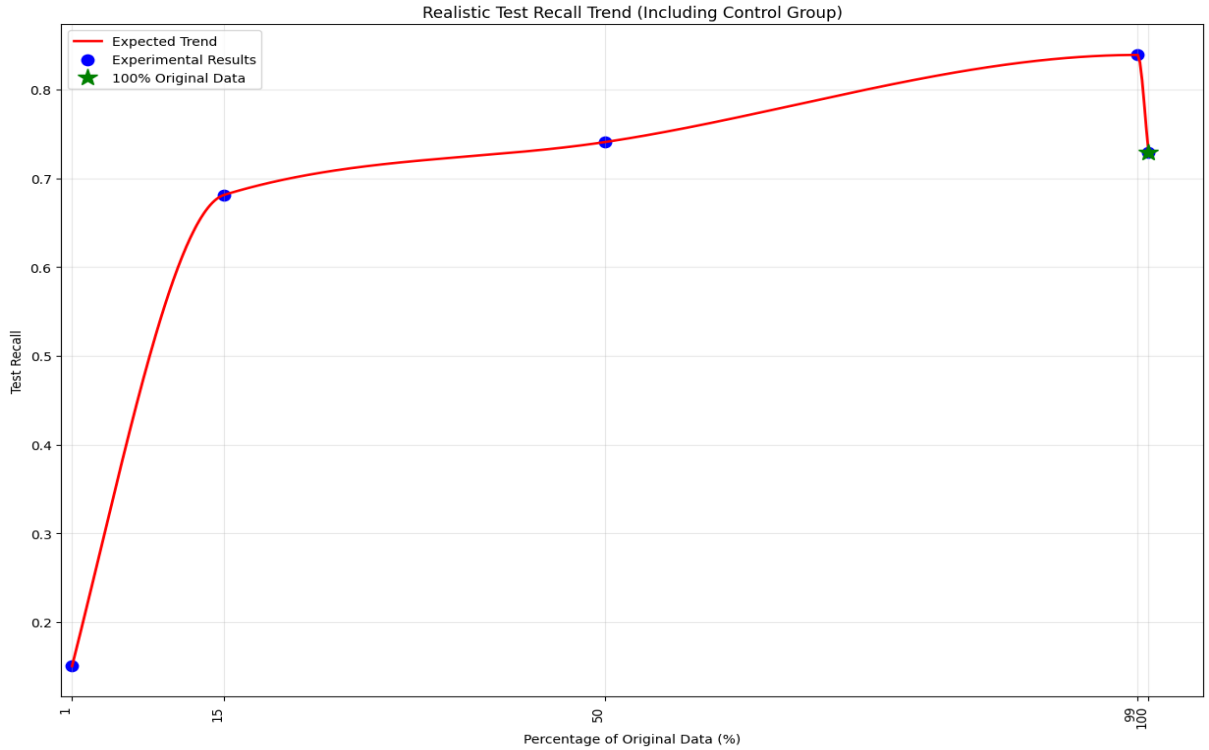
46. ábra Precision PCHIP közelített értékek az eredeti adatok %-os arányának függvényében

Megvizsgálva a közelített eredményeket, ugyancsak alátámasztásra kerül a korábban említett feltevésem, miszerint feltétlenül kellene további mintavételezés a 90%-tól 100%-os intervallumra. Ennek oka, hogy még a PCHIP-s közelítéssel is a 99%-ot reprezentáló adatpontnál éles törésvonal figyelhető meg.

Habár máshol nem észlelhető ilyen mértékű éles törés, ugyanakkor továbbra is érdemesnek tűnik további mintavételezési pontokat felvenni a 50% alatti területen, hogy pontosabb képet kapjunk az tényleges teljesítményről.

Recall PCHIP közelítése

Az 5 adatcsoporton a recall közelítés eredménye a 47. ábrán látható.



47. ábra Recall PCHIP közelített értékek az eredeti adatok %-os arányának függvényében

Elemezve a közelített eredményeket, ismételten feltétlenül kellene további mintavételezés a 90%-tól 100%-os intervallumra, mivel még a PCHIP-s közelítéssel is a 99%-nál éles törésvonal figyelhető meg. Érdekes ugyanakkor, hogy a Recall esetében egy újabb potenciálisan kivizsgálható tartomány is körvonalazódik. Nevezetesen, habár máshol nem észlelhető éles törés, még érdemesnek tűnik további mintavételezési pontokat felvenni a 20% alatti területen, hogy pontosabb képet kapjunk az tényleges teljesítményről.

