Képgenerálás diffúziós modellel

Mélytanulás Házi feladat

Csik Laura – Z09RRY

Zsáli Zsombor – A6HFRW

1 Bevezetés

Az elmúlt évtizedben a generatív mesterséges intelligencia modellek nagy fejlődésen mentek keresztül. Köszönhető ez annak is, hogy az ilyen jellegű modellek használatára a társadalomban is egyre nagyobb igény mutatkozik. Az elmúlt években a hírekben és a szociális média portáljain is gyakran jelentek meg ilyen eredményekkel kapcsolatos írások, amelyeket a lelkes technológia iránt érdeklődők hamar fel is fedeztek. Számos példa volt a weben megjelent publikussá és kipróbálhatóvá váló alkalmazásokra, legyen szó festmények, arcok vagy éppen csak véletlenszerű képek generálásáról

Egy viszonylag új és feltörekvő irányzat a diffúziós modellek használata képgenerálásra azóta, hogy a DDPM-ek elkezdtek elterjedni a tudományos világban (Ho, Jain, & Abbeel, 2020) (Nichol & Dhariwal, 2021) (Dhariwal & Nichol, 2021). Ezen modellek vizsgálata nagyon célravezető lehet, hiszen az architektúrájuk sokszínűségének kihasználásával nagyon komoly eredmények érhetőek el velük.

2 Adat

A feladat megoldásához két adathalmazt használtunk, a CelebA-t, illetve a Danbooru Faces-t. Ezáltal meg tudtuk vizsgálni, hogy ugyanaz a modell képes-e két nagyon eltérő, de arcokat tartalmazó adattal hasonló eredményeket elérni.

A CelebA egy híresség arcáról készült képeket tartalmazó adatbázis (Li, 2017). Mivel több, mint 200.000 képet tartalmaz, így nem tudtuk ellenőrizni, hogy mindegyiknek megfelelő-e a minősége. A képek eredetileg 218x178 méretűek, ezeket átméreteztük 80x64-re, hogy a modellnek könnyebb dolga legyen a tanulás során.

A Danbooru eredetileg anime képkockákat tartalmaz, de ebből készült egy olyan adatbázis, melyben már csak arcok láthatóak (An, 2020). Érdekesnek gondoltuk, hogy milyen minőségbeli különbégek fognak fellépni akkor, ha nem valós emberi arcokat kell generálni, hanem rajzoltakat.

Ezzel az adathalmazzal olyan probléma merült fel, hogy minden kép nagyon különbözött a másiktól, hiszen egy anime nagyon színes, mindegyiket máshogyan rajzolják meg. Emiatt várható, hogy a modell kevésbé fog jól teljesíteni.

Átválogatásra ebben az esetben sem volt lehetőség, több, mint 300.000 kép állt rendelkezésre. Ezek a képek 512x512 méretűek voltak, ebben az esetben is átméreteztük őket 64x64-re.

2.1 Adatbetöltés

Az adatok betöltésére egységesen a pytorch által nyújtott függvények adtak lehetőséget, így a torchvisionből átvett Imagefolder illetve Transform segítségével alakítottuk át a képeket a megfelelő formába. Ezen felül a Dataloader, illetve a random_split() használatával töltöttük be őket a modellnek, mint tanító és validációs halmazok. A tesztelésre szánt képek generatív modell miatt más formában kerültek elkülönítésre.

3 Modell

3.1 Diffúziós modellek

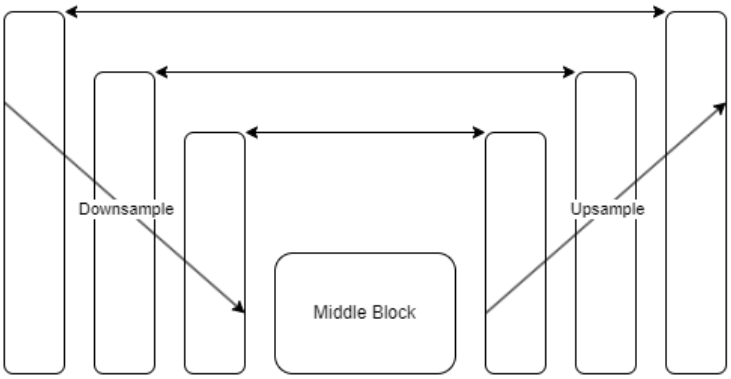
A diffúziós modellek működése alapvetően egy nagyon egyszerű gondolaton alapul (nagyon bonyolult matematikai háttérrel), amelyet két fázisra oszthatunk: Forward diffusion és Backward diffusion.

3.1.1 Forward diffusion

Az első fázisban az adott tanító adathalmazbeli képhez a modell több lépésben (esetünkben 1000) zajt ad, amíg az egy rögzített szórású 0 várható értékű Gauss zajjá nem válik.

