Text-To-Image Stable Diffusion Model

Lazić, Ivan

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Ivan Lazić

Text-to-image Stable Diffusion model

Završni rad

Osijek, 2023.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Ivan Lazić

Text-to-image Stable Diffusion model

Završni rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Osijek, 2023.

Sažetak: U ovom radu ćemo objasniti način funkcioniranja *text-to-image* modela i pokazati primjer kontrole Stable Diffusion modela pomoću ControlNet modela. Prije opisa rada spomenutih modela, proći ćemo kroz njihovu povijest. Nakon toga ćemo opisati način rada i matematičku pozadinu modela Stable Diffusion. Na kraju ćemo u grafičkom sučelju ComfyUI pokazati nekoliko primjera generiranja i poboljšanja kvalitete slike i kontrole Stable Diffusion modela predprocesorima kako bi preciznije upravljali njegovim rezultatom.

Ključne riječi: Text-to-image model, predprocesor, Stable Diffusion, ControlNet, ComfyUI

Text-to-image models and style transfer

Abstract: In this paper, we will explain how the text-to-image models work and show an example of controlling such a model using ControlNet models. Before describing how these models function, we will go through the history and mathematical background of Stable Diffusion. Finally, in the ComfyUI graphical interface, we will show several examples of generating and improving image quality, as well as controlling the Stable Diffusion model with preprocessors in order to more precisely manage its result.

Keywords: Text-to-image model, preprocessor, Stable Diffusion, ControlNet, ComfyUI

Sadržaj

1. Uvod

Generiranje slike iz prirodnog ljudskog jezika složen je proces. Proces koji se dijeli na razumijevanje teksta i samo generiranje slika. Razumijevanje teksta kod većine modela radi pomoću *Transformer* modela, dok generiranje slika koristi neke od algoritama od kojih su najkorišteniji GAN i Diffusion algoritam.

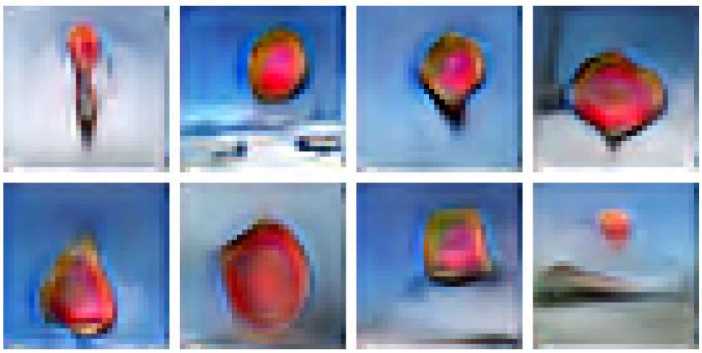
U ovom radu ideja je pokazati kako je došlo do takvih modela, kako rade te kako ih koristiti i kontrolirati njihov rezultat bez pisanja koda.

Temelji modela za generiarnje slika potječu još od kasnih 90-ih i ranih 2000-ih pojavom neuralnih mreža. Pomoću njih su nastali prvi modeli koji su služili za prepoznavanje rukom pisanih znakova od kojih je najpoznatiji LeNet-5, nastao 1998. godine.

Napretkom neuralnih mreža nastala je potreba za velikim skupovima podataka koji bi služili za treniranje. 2009. godine izašao je ImageNet, tadašnji najveći skup od 14 milijuna ručno opisanih slika. Konvolcijska neuralna mreža trenirana na tom skupu, koja je ujedno činila prvi klasifikator slika, izašla je 2012. godine pod imenom AlexNet.

Prvi generativni model GAN (Generative Adversarial Network) nastao je 2014. godine. GAN se sastoji od dvije neuralne mreže koje se natječu u obliku igre nultog zbroja što podsjeća na mimikriju u evolucijskoj biologiji.

Nadogradnja na generativne modele nastaje u obliku *text-to-image* modela. Prvi takav nastao je 2015. godine pod imenom AlignDRAW. Iako su generirane slike bile malih dimenzija i nisu bile realistične, model je mogao generirati dosad neviđene zahtjeve. AlignDRAW je prvi model koji koristi varijacijski autoenkoder (VAE) pomoću kojega reprezentacija slike nije morala biti u prostoru piksela nego u latentnom prostoru.