8. Összefoglalás

Az eredményekből leszűrhető, hogy a kisebb méretű adatbázisok (1000-2000 kép/osztály) szintetikus dúsítása jelentős eredmény romlása nélkül tudta megnövelni a rendelkezésre álló adatmennyiséget. Még ~800kép/osztály adatbázis szintetikusan történő duplájára való dúsítása során is közel 10%-os teljesítmény béli romlás lett mérve.

További fejlesztési lehetőségként meg kell említeni, hogy habár ezen mérések kisebb modellen lettek tesztelve, érdemes lehet a jövőben nagyobb modelleken és bonyolultabb problémákon is kiértékelni ezen teszteket. Ezáltal mélyebbre hatóan is megvizsgálható lehet, hogy milyen kihatással vannak az adatbázis nagy mennyiségű növelésével más típusú, méretű, komplexitású modellek esetében.

Ilyen mélyebbre ható kutatás lehetne az, hogy a sok, már létező modellen is kiértékelhetőek lennének a kutatási eredmények. Erre nyújthat megoldást a StudioGAN [26], mely egy Pytorch könyvtár, mely közvetlen hozzáférést biztosít a már jól bevált és tesztelt GAN implementációkhoz, mind feltétel nélküli és feltételes verzióban is. A könyvtár lehetőséget nyújt különböző GAN típusok azonnali hozzáféréséhez és gyors teszteléséhez egy egységesített környezetben, hogy a generatív kutatók zökkenőmentesen tudjanak tesztelni és összehasonlítani nagyobb GAN modelleket, kezdve az egyszerűbb DCGAN-tól[41] teljesen a ADCGAN-ig[42] bezárólag. Továbbá több, már kész beépített kiértékelési módszerekhez is azonnali hozzáférést biztosít, hogy mélyrehatóan össze lehessen hasonlítani az általa tanított modelleket, felhasználva különböző metrikákat, mint Inception Score (IS) [27], vagy Frechet Inception Distance (FID) [28].

Alapvetően a StudioGAN egy eléggé robosztus és jó eszközt biztosít arra, hogy nagyobb modelleket saját magunk is teszteljünk anékül, hogy optimális hiperparamétereket kellene saját magunknak keresni. Ennek oka, hogy a src/configs könyvtárban megtalálható jól ismert adatbázisokra szabott konfigurációs paraméterek egy gyűjteménye, mint például a CIFAR10[43], CIFAR100[44], ImageNet[45], CUB200[46].

Jó magam is kipróbáltam a ImageNet/BigGAN-256-TTUR [29] és az ImageNet/SNGAN-256 modellt [30] az ImageNet egy csökkentett részére, mely csak a kutyák és macskák osztályát tartalmazta.

A StudioGAN továbbá beépített WandB(Weights and Biases) [31] integrációval jár, így élőben lehet nyomon követni a modell teljesítményét különböző metrikák figyelésével, és a súlyok monitorozásával mélyebb betekintést nyerhetünk a modell aktuális teljesítményéről és állapotáról. Az eddigi futtatásokról készített összehasonlító reprezentációt az alábbi helyen lehet megtalálni: [32].

Ebben a jelentésben összehasonlítottam a BigGAN-t[47], melyet az adatbázis 50%-án tanítottam, az SNGAN-t[48], melyet az adatbázis 50%-án tanítottam, és a BigGAN-t, mely az adatok 100%-án tanítottam.

Mint ahogy az látható, más fajta modellek teljesen más érzékenységet mutatnak az adatok mennyiségére, mivel a jelentésemből is látható, a BigGAN sokkal rosszabbul reagált az adatbázis drasztikus csökkenésére, mint az SNGAN.

Az 48. ábrán látható, hogy a BigGAN sokkal komplexebb és összetettebb minták megtanulására is képes. Ezáltal a modell képes lehet a valós adatokhoz nagy mértékben hasonló, realisztikus képek előállítására. Ennek tudatában érdemes lehet a jövőben további kutatások során más területeken is letesztelni a generatív modellek érzékenységét a modellek betanításához szükséges adat mennyiségének potenciális csökkentésére.