3.1.2 Backward diffusion

A második fázisban a modell a zajosított képen próbálja meg megtippelni a zaj mértékét és iterációnként eltávolítani azt, amíg az eredeti képet megfelelőjét vissza nem kapja. Ennek megvalósításához egy speciális neurális hálót használunk. A U-Net a nevét az alakjáról kapta, ugyanis lényege az, hogy konvolúciós rétegek használatával a képet néhány lépésben egyre kisebb felbontásúra, valamint egyre többcsatornásra transzformálja, majd egy pontot elérve a másik oldali rétegeivel visszaalakítja a bemenettel megegyező formájúra.



3.1.3 Reziduális kapcsolatok

Egy másik fontos része a diffúziós modell U-Net-jének, a reziduális kapcsolat a megfelelő rétegek között. Ez lényegében azt jelenti, hogy a háló egyik oldali rétege a másik oldali megfelelőjével megosztott súlyokkal rendelkezik, ezzel növelve a zajpredikció pontosságát.

3.1.4 Attention rétegek

Bár nem alapvető része a DDPM-eknek az attention réteg, mégis sokat segít a modell teljesítményén Transformer architektúrák alapkövének számító Attention rétegek használata. (Vaswani, és mtsai., 2017)

4 Tanítás

A modell tanítása docker környezetben történt, a megfelelő NVIDIA GPU architektúra használata mellett. A training konténerbe felmásolt forrásfájlok tartalmazzák a modell teljes egészét, valamint a modellre épített Trainer osztályt, amelynek metódusai összefogják a modell inicializálási, tanítási, validációs, illetve tesztelési lehetőségeit is, rugalmas kereteket adva a felhasználó számára a megfelelő hiperparaméterek beállításához.

4.1 Inkrementális modellfejlesztés

A modell fejlesztésére alapvetően kétféle lehetőség áll rendelkezésre: a modell, különösképpen a U-Net architektúrájának egyre komplexebbé tétele, valamint a hiperparaméterek (learning rate, batch_size, modell zajosítási iterációi, konvolúciós rétegek paraméterei stb.)

4.1.1 Baseline modell

Alapvető modellként az interneten talált nagyon egyszerű implementációból indultunk ki (DeepFindr, 2022.). Ezt kezdtük el fejleszteni komplexitás hozzáadásával, valamint a hiperparaméterek hangolásával egyéb interneten is fellelhető információk és ötletek felhasználásával.

4.1.2 Attention rétegek bevezetés

A következő nagy lépés az Attention rétegek bevezetése volt a modellünkbe, ami már egy viszonylag nagy fejlesztésnek minősül, hiszen a modell alapvető architektúrája is bonyolult, az Attention blokkok hozzáadása pedig tovább növeli a komplexitást.

4.1.3 Hiperparaméterek hangolása

A hiperparaméterek finomhangolása manuálisan történt. A modell nagy erőforrás- és időigénye miatt nem volt lehetőség a modell paramétereinek mélyreható hangolását tesztelni, így az automatikus hangolás nem jöhetett szóba.

4.2 Tanítás folyamata

Egy ekkora modell betanítása hatalmas idő- és erőforrásigénnyel bír, amelyet komoly CPU-val, GPU-val és memóriával rendelkező infrastruktúrához való hozzáférés nélkül nagy nehézséget jelenthet megvalósítani.

A tanítás egyszerűen megvalósítható azonban abból a szempontból, hogy az implementált Trainer osztály függvényeinek egyszerű meghívásával lebonyolíthatjuk a teljes tanítási folyamatot, ahogy a házi feladatban látható Jupyter Notebookban is megvalósul:

- adatbeolvasás
- Trainer példányosítása
- Trainer inicializálása a megfelelő függvényekkel, amennyiben szükséges
- training függvény meghívása.

5 Kiértékelés

Ahogyan az előbbi fejezetben is említésre került a modell nagysága miatt, nehézségek merülnek fel az egyes modellek kiértékelése szempontjából, így a már említett 3 modellállapot mentén mutatnánk be a kiértékelések eredményeit.

5.1 Metrikák

5.1.1 FID – Frechét Inception Distance

A generatív modellek, azon belül is a képszintetizációs modellek egyik legtöbbet használt mérőszáma a FID. Az FID score lényegében két egyenlő elemszámú képhalmaz között hivatott meghatározni távolságot. Ezt úgy teszi, hogy egy matematikai képlet alapján a két halmaz képeinek egy valószínűségi eloszlását számolja ki, azaz várható értéket és szórást. Majd a két halmaz eloszlásainak távolságát adja vissza., azaz minél kisebb értéket kapunk, annál jobb.

5.1.2 Loss

Egy mesterséges intelligencia feladatnál általában egyértelmű, de megemlítenénk, hogy a modellek pontosságát a training, illetve főként a validation loss is meg tudja mutatni. Mivel a loss függvény minimalizálása a cél a modell tanításakor, így lényegében minél alacsonyabb értéket kapunk, annál jobb.

Természetesen figyelni kell arra, hogy a training loss nagyon alacsony értéke, a validation loss magas értéke mellett overfittingre enged következtetni, ezért fontos a modellek teljesítményének megfigyelése.

5.2 Modellek

5.2.1 Baseline

A Baseline modell esetében sajnos mind a validációs loss, mind pedig az FID score érték még egészen magas volt, ami a generált képeken is meglátszódott.

