Vpliv delitve latentnega prostora na učenje generativnih nasprotniških mrež na vektorskih podatkih

Luka Lukač, Niko Lukač, Damjan Strnad

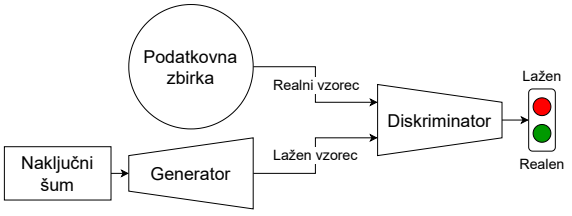
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
E-pošta: luka.lukac@um.si

Latent space partitioning influence on generative adversarial network training for vector data

In the paper, the effects of random and semi-regulated latent space sampling on the training of a conditional generative adversial network for the generation of written characters are compared. The semi-regulated sampling of latent vectors is achieved by partitioning the latent space according to randomly placed class centroids. The potential advantage of such approach is improved training convergence in distributed environments where multiple models could share the latent space partitioning. However, it is shown that the completely random sampling is actually the better choice, because the semi-regulated sampling cripples the efficiency of the generative adversial network training both in terms of realistic character generation as well as their diversity.

1 Uvod

Generativne nasprotniške mreže (angl. *generative adversarial network*, krajše GAN) so priljubljena metoda strojnega učenja za generiranje podatkov s podobnimi značilnostmi, kot jih imajo podatki v učni množici [1]. GAN sestoji iz dveh modelov nevronskih mrež: iz generatorja, ki iz latentnega vektorja tvori vzorec, in iz diskriminatorja, ki oceni kakovost generiranega vzorca. Pri učenju GAN-a se izmenjujeta diskriminator in generator, pri čemer je cilj generatorja generirati čim bolj realistične vzorce in s tem pretentati diskriminator, da le-ta ne zna razlikovati med realnimi in generiranimi vzorci. Delovanje GAN-a shematsko prikazuje slika 1.



Slika 1: Shema delovanja GAN-a.

Umetne vzorce generator generira iz naključnih vektorjev, vzorčenih iz latentnega prostora. Navadno se v ta namen vektorje znotraj latentnega prostora vzorči po Gaussovi porazdelitvi [2, 3]. Tovrstno vzorčenje je sicer enostavno za implementacijo, vendar lahko v določenih primerih povzroči nestabilnost in divergenco učenja [4]. Poznamo tudi vzorčenje latentnih vektorjev z delitvijo latentnega prostora na podprostore, iz katerih nato v fazi učenja vzorčimo latentne vektorje za posamezne vzorce [5]. V tem članku bomo primerjali efekte naključnega vzorčenja latentnih vektorjev in vzorčenja z delitvijo latentnega prostora. Ideja za predlagano metodo vzorčenja na podlagi delitve latentnega prostora je zagotoviti večjo stabilnost učenja in s tem izboljšati konvergenco ter učinkovitost modelov z isto latentno delitvijo v porazdeljenih okoljih. Nekatere sorodne metode [4, 6, 7, 8] se sicer ukvarjajo s problemom učinkovitega vzorčenja latentnega prostora, a nobena ne operira z vektorskimi podatki.

Nadaljevanje članka sestoji iz poglavja 2, v katerem opišemo delitev latentnega prostora in izbiranje latentnih vektorjev v razdeljenem prostoru, poglavja 3, kjer predstavimo rezultate in primerjamo delovanje GAN-a brez in z delitvijo latentnega prostora, in poglavja 4, v katerem strnemo rezultate.

2 Vzorčenje latentnih vektorjev z delitvijo latentnega prostora

Pred začetkom učenja GAN-a metoda za vsak razred c_i v učni množici z Gaussovim vzorčenjem ustvari karakteristični vektor $z_i^c \in Z$ v d-dimenzionalnem latentnem prostoru. Pri generiranju vzorcev z generatorjem imamo dve možnosti vzorčenja latentnih vektorjev z:

1. Za vsak vzorec razreda c_i, ki ga generiramo, kot latentni vektor z vzamemo njegov pripadajoči karakteristični vektor z_i^c in vanj vnesemo Gaussov šum.
2. Naključno vzorčimo latentni prostor po Gaussovi porazdelitvi, dokler evklidska razdalja med dobljenim latentnim vektorjem z in karakterističnim vektorjem z_i^c razreda vzorca c_i, ki ga želimo generirati, ni najmanjša med razdaljami do posameznih karakterističnih vektorjev v Z.