Slika 1: Slike koje AutoDRAW generira iz upita "A stop sign is flying in blue skies." [3]

Prvi modeli koji su mogli generirati realistične slike iz nespecijaliziranih skupova podataka su DALL-E (2021.) i Stable Diffusion (2022.). Oba modela koriste CLIP (Contrastive Language-Image Pre-training) neuralnu mrežu koja je trenirana na relacijama između teksta i slike, ali se razlikuju u načinu generiranja slike. DALL-E koristi GAN, dok Stable Diffusion koristi Diffusion model.

Slika 2: Slike generiane modelom DALL-E

Slika 3: Slike generirane modelom Stable Diffusion

2. Stable Diffusion

Stable Diffusion je open-source i može se pokretati lokalno na osobnom računalu i zbog toga je odabran kao fokus ovoga rada. U nastavku ćemo objasniti njegov način rada i matematičku pozadinu.

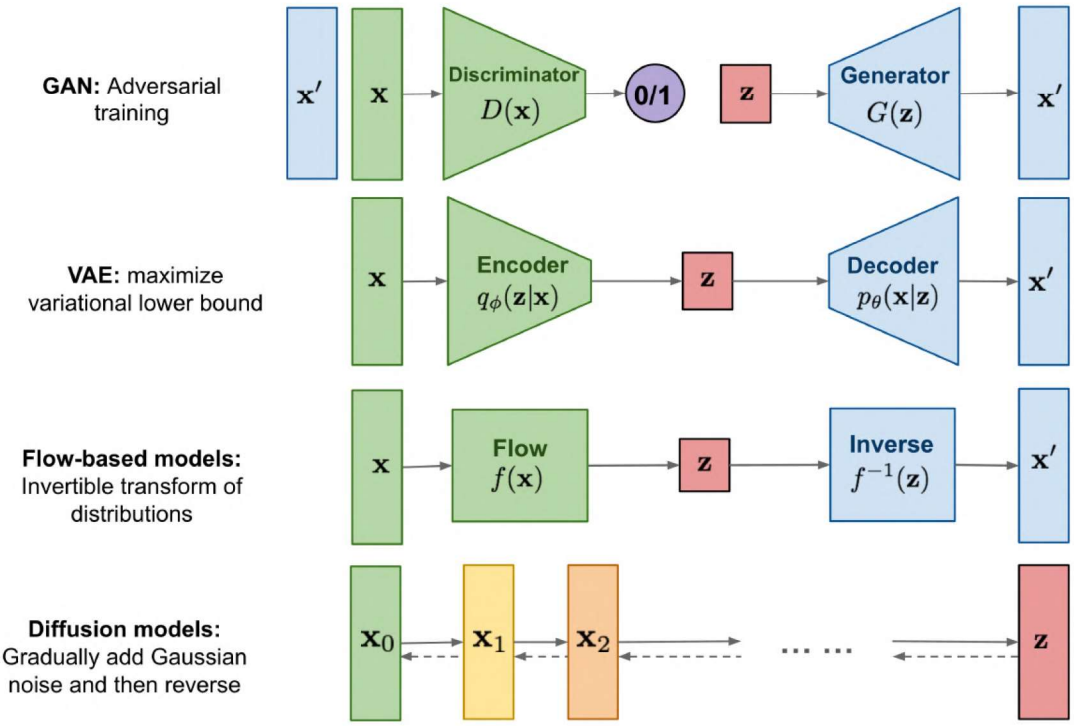
2.1. Način rada text-to-image modela

Text-to-image modele možemo podijeliti na tri dijela:

1. Kodiranje teksta
2. Generiranje slike
3. Evaluacija

Za kodiranje teksta često se koriste LSTM neuralne mreže, ali Transformer modeli postaju sve popularniji radi bržeg procesa treniranja.

Za generiranje slika koriste se već spomenuti GAN i u zadnje vrijeme popularniji Diffusion modeli.

