Image generation with diffusion models

Képgenerálás diffúziós modellekkel

Mélytanulás nagy házi feladat dokumentáció

Bevezetés

Az elmúlt néhány évben egyre inkább előtérbe kerültek a képgenerálásra szolgáló mesterséges intelligencia modellek. Az olyan generatív modellek, mint a Stable Diffusion vagy a DALL-E a mindennapi felhasználóknak is hasznos vagy szórakoztató eszköz lehet például illusztrációk vagy mémek gyártásához. A manapság használt képgeneráló modellek többsége diffúziós modell, azon belül is DDPM (Denoising Diffusion Probabilistic Model), és ebben a házi feladatban mi is egy hasonlónak a megalkotását próbáltuk ki.

Egy diffúziós modell, szemben például egy VAE-vel (variációs autoenkóder) egy egyszerű eloszlású zajt konvertál képpé, jelen esetben fokozatosan zajtalanítva a képet, teljes zajból kiindulva. A DDPM lényegében egy parametrizált Markov-lánc, amelyet két irányban járhatunk be. Az “előre” irányú diffúziós folyamat (q) apránként normál eloszlású zajt ad a képhez, míg a visszairányú folyamat (p_θ) lépésről lépésre zajtalanítja a képet. Előbbi egy előre meghatározott, utóbbi egy tanult folyamat.

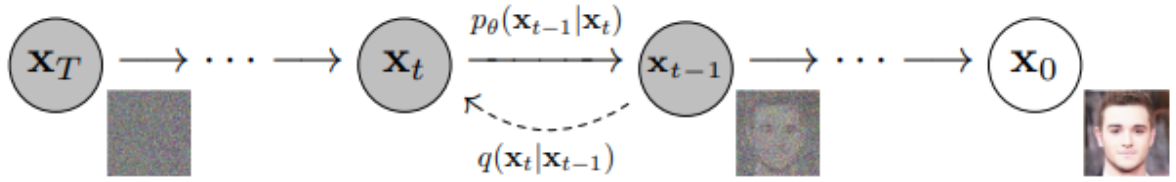


Figure 2: The directed graphical model considered in this work.

1. ábra: a DDPM folyamatábrája, forrás: https://arxiv.org/pdf/2006.11239

Az egyes állapotokat egy t változóval indexelik, amely 0-tól T-ig fut, az x_t pedig egy zajos kép, ahol x_0 a tiszta kép, x_T pedig a tiszta zaj.

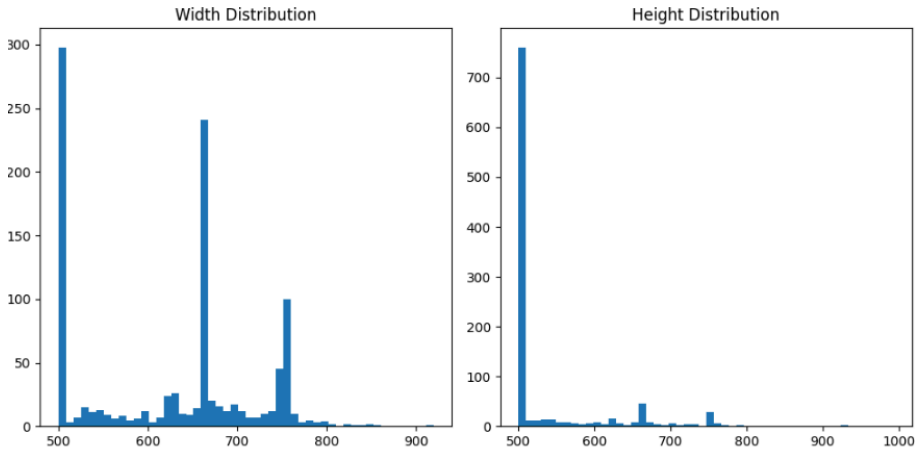
Ahhoz, hogy a visszirányú folyamathoz legyen egy használható veszteségfüggvényünk, a tanító adathalmaz képeinek komplex eloszlását egy normál eloszlással közelítjük (ez kapóra jön, hiszen a hozzáadott zaj is normális eloszlású). Egy normális eloszlást a várható értéke és a szórása jellemez, az általunk alapul vett tudományos munka (Ho és társai, 2020.) azonban úgy találta, hogy elegendő, ha a szórást rögzítjük, és így a neurális háló csak a várható értéket tanulja.