Prva možnost omogoča najstabilnejše učenje, saj za posamezen razred vedno izberemo vektor blizu z_i^c, vendar je posledično raznolikost med generiranimi vzorci is-

tega razreda majhna. Ta pojav pri GAN-ih pogosto imenujemo padec modusa (angl. *mode collapse*) [9]. Ideja za delitvijo prostora je, da podobne latentne predstavitve določajo podobne, a še vedno dovolj raznolike generirane vzorce.

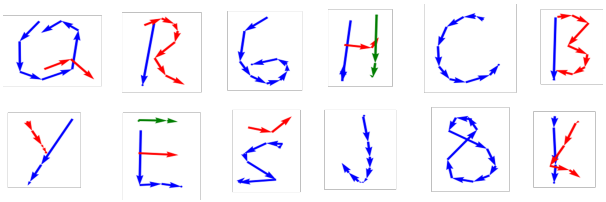
Izbiranje latentnih vektorjev z delitvijo latentnega prostora z našo implementacijo je prikazano v psevdokodu 1. Za vzorec, ki ga generiramo, najprej odčitamo pripadajoč karakterističnі latentni vektor z_i^c iz sekljalne razpredelnice (vrstica 6). Znotraj celotnega latentnega prostora vzorčimo začetni latentni vektor z_0 po Gaussovi porazdelitvi (vrstica 7) in izračunamo smerni vektor s med vzorčenim ter karakterističnim vektorjem (vrstica 8). Ta vektor določa smer proti podprostoru, znotraj katerega je razdalja $D_i \in D$ od kateregakoli vzorčenega vektorja z do pripadajočega karakterističnega vektorja z_i^c manjša kot razdalja $D_j \in D,\ j \neq i$ do poljubnega drugega karakterističnega vektorja $z_j \in Z,\ j \neq i$ znotraj drugega podprostora. Vzorčen latentni vektor z premaknemo v smeri proti karakterističnemu vektorju razreda za naključno velik premik (vrstica 11) in preverimo, če se dobljeni vektor z nahaja znotraj podprostora danega karakterističnega vektorja (razdalja do njegovega karakterističnega vektorja je manjša kot druge razdalje). V tem primeru ga vrnemo kot rezultat funkcije (vrstica 14). V nasprotnem primeru izbiramo naključno velike premike, dokler se dobljen vektor z ne nahaja znotraj podprostora izbranega karakterističnega vektorja.

Psevdokod 1 Vzorčenje latentnega vektorja v razdeljenem.

```
1:  function VZORČI-LATENTNI-VEKTOR(c_i, Z, d)
2:                ▷ c_i: razred, katerega vzorec generiramo
3:                ▷ Z: množica karakterističnih vektorjev
4:                        ▷ d: dolžina latentnega vektorja
5:                              ▷ Rezultat: latentni vektor z
6:      z_i^c ← PridobiKarakterističniVektor(c_i, Z)
7:      z_0 ← VzorčiLatentniProstor(d)
8:      s ← z_i^c − z_0                          ▷ Smerni vektor
9:      loop
10:         r ← NaključnoŠtevilo(0, 1)
11:         z ← z_0 + r · s
12:         D ← IzračunRazdalj(z, Z)
13:         if IndeksNajmanjšega(D) == i then
14:             return z
15:         end if
16:     end loop
17: end function
```

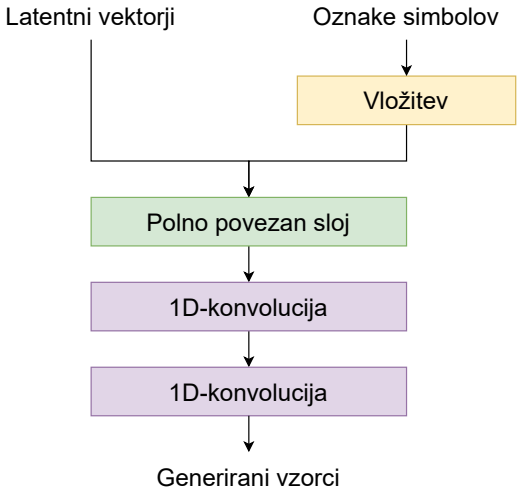
3 Rezultati

Pri testiranju metode smo uporabili lastno podatkovno zbirko črk in številk v vektorskem zapisu. Sestavljajo jo črke slovenske in angleške abecede ter številke, kar skupaj predstavlja 39 različnih simbolov. Pred generiranjem vzorcev surove podatke (v obliki zaporedja točk) predobdelamo, tako da izvedemo normalizacijo "min-max" (glede na daljšo stranico oklepajočega okvirja), poenostavimo zaporedja točk z uporabo algoritma Douglas-Peucker [10] in jih pretvorimo v vektorsko predstavitev fiksne dolžine. Vsak vektor med zaporednima dvema točkama je opisan s tremi vrednostmi: s premikom po osi x, premikom po osi y in z indikatorjem aktivnosti peresa (ali je poteza vidna ali predstavlja premik dvignjenega peresa). Vsak simbol je predstavljen z 10 vektorji, pri čemer je maksimalno število potez 5 (izbrano empirično za prepoznavne simbole z malo šuma). Primere tako predobdelanih podatkov prikazuje slika 2.