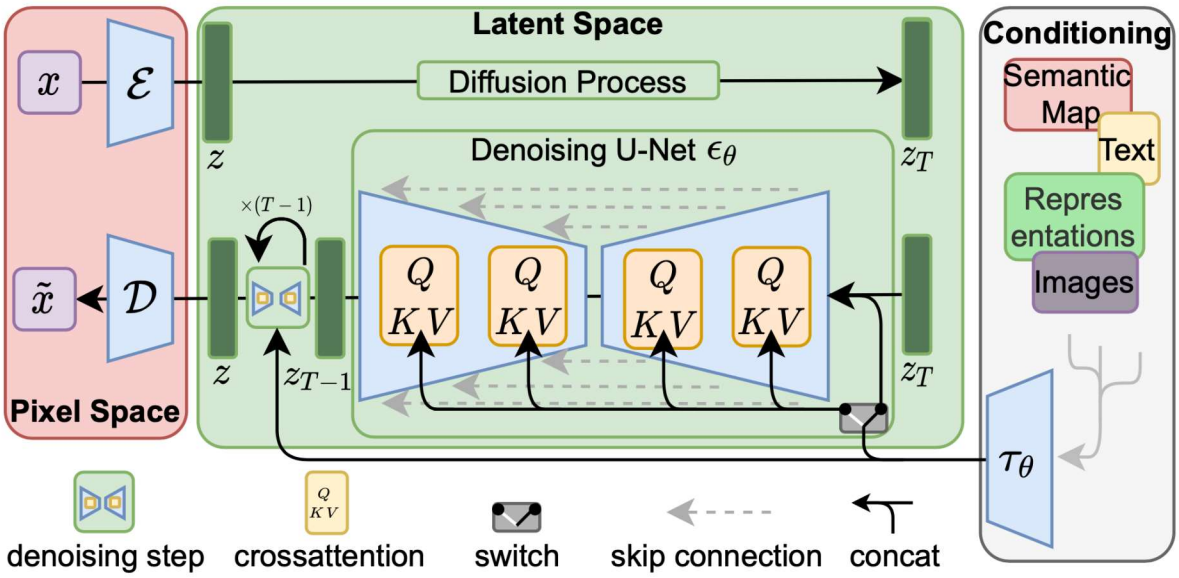


Slika 4: Različite metode generiranja slika [6]

Za evaluaciju se najčešće koristi Inception score (IS) kojeg računa Inceptionv3 klasifikacijski model. Što obilježje u slici ima veću vjerojatnost, koju predviđa evaluacijski model, to je slika točnije generirana i skor se povećava.

2.2. Način rada Stable Diffusion modela

Stable Diffusion je sličan ostalim modelima baziranim na principu difuzije, ali razlikuje se u tome što koristi latentni prostor koji je 48 puta manji što čini model znatno bržim. Takav model zovemo latentni difuzijski model. Dodatno Stable Diffusion je treniran na daleko većem skupu podataka, koristi CLIP model koji manipulira prediktorom šuma i U-Net kao prediktor.



Slika 5: Arhitektura modela Stable Diffusion [4]

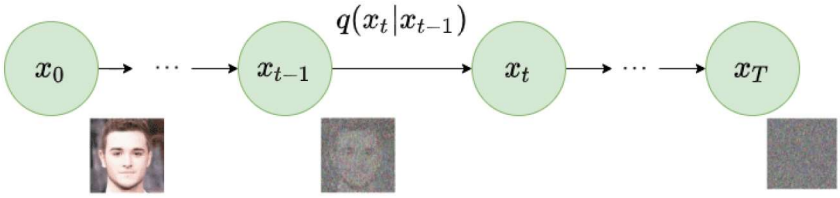
2.3. Difuzijski model

Difuzijski modeli [1] baziraju se na dva procesa, difuzija prema naprijed (dodavanje šuma) i difuzija prema natrag (uklanjanje šuma). Formuliraju se kao Markovljev lanac od T koraka, što znači da svaki korak ovisi samo o prethodnom.

Proces difuzije prema naprijed odvija se na način da podatku x_0 iz distribucije postojećih podataka $q(x)$ dodamo malu količinu Gaussovog šuma u T koraka. U svakom koraku Markovljevog lanca dodajemo Gaussov šum s varijancom β_t podatku x_{t-1}, čime dobijemo novu latentnu varijablu x_t s distribucijom $q(x_t|x_{t-1})$. Taj proces možemo formulirati na sljedeći način:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t : \mu_t = \sqrt{1-\beta_t}x_{t-1}, \Sigma_t = \beta_t I),$$

gdje je I jedinična matrica što znači da je standardna devijacija β_t neovisno o dimenziji.