Ez a várható érték átparametrizálható, hogy a háló valójában azt tanulja megbecsülni, hogy egy zajos képnél egy adott t érték mellett mi maga a zaj. A tippelt és valódi normál eloszlású zaj között pedig egy egyszerű MSE (átlagos négyzetes hiba) segítségével határozható meg a különbség, és így ezt a veszteségfüggvényt optimalizálhatja a háló.

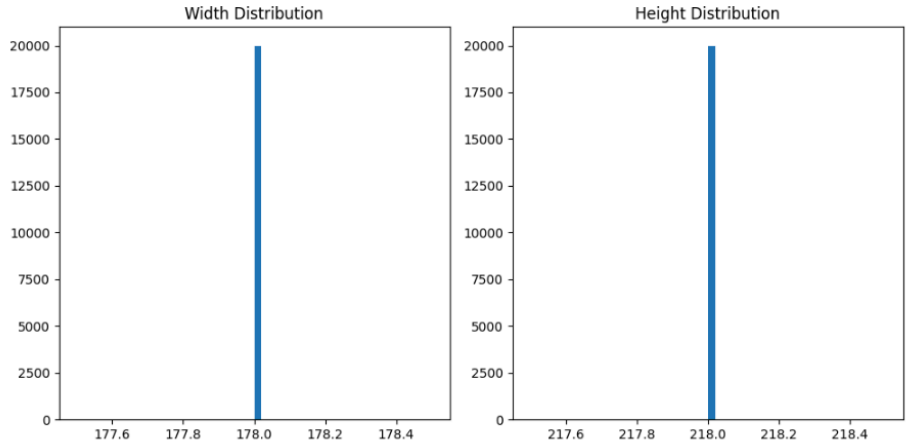
A háló kimenete tehát egy ugyanakkor méretű tenzor (“kép”), mint a bemenete, ebben pedig hasonlít egy variációs autoenkóderre. Így adja magát, hogy a háló felépítésében is legyenek hasonlóságok. A tudományos cikkben egy U-Net nevű architektúrát használnak, amely ugyancsak használ bottleneck-et, vagyis egy alacsonyabb dimenziószámú rejtett réteget, amely a képek legfontosabb információit tartalmazza. A háló először leskálázza erre a méretre az adatot, majd a végén felskálázza. Az általunk kiindulási alapként használt blogposzt emellett a Transformer architektúrában használatos Attention-rétegeket is alkalmaz.

Adatfeldolgozás

A feladatleírás alapján a *Flowers102* és a *CelebA* adathalmazokat felhasználva tanítottuk, és teszteltük a modelljeinket. Az adathalmazok betöltése után ellenőriztük azok tulajdonságait és tisztítást végeztünk rajtuk.



a) Flowers102



b) CelebA

2.ábra: Adathalmazokban található képek méretei

Mint az 1. ábrán látható a *Flowers102-ben* található képek nem egységes méretűek, amíg a *CelebA-ban* találhatóak igen. Tehát már innen látható, hogy további adatfeldolgozás szükséges. A képeket egységes méretre hozzuk, augmentáljuk (horizontal flip), tenzorrá alakítjuk és normalizáljuk. Ennek eredményeképp tehát mindkét adathalmaz betöltése során

azonos tulajdonságokkal bíró képeket kapunk. Ezt követően, mivel az adathalmazok jó minőségűek ezért nincs sok tisztításra szükség. Tisztítás során a tenzorok értékeit ellenőrizzük, akkor felel meg egy tenzor ha egyik értéke sem *inf,-inf* vagy *nan.*

Az adatfeldolgozás és tisztítás után a következő lépésben felbontjuk mind a kettő adathalmazt tanító, tesztelő és validáló részre, majd ezeket lementjük a *"./data/prepared_datasets"* mappába. Innen lesznek majd kiolvashatóak a modellek számára.