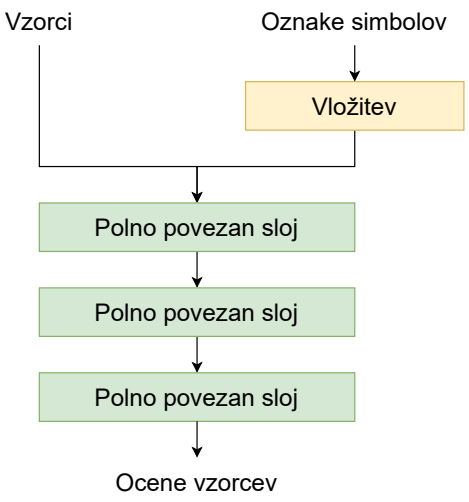


Slika 2: Primeri predobdelanih podatkov v vektorskem zapisu. Modra barva označuje poteze pred morebitnim dvigom peresa, rdeča barva poteze po prvem dvigu, zelena barva pa poteze po dveh dvigih.

Med učenjem GAN-a merimo izgubo z binarno prečno entropijo. Zaradi okrnjenega nabora podatkov uporabimo postopek razširitve učne množice (angl. *data augmentation*), pri kateri z dodajanjem šuma iz vsakega učnega vzorca dobimo 10 umetnih vzorcev. Za povečanje stabilnosti učenja in preprečevanja eksplodirajočih gradientov režemo gradiente po normi L2. Oba modela, generator in diskriminator, poleg osnovnega vhoda vsebujeta dodaten signal, ki pove, kateri simbol želimo generirati (v primeru generatorja) ali oceniti (v primeru diskriminatorja): zato govorimo o pogojnem GAN-u (angl. *conditional generative adversial network*, krajše cGAN). Generator je sestavljen iz treh konvolucijsko povratnih plasti ConvGRU in 1D-konvolucije, diskriminator pa je triplastni perceptron. Arhitekturi generatorja in diskriminatorja prikazuje slika 3.



(a)



(b)

Slika 3: Arhitekturi generatorja (a) in diskriminatorja (b).

Hiperparametre učenja smo določili eksperimentalno, zbrani pa so v tabeli 1. Prikazane hiperparametre smo uporabili tako za učenje GAN-a brez delitve latentnega prostora kot tudi GAN-a z delitvijo latentnega prostora.

Tabela 1: Izbrani hiperparametri učenja.

Hiperparameter	Vrednost
Število epoh	10.000
Velikost minipaketa	64
Velikost latentnega vektorja	100
Stopnja učenja	0,002

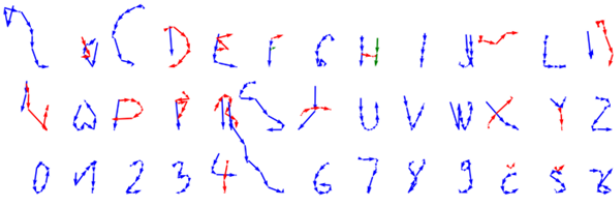
Rezultate učenja generatorja, ko latentni prostor ni bil razdeljen v podprostore, prikazuje slika 4, pri čemer smo generirali simbole dvakrat z različnimi latentni vektorji. Vidimo lahko, da so vsi simboli prepoznavni in da generator proizvaja vzorce s podobnimi značilnostmi, kot jih imajo izhodiščni učni vzorci. Opazimo lahko, da padca modusa pri posameznih simbolih ni, saj so ti pri večkratnem generiranju raznoliki in imajo drugačne značilnosti na sliki 4 (opazno predvsem pri simbolih Y in 4). Rezultate učenja generatorja z delitvijo latentnega prostora prikazuje slika 5. Čeprav se je generator naučil proizvajati določene razrede vzorcev, pa določenih vzorcev ne zmore generirati (denimo A, B, K in 5). Opazimo lahko tudi, da pri večkratnem generiranju določeni simboli nimajo velike raznolikosti, kar lahko nakazuje na (delni) pojav padca modusa [11].