48. ábra Egy mintakép a BigGAN tanításáról, 1. sor a macskák osztálya, 2. sor a kutyák osztálya

9. Irodalomjegyzék

[1] Cusworth, S., Gkoutos, G. V., & Acharjee, A. (2024). A novel generative adversarial networks modelling for the class imbalance problem in high dimensional omics data. *BMC Medical Informatics and Decision Making*, *24*(1). https://doi.org/10.1186/s12911-024-02487-2

[2] Faltings, U., Bettinger, T., Barth, S., & Schäfer, M. (2021). Impact on Inference Model Performance for ML Tasks Using Real-Life Training Data and Synthetic Training Data from GANs. Information, 13(1), 9. https://doi.org/10.3390/info13010009

[3] Kumar, S., Roy, P. P., Dogra, D. P., & Kim, B. (2023, November 19). A comprehensive review on sentiment analysis: Tasks, approaches and applications. arXiv.org. https://arxiv.org/abs/2311.11250

[4] Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., & Wang, L. (2022b, May 27). GIT: a generative image-to-text transformer for vision and language. arXiv.org. https://arxiv.org/abs/2205.14100

[5] Hui, J. (2019b, January 26). GAN — A comprehensive review into the gangsters of GANs (Part 2). Medium. https://jonathan-hui.medium.com/gan-a-comprehensive-review-into-the-gangsters-of-gans-part-2-73233a670d19 (Hozzáférés dátuma: 2025. Ápr. 18.)

[6] Zhou, Z., & Firestone, C. (2019). Humans can decipher adversarial images. Nature Communications, 10(1). https://doi.org/10.1038/s41467-019-08931-6

[7] Papers with Code - R1 Regularization Explained. (n.d.). https://paperswithcode.com/method/r1-regularization

[8] Kaggle: your machine learning and data science community. (n.d.). https://www.kaggle.com/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[9] Skin cancer (ISIC Images). (2024b, September 17). Kaggle. https://www.kaggle.com/datasets/rm1000/skin-cancer-isic-images/data (Hozzáférés dátuma: 2025. Ápr. 18.)

[10] Mino, A., & Spanakis, G. (n.d.). LoGAN: Generating Logos with a Generative Adversarial Neural Network Conditioned on color. arXiv.org. https://arxiv.org/abs/1810.10395

[11] Mittal A. (2024, November 15). A diffúziós modellek megértése: mélyrepülés a generatív AI-ba. Unite.AI. https://www.unite.ai/hu/A-diff%C3%BAzi%C3%B3s-modellek-meg%C3%A9rt%C3%A9se-m%C3%A9lyrep%C3%BCl%C3%A9s-a-generat%C3%ADv-ai-ba/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[12] LightningDataModule — PyTorch Lightning 2.5.1 documentation. (n.d.). https://lightning.ai/docs/pytorch/stable/data/datamodule.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[13] RandomHorizontalFlip — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.RandomHorizontalFlip.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[14] RandomVerticalFlip — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.RandomVerticalFlip.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[15] RandomRotation — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.RandomRotation.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[16] RandomResizedCrop — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.RandomResizedCrop.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[17] Normalize — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.Normalize.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[18] ToTensor — Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/generated/torchvision.transforms.ToTensor.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[19] Chollet, F. (2016, October 7). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv.org. https://arxiv.org/abs/1610.02357

[20] Fritsch, F. N. (1982). Piecewise Cubic Hermite Interpolation Package. Final specifications. https://doi.org/10.2172/6838406

[21] ConvTranspose2d — PyTorch 2.6 documentation. (n.d.). https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[22] Upsample — PyTorch 2.6 documentation. (n.d.). https://pytorch.org/docs/stable/generated/torch.nn.Upsample.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[23] Conditional generative adversarial Nets. (n.d.). Ar5iv. https://ar5iv.labs.arxiv.org/html/1411.1784 (Hozzáférés dátuma: 2025. Ápr. 18.)

[24] Kormányos, A. (2022). ADAM optimalizációs eljárás. https://akormanyos.web.elte.hu/ https://akormanyos.web.elte.hu/teaching/halnum19la/slides2022/adam-optimization.pdf (Hozzáférés dátuma: 2022. nov. 2.)