Architektúra

A modellezéshez a már említett blogposzt architektúráját használtuk fel, illetve implementáltuk újra. Az architektúra egy fontos eleme a reziduális kapcsolat, amely segítségével jóval könnyebben konvergálhat a háló, azáltal, hogy bizonyos rétegeket átugorhat. Erre épít a háló egyik alap építőköve, a ResNet blokk, amely két, egyenként egy konvolúciós rétegből, egy normalizálásból és egy SiLU aktivációs függvényből álló blokkot tartalmaz. Ezekből a második blokkot ugorja át a reziduális kapcsolat segítségével, végül a két értéket összeadja. Ennek alternatívája a ConvNext blokk, amely az azt kidolgozó cikk szerint modernizálja a ResNet-et. Végül ez utóbbi lett az alapmérezetezett, és két konvolúciós réteget tartalmaz normalizálással, GELU aktivációval, és végül reziduális kapcsolattal.

Ezen kívül a Tranformer architektúrából ismert Attention rétegek is kerültek a konvolúciós blokkok közé. Ennek két variánsa is megjelenik. Az egyik egy hagyományos multi-head self-attention, a másik egy lineáris variáns, amely időben és memória használatban nem négyzetesen, hanem lineárisan skálázódik. Az Attention rétegek előtt egy (csoportos) normalizálás is helyet kap.

Ezekből az építőelemekből áll elő az U-Net háló. Először egy konvolúciós réteg, majd (a paramétereknek megfelelő mennyiségű) leskálázó blokkok következnek, amelyek két konvolúciós blokkból, egy normalizált és reziduális kapcsolattal ellátott lineáris attention rétegből és egy leskálázó műveletből állnak. A háló közepén, a bottlenecknél két konvolúciós blokk fog közre egy hagyományos attention réteget. A felskálázó blokkok hasonlóan működnek a leskálázóhoz (leszámítva a végén a műveletet), és legvégül egy újabb konvolúciós réteg foglal helyet a kimeneti oldalon.

A háló egészében opcionálisan használható az időbeli beágyazás (time embedding), amely segítségével a háló megkapja az információt, hogy éppen milyen zajszinten tanul. A t változót a blogposzt javaslata alapján sinusos pozíciós beágyazással kódoltuk.

Variánsok

A megvalósított architektúrához különböző variánsokat is készítettünk. Az egyik modell attention-rétegek nélküli, egy másik pedig nem előre, hanem utólag normalizált attention rétegeket tartalmaz (a blogposzt szerint nincs teljes konszenzus, hogy mi a normalizálás és

az attention helyes sorrendje). Emellett a különböző aktivációs függvényeket is kipróbáltuk, lett egy-egy modell csak ReLU, csak SiLU és csak GELU aktivációs függvényekkel (a lineáris aktivációk a helyükön maradtak). Egy további variáns pedig a ConvNext blokkok helyett a ResNet blokkok használata volt.

Tanítás

A tanításhoz először az előre irányú folyamatot, vagyis a zaj hozzáadását kellett megvalósítani. Ehhez különböző szórás-ütemezést használhatunk, amely például lineáris, négyzetes, cosinus- és szigmoid-függvény alapján működnek. A blogposzt egy cikkre hivatkozva a cosinus-változatot javasolja, de próbálkozásaink alapján a Flowers képein sokkal rosszabb minőségű képeket eredményezett, mint a lineáris, így az utóbbit használtuk.

Ezután a zaj mintavételezéséhez és a visszairányú folyamathoz szükséges segédváltozók kiszámítása következik, majd ezek segítségével valósul meg a két folyamat. Az előreirányú zajt mintavételez, majd fokozatosan a bemenethez adja, míg a visszairányú fokozatosan eltávolítja a zajt. (Egyébként míg utóbbi egy ciklusban történik, a zaj hozzáadása egy adott t időpillanathoz egy lépésben is megtörténhet).