Tabela 2 prikazuje povprečno kvadratno razliko (angl. *mean squared difference*, krajše MSD) in normalizirano korenjeno povprečno kvadratno razliko (angl. *normalised root mean squared difference*, krajše NRMSD) med pari vzorcev istih razredov glede na način vzorčenja latentnega prostora. Izkaže se, da sta pri delitvi latentnega prostora MSD in NRMSD večji, saj generator ne zmore

(a)

(b)

Slika 4: Rezultati generatorja brez delitve latentnega prostora: (a) Prva iteracija generiranja. (b) Druga iteracija generiranja.



(a)

(b)

Slika 5: Rezultati generatorja z delitvijo latentnega prostora: (a) Prva iteracija generiranja. (b) Druga iteracija generiranja.

kvalitetno generirati določenih vzorcev in namesto njih generira raznolike, a nerealne krivulje.

Tabela 2: Primerjava generiranih rezultatov brez in z delitvijo latentnega prostora.

Vzorčenje latentnih vektorjev	MSD	NRMSD
Brez delitve latentnega prostora	0,016761	0,067375
Z delitvijo latentnega prostora	0,038333	0,105576

Razlog za nezmožnost generiranja določenih vzorcev bi lahko bil ta, da latentni vektor nima vedno enake porazdelitve. Predlagana metoda vzorči latentni prostor po Gaussovi porazdelitvi, nato pa pomika začetni vzorec v smeri podprostora izbranega razreda. Posledično so komponente latentnih vektorjev poljubno porazdeljene, zato se generator ni zmožen naučiti preslikave latentnih vek-

torjev v realne vzorce.

4 Zaključek

Članek predstavlja študijo vpliva delitve latentnega prostora v podprostore na učenje GAN-ov z vektorskimi podatki. Predlagana metoda vzorčenja latentnih vektorjev v razdeljenem latentnem prostoru vsakič izbere drug latentni vektor, ki pa se nahaja znotraj latentnega podprostora posameznega razreda. Metodo smo testirali na lastni podatkovni zbirki simbolov slovenske in angleške abecede v vektorski obliki. Rezultati izkazujejo, da učenje GAN-a brez delitve latentnega prostora deluje bolje kot z delitvijo latentnega prostora, saj se pri delitvi latentnega prostora stabilnost učenja ne poveča. Poleg tega se pojavljajo posamezni simboli, kjer je opazen padec modusa, in simboli, kjer generator ne zajame prave gostote porazdelitve realnih podatkov. Glede na opisano delitev latentnega prostora v teoriji sicer ponuja stabilnejše učenje, a se v praksi pri vektorskih podatkih ne izkaže kot uporabna tehnika za učenje GAN-a.

Zahvala

Raziskovalno delo je sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije v okviru programa mladih raziskovalcev s številko 0796-59772 in raziskovalnega programa s številko P2-0041.

Literatura

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27 (NIPS 2014)* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), (Montreal, QC, Canada), Dec. 2014.

[2] J. Zhang, "Generative adversarial networks," Feb. 2018. Accessed: Jun. 5, 2025. [Online.] Available: https://tjmachinelearning.com/lectures/1718/gan/.

[3] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," *arXiv:1707.05776*, May 2019.

[4] Z. Hou, N. Lang, and X. Zhou, "WL-GAN: Learning to sample in generative latent space," *Information Sciences*, vol. 700, p. 121834, May 2025.

[5] Y. Li, Y. Mo, L. Shi, and J. Yan, "Improving generative adversarial networks via adversarial learning in latent space," in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), (New Orleans, LA, USA), Dec. 2022.

[6] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent space clustering in generative adversarial networks," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, vol. 33, (Honolulu, HI, USA), pp. 4610–4617, AAAI Press, July 2019.

[7] M. Armandpour, A. Sadeghian, C. Li, and M. Zhou, "Partition-guided GANs," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Nashville, TN, USA), pp. 5099–5109, IEEE, June 2021.

[8] A. Asperti and V. Tonelli, "Comparing the latent space of generative models," *Neural Computing and Applications*, vol. 35, pp. 3155–3172, Feb. 2023.

[9] Y. Kossale, M. Airaj, and A. Darouichi, "Mode collapse in generative adversarial networks: An overview," in *Proceedings of the 8th International Conference on Optimization and Applications (ICOA)* (H. Hachimi, F. Masulli, S. Rovetta, and M. Ribaudo, eds.), (Genoa, Italy), pp. 1–6, IEEE, Oct. 2022.

[10] A. Saalfeld, "Topologically consistent line simplification with the Douglas-Peucker algorithm," *Cartography and Geographic Information Science*, vol. 26, no. 1, pp. 7–18, 1999.

[11] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv:1809.11096*, Feb. 2019.