[25] Chen, Z., Luo, T., & Wang, G. (2024, October 26). On Multi-Stage Loss Dynamics in Neural Networks: Mechanisms of plateau and descent stages. arXiv.org. https://arxiv.org/abs/2410.20119

[26] Postech-CVLab. (n.d.-b). GitHub - POSTECH-CVLab/PyTorch-StudioGAN: StudioGAN is a Pytorch library providing implementations of representative Generative Adversarial Networks (GANs) for conditional/unconditional image generation. GitHub. https://github.com/POSTECH-CVLab/PyTorch-StudioGAN (Hozzáférés dátuma: 2025. Ápr. 18.)

[27] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016, June 10). Improved techniques for training GANs. arXiv.org. https://arxiv.org/abs/1606.03498

[28] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017, June 26). GANs trained by a two Time-Scale update rule converge to a local Nash equilibrium. arXiv.org. https://arxiv.org/abs/1706.08500

[29] Postech-CVLab. (n.d.-c). PyTorch-StudioGAN/src/configs/ImageNet/BigGAN-256-TTUR.yaml at master · POSTECH-CVLab/PyTorch-StudioGAN. GitHub. https://github.com/POSTECH-CVLab/PyTorch-StudioGAN/blob/master/src/configs/ImageNet/BigGAN-256-TTUR.yaml (Hozzáférés dátuma: 2025. Ápr. 18.)

[30] Postech-CVLab. (n.d.-d). PyTorch-StudioGAN/src/configs/ImageNet/SNGAN-256.yaml at master · POSTECH-CVLab/PyTorch-StudioGAN. GitHub. https://github.com/POSTECH-CVLab/PyTorch-StudioGAN/blob/master/src/configs/ImageNet/SNGAN-256.yaml (Hozzáférés dátuma: 2025. Ápr. 18.)

[31] Weights & biases. (n.d.). W&B. https://wandb.ai/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[32] Nyika, B. (2025). BigGan data50% vs SNGAN data50% vs BigGan data100%. https://wandb.ai/nyikabenedek1212-self/uncategorized/reports/BigGan-data50-vs-SNGAN-data50-vs-BigGan-data100---VmlldzoxMjExNDU2OA?accessToken=0kccjwnw0hrlsotvtbu6041kqiur888lrzyrjhnfaz0emfj05dqlag71a6rtck32 (Hozzáférés dátuma: 2025. Ápr. 18.)

[33] Papers with Code - Improving Language Understanding by Generative Pre-Training. (2018, June 11). https://paperswithcode.com/paper/improving-language-understanding-by

[34] Kingma, D. P., & Welling, M. (2013, December 20). Auto-Encoding variational Bayes. arXiv.org. https://arxiv.org/abs/1312.6114

[35] Ho, J., Jain, A., & Abbeel, P. (2020, June 19). Denoising diffusion probabilistic models. arXiv.org. https://arxiv.org/abs/2006.11239

[36] Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), 11–15. https://doi.org/10.1145/361237.361242

[37] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. International Conference on Machine Learning, 8821–8831. http://proceedings.mlr.press/v139/ramesh21a/ramesh21a.pdf

[38] OpenAI. (2021). DALL-E. https://openai.com/index/dall-e-2/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[39] OpenAI. (2022). Chat-GPT. https://chatgpt.com/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[40] Hangzhou DeepSeek Artificial Intelligence Basic Technology Research Co., Ltd. (2023). Deepseek. https://chat.deepseek.com (Hozzáférés dátuma: 2025. Ápr. 18.)

[41] Radford, A., Metz, L., & Chintala, S. (2015, November 19). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv.org. https://arxiv.org/abs/1511.06434

[42] Hou, L., Cao, Q., Shen, H., Pan, S., Li, X., & Cheng, X. (2021, July 21). Conditional GANs with Auxiliary Discriminative Classifier. arXiv.org. https://arxiv.org/abs/2107.10060

[43] CIFAR-10 and CIFAR-100 datasets. (n.d.). https://www.cs.toronto.edu/~kriz/cifar.html (Hozzáférés dátuma: 2025. Ápr. 18.)