A tanító ciklus viszonylag standard módon működik: megadott epoch-számig batch-enként végigmegy az adaton, majd loss-t számol és optimalizál. A loss kiszámításához először zajt ad hozzá, becslést ad a zajra, majd a veszteségfüggvény adja meg a tippelt és valós zaj eltérését. A veszteség függvények közül a SmoothL1Loss-t alkalmaztuk, amely az MSE-hez képest kevésbé érzékeny a kiugró elemekre (és amely egy pszudo-Huber loss-nak felel meg).

A tanítás után pedig megtörténhet a képek generálása: tiszta zajból kiindulva egy ciklussal apránként egyre tisztább kép alakul ki, amíg t=0-nál meg nem születik a végleges kép.

Hiperparaméter-optimalizálás

A projekt során manuális hiperparaméter-optimalizálást végeztünk, ugyanis egy teljes körű automatikus optimalizálás nagyon költséges és időigényes lett volna. Az egyik fontos szempont a batch-ek mérete volt, úgy találtuk, hogy a 32-es adja a legjobb eredményeket, 64-es és 128-as batch-méret mellett kevésbé voltak élethűek a képek. Az optimalizációra Adam-et használtunk, a learning rate szintén próbálkozás alapján állt be 1e-3-ra, ennek ötszöröse, de már háromszorosa is túl soknak bizonyult, kevésbé konvergált. Az epoch-számra 40 alakult ki az általános megoldásnak, 20 epoch mellett sokkal kevésbé teljesítettek jól a modellek. Ugyanakkor egyszer megpróbáltuk 80 epoch mellett is, amely jobb eredményeket adott, mint 40 után, de nem annyival, hogy érdemes legyen kétszer annyi időt szánni a tanításra.

Kiértékelés

A modellek kiértékelését több különböző módszerrel végeztük:

- FID score
- KID score
- Diversity score
- Inception score

FID Score (Fréchet Inception Distance)
A valós és a generált képek eloszlása közötti hasonlóságot méri. Egy előre betanított Inception-hálózatot használ a két képhalmazból származó feature-embedding-ek kinyerésére, és kiszámítja a köztük lévő Fréchet-távolságot. Az alacsonyabb FID-pontszám azt jelzi, hogy az eloszlások jobban hasonlítanak egymáshoz, a tökéletes 0 pontszám pedig azonos eloszlásokat jelent.

KID Score (Kernel Inception Distance)
A Kernel Inception Distance (KID) a FID alternatívája. A FID-hez hasonlóan ez is egy előre betanított Inception-hálózatból kinyert jellemzőkre támaszkodik. A Fréchet-távolság helyett azonban a valós és a generált kép eloszlások közötti maximális átlagos eltérést (MMD) számítja ki egy polinomiális kernel segítségével. Eredmények értelmezése a FID-el azonos.

Diversity Score
Azt méri, hogy mennyire változatosak a generált képek. Összehasonlítja a generált képek közötti különbségeket, általában a képjellemzők közötti távolságok függvényében (mint például a koszinusz hasonlóság vagy az euklideszi távolság).
Amíg a FID és a KID a minőséget és a sokféleséget egyetlen pontszámban kombinálják, a Diversity Score kifejezetten a generált képek változatosságára összpontosít. Minél nagyobb értéket kapunk annál jobbak (változatosabbak) a generált képek.