5.2.2 Attention rétegek

Az Attention rétegek bevezetése, így a modell komplexitásának növelése sikeresen javított a modellen. Természetesen az Attention rétegek bevezetése új hiperparamétereket is hozott magával, mivel a modellben paraméterezhető lett, hogy mely helyeken legyen egyáltalán Attention réteg, a U-Net konvolúciós rétegei után kapcsolva. Az FID score és a validációs adathalmazon vett loss is csökkent.

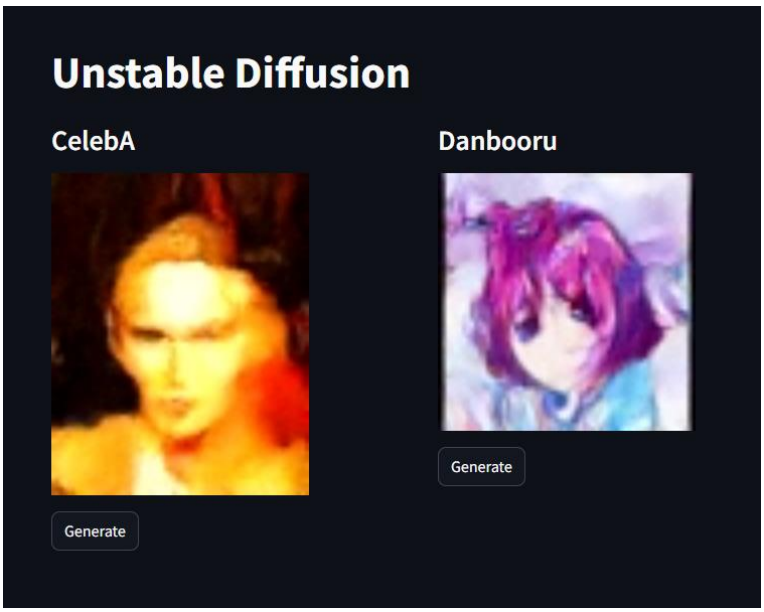
5.2.3 Végső modell

A modell komplexitását már nem növeltük, azonban a paraméterek finomhangolásával sikeresen tudtunk előidézni némi javulást a metrikákban, mint:

- learning rate csökkentése
- Attention rétegek változtatása
- konvolúciós rétegek paramétereinek változtatása
- iterációs lépések számának növelése

	CelebA			Danbooru		
	FID	**Val loss**	**Train loss**	**FID**	**Val loss**	**Train loss**
Baseline	43.4380	0.0211	0.0159	33.8833	0.0279	0.0307
Middle	32.5785	0.0158	0.0120	25.3052	0.0208	0.0229
Final	**27.1487**	0.0133	0.0099	**21.4451**	0.0178	0.0193

6 Vizuális megjelenítés



A modellhez készítettünk egy webes alkalmazást, ahol gombnyomásra lehet új képeket generáltatni a modellel mindkét adathalmazhoz. Ennek a megvalósításához a Streamlit Python könyvtárat választottuk.

A felület elkészítése nem volt bonyolult, a Streamlit előre elkészített vizuális komponenseket nyújt a fejlesztők számára. A címsor alatt két oszlop tartalmazza a két adathalmazhoz tartozó generálási felületet. Ha a felhasználó megnyomja a ’Generate’ gombot, akkor a modell generál egy képet és az megjelenik.

7 Konklúzió

Az elmúlt évek nagy előrehaladása a diffúziós modellek területén mindenképpen kiemelkedő hatással van a generatív mesterséges intelligencia modellek fejlődésére. Kutatásuk fontos lesz az elkövetkezendő években is, mivel a megjelenésük óta eltelt rövid idő alatt is nagyon sok korábbi modellt maguk mögé utasítottak.

A félév során implementáltunk egy diffúziós modellt, majd ezt fejlesztve, extra komplexitást a neurális háló architektúrájához adva, a hiperparamétereket finomhangolva jutottunk el a végső megoldásunkhoz.

A megoldásunkat teszteltük a teljes inkrementális modellfejlesztés alatt, így látható előrelépést tapasztaltunk a modell pontosságában, generalizáló képességében, hatékonyságában.

8 Hivatkozások

An, S. (2020). *Kaggle*. Letöltés dátuma: 2023. 12 09, forrás: https://www.kaggle.com/datasets/subinium/highresolution-anime-face-dataset-512x512

DeepFindr. (2022.). *Diffusion models from scratch in PyTorch*. Forrás: https://www.youtube.com/watch?v=a4Yfz2FxXiY

Dhariwal, P., & Nichol, A. (2021). *Deiffusion Models Beat GANs on Image Synthesis.*

Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models.* Forrás: https://arxiv.org/pdf/2006.11239.pdf

Li, J. (2017). *Kaggle*. Letöltés dátuma: 2023. 12 09, forrás: https://www.kaggle.com/datasets/jessicali9530/celeba-dataset

Nichol, A., & Dhariwal, P. (2021). *Improved Denoising Diffusion Probabilistic Models.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need.* Forrás: https://arxiv.org/pdf/1706.03762.pdf