[44] Papers with Code - CIFAR-100 Dataset. (n.d.). https://paperswithcode.com/dataset/cifar-100 (Hozzáférés dátuma: 2025. Ápr. 18.)

[45] ImageNet. (n.d.). https://image-net.org/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[46] Perona Lab - CUB-200-2011. (n.d.). https://www.vision.caltech.edu/datasets/cub_200_2011/ (Hozzáférés dátuma: 2025. Ápr. 18.)

[47] Brock, A., Donahue, J., & Simonyan, K. (2018, September 28). Large scale GAN training for high fidelity natural image synthesis. arXiv.org. https://arxiv.org/abs/1809.11096

[48] Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018, February 16). Spectral normalization for generative adversarial networks. arXiv.org. https://arxiv.org/abs/1802.05957

10. Mesterséges Intelligencia Használatának Dokumentációja (TVSZ 22. § szerint)

1. MI alkalmazás: Képek formátumellenőrzésének automatizálása

Alkalmazott MI eszköz: Deepseek (aktuális verzió: 2024.07.09)
Dátum: 2025
Cél: A képek formátumának ellenőrzése (RGB mód, 224x24 méret).

Felhasználói input:

```
healthy_images = [f for f in os.listdir(os.path.join(path, 'benign'))]
infected_images = [f for f in os.listdir(os.path.join(path, 'malignant'))]
print(f"Number of healthy images: {len(healthy_images)}")
print(f"Number of infected images: {len(infected_images)}")
```

Specifikus kérés:
"Could you continue, that it will check all images of both healthy and infected, that they are truly RGB images of 224x224?"

MI által generált válasz:

```
import os
from PIL import Image

# ... [részletes kód a 224x224 és RGB ellenőrzéshez, lásd a teljes kódot 13-14. oldalon] ...
```

Funkcionalitás:

- A kód ellenőrzi minden kép méretét (224x24) és színmódját (RGB).
- Hibás fájlokat kategorizál és jelentést készít.

2. MI alkalmazás: TVSZ 22. §-nak megfelelő dokumentáció készítése

Alkalmazott MI eszköz: Deepseek (aktuális verzió: 2024.07.09)
Dátum: [2025.04.08.]
Cél: A TVSZ 22. § szerinti strukturált dokumentáció kialakítása, beleértve a jelen párbeszédet is.

Felhasználói input:

"Kérlek tüntesd fel a jelenlegi párbeszédet is [...] feleljen meg a 22. §-nak. Legyen egyértelmű, hogy téged használtam."

MI által generált válasz (jelen dokumentum):

- Strukturált formátum a TVSZ követelményei szerint.
- Mindkét MI-interakció leírása (képellenőrzés és dokumentáció).
- Transzparens feltüntetés az alkalmazott MI eszközről (Deepseek, verzió, dátum).

TVSZ 22. §-nak való megfelelés:

1. **Segédeszközként való használat (22§ (2)a):**
    a. Az MI nem helyettesítette a saját kutatómunkámat, csak technikai segítséget nyújtott (kódgenerálás, formázás).
2. **Transzparencia (22§ (2)c):**
    a. Az alkalmazott eszközök (Deepseek), verziók és a pontos felhasználási mód dokumentálva lettek.
3. **Felelősségvállalás (22§ (2)b):**
    a. A generált kódot és dokumentációt a szerző ellenőrizte, validálta.
4. **Dokumentációkötelesség (22§ (4)):**
    a. Az MI által generált tartalmak és a felhasználói kérések egyértelműen elkülönítve szerepelnek.

Nyilatkozat:
A fent dokumentált MI-használat során betartottam az Informatikai Kar TVSZ 22. §-ában foglalt etikai és technikai követelményeket. A munkámban alkotói szerepem elsődleges, az MI eszközök kizárólag segédletekként szolgáltak.

Nyika Benedek, 2025.04.17.

Köszönetnyilvánítás

Szeretnék köszönetet mondani konzulensemnek Dr. Bogacsovics Gergőnek, hogy fáradhatatlanul segítette és irányította a munkámat. Köszönöm türelmét és szakmai segítségét!

Végül, de nem utolsósorban szeretném megköszönni családomnak, hogy kitartóan támogattak eddigi egyetemi éveim alatt.