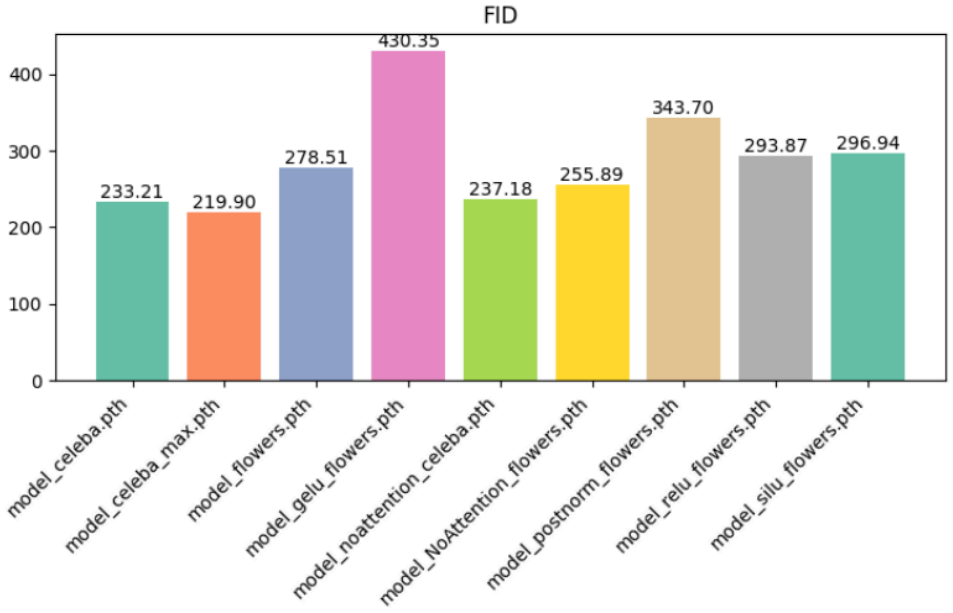
Inception Score
A generált képek valósághűségét és változatosságát méri. A generált képeket egy előre betanított Inception-hálózaton vezeti át, hogy megkapja az osztályok valószínűségi eloszlását, amiből méri, hogy a modell mennyire magabiztos a kép osztályozásában (realizmus) és mennyire változatosak a prediktált osztályok (változatosság). Minél nagyobb értéket kapunk annál jobbak a generált képek.

Kapott eredmények

Baseline model (VAE) eredménye:

```
FID Score: 330.67
Inception Score: 1.47 ± 0.06
KID Score: 0.3762 ± 0.0151
Diversity Score: 22.57
```

Különböző diffusion modellek eredményei:



Legjobb modellek:

- celeba-max.pth:219,90
- model_celeba.pth: 233.21

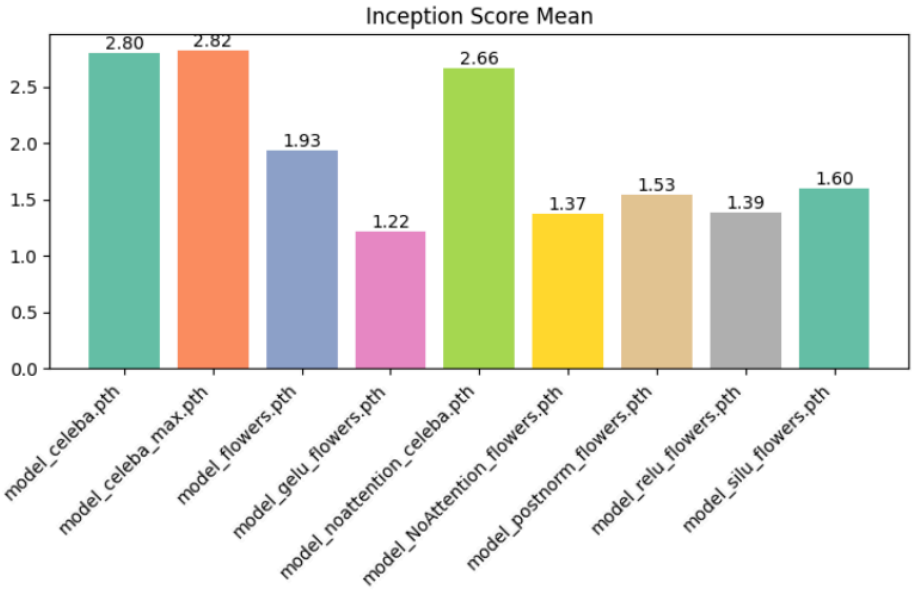
Ezek a modellek rendelkeznek a legalacsonyabb FID pontszámmal, ami azt jelenti, hogy a valós adat eloszláshoz a legjobban hasonlító eloszlású képeket generálják

Legrosszabb modell:

- model_gelu_flowers.pth: 430,35

Ez a modell rosszul teljesít, ami arra utalhat, hogy valami hiba történhetett a tanításnál, ugyanis ennek a modellnek nagyon közeli eredményt kellene adnia a model_flowers-hez, hiszen a ConvNext blokkokban eleve GELU-t használunk.

A *celebA-n* tanított modellek jobban teljesítenek a *flowers102-n* tanított modelleknél, valószínűleg az adathalmaz összetettségében vagy a modellképzésben mutatkozó különbségek miatt (pl.: több elérhető tanuló adat a *celebA*-nál).



Legjobb teljesítményű modellek:

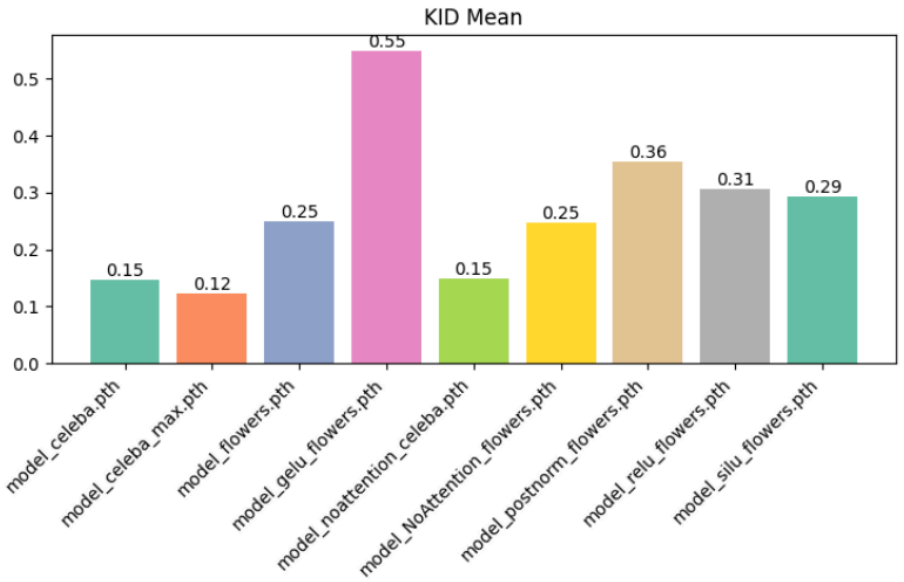
- model_celeba_max.pth: 2.82
- model_celeba.pth: 2,80

Ez a két modell éri el a legmagasabb Inception Scores értéket, ami azt jelzi, hogy az általuk generált képek változatosak és jó minőségűek. A celeba modellek ebben az értékelésben továbbra is felülmúlják a többi modellt.

Legrosszabbul teljesítő modell:

- model_gelu_flowers.pth: 1,22

Ez a modell mutatja a legalacsonyabb Inception Score-t, ami arra utal, hogy az általa generált képek nem sokszínűek és nem elég jó minőségűek. Ez ismét a vélhető hibának tudható be.



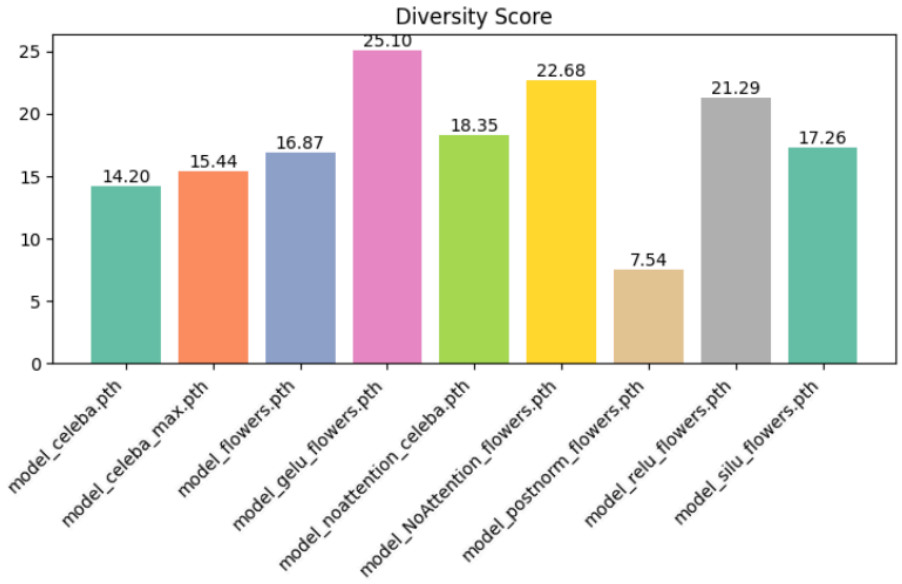
Legjobb teljesítményű modellek:

- modell_celeba_max.pth: 0.12
- model_celeba.pth: 0.15

Ezek a modellek érik el a legalacsonyabb KID pontszámokat, ami azt jelzi, hogy a generált képeik jobban illeszkednek a valós adatok eloszlásához.
Legrosszabbul teljesítő modell:

- model_gelu_flowers.pth: 0,55

Ennek a modellnek van a legmagasabb KID pontszáma, ami tovább erősíti a FID-ből és az Inception score-ból levont következtetést, miszerint a tanításba hiba csúszott.
A *celebA* itt is jobban teljesít a *flowers102*-nél.



Legjobb teljesítményű modellek:

- model_gelu_flowers.pth: 25.10
  Ez a modell éri el a legmagasabb Diversity Score értéket, ami a többi modellhez képest kivételes változékonyságot jelez a kimenetben.Mivel mindenhol máshol ez a legrosszabbul teljesítő modell, lehetséges, hogy a magas diverzitás nem pozitívum, hanem hibára utal.

- model_NoAttention_flowers.pth: 22,68
  Annak ellenére, hogy más metrikákban (FID, KID, IS) rosszul teljesít, ez a modell viszonylag magas diverzitást mutat, ami valószínűleg a túlzott általánosításnak köszönhető.

Legrosszabbul teljesítő modell:

- model_postnorm_flowers.pth: 7.54
  Ennek a modellnek van a legalacsonyabb pontszáma, ami arra utal, hogy nagyon hasonló képeket generál, kevés eltéréssel. A poszt-normalizálás használata korlátozhatja a sokféleséget.

A *flowers102* adathalmazon tanított modellek összességében magasabb diverzitási pontszámot mutatnak a *celebA* adathalmazon tanított modellekhez képest, ami valószínűleg a minták, színek és jellemzők nagyobb változatosságának köszönhető.

Összegzés

Az eredmények alapján nagyrészt túlteljesítik a modellek az eredeti baseline VAE-t. Ami jó jel, elértük amit szerettünk volna.
Összességében legjobb modell a model_celeba_max.pth lett.

- Legalacsonyabb FID : Jelzi, hogy olyan képeket generál, amelyek nagymértékben hasonlítanak a valós adat eloszláshoz.
- Legalacsonyabb KID.
- Legmagasabb Inception Score: A képminőség és a diverzitás közötti egyensúlyra utal.
- Mérsékelt diverzitás: Elegendő változatosságot biztosít a realizmus feláldozása nélkül.

A különbség a model_celeba.pth és a model_celeba_max.pth között egyrészt 10-zel több tanító epochban, másrészt a tanító adatban van (30000 helyett 80000). Ennyi fért maximálisan a memóriába. Ezért is nagyon hasonló a két model eredménye. Tehát ha minden modellt egyenlő esélyekkel tanítottunk volna, model_celeba.pth lett volna a legjobb opció.

Még egy celeba relu model is betanításra került, de mivel nagyon rossz értékeket mutat ezért nem szerepeltettük a fentebbi gráfokon.

Mint az eredményekből is látható a *celebA-n* tanított modellek jobban teljesítettek, ez valószínűleg a nagyobb mennyiségű tanító adatnak köszönhető.

LLM használat

A házi feladat során kódgenerálásra, kódok átstrukturálására és kommentezésére használtunk LLM-eket (Claude, ChatGPT).