Slika 6: Difuzija prema naprijed [1]

Budući da $q(x_t|x_{t-1})$ dolazi iz normalne distribucije definirane s μ_t i Σ_t, možemo doći od ulaza x_0 do x_t matematički definirano na sljedeći način:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}),$$

gdje $q(x_{1:T}|x_0)$ označava da primijenimo q T puta. To stvara problem kada je T velik, kojeg možemo riješiti trikom reparametrizacije koji kaže da možemo očekivanje zapisati u obliku takvom da distribucija bude neovisna o parametru. To nam omogućava da ako definiramo $\alpha_t = 1 - \beta_t, \overline{\alpha_t} = \prod_{s=0}^{T} \alpha_s, \epsilon_0, \ldots, \epsilon_{t-2}, \epsilon_{t-1} \sim \mathcal{N}(0, I)$ možemo rekurzivno pokazati:

$$\begin{aligned} x_t &= \sqrt{1-\beta_t} \\ &= \sqrt{\alpha_t}x_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\ &= \ldots \\ &= \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon_0 \end{aligned}$$

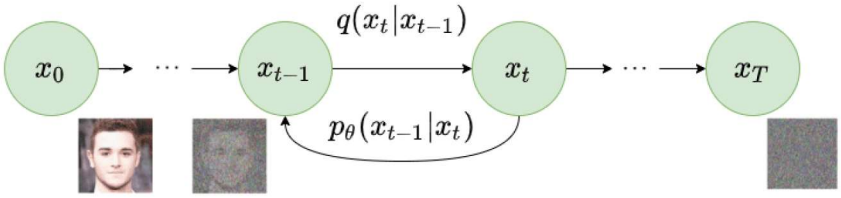
i definirati x_t na sljedeći način:

$$x_t \sim q(x_t|x_0) = \mathcal{N}(x_t : \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})I)$$

Budući da je β_t hiperparametar, možemo unprijed izračunati α_t i $\overline{\alpha_t}$ za svaki korak, što nam omogućuje uzorkovanje latentnte varijable x_t u svakom koraku.
U procesu difuzije prema natrag cilj nam je rekreirati uzorak $x_0 \sim q(x)$ iz Gaussovog šuma $x_T \sim \mathcal{N}(0, I)$. Ovaj proces aproksimiramo neuralnom mrežom.

$q(x_{t-1}|x_t)$ nam nije poznat i njega aproksimiramo parametriziranim modelom p_θ. Budući da je $q(x_{t-1}|x_t)$ iz normalne distribucije, za dovoljno mali β_t možemo odabrati p_θ također iz normalne distribucije i parametrizirati srednju vrijednost i varijancu na način:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$



Slika 7: Difuzija prema natrag [1]

Ako primijenimo obrnutu formulu na svakom koraku možemo iz x_t doći do distribucije:

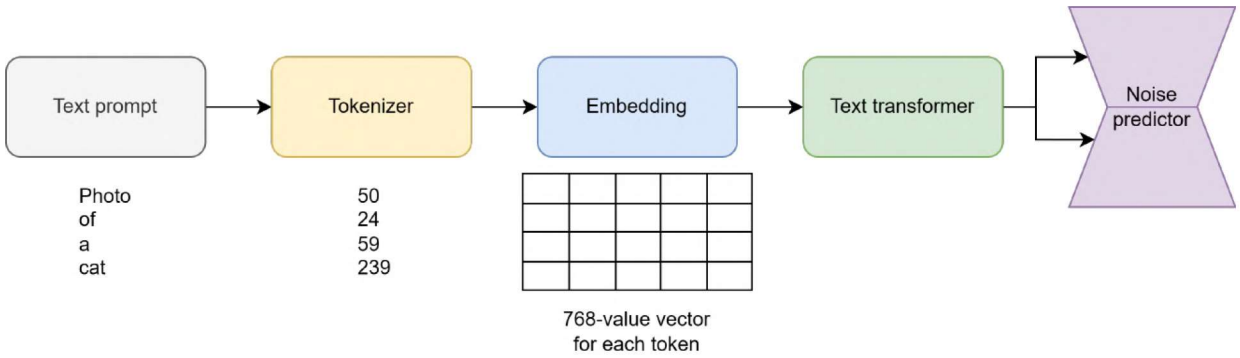
$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

Dodatnim uvjetovanjem modela na korak t, on će naučiti predviđati Gaussove parametre za svaki korak.

2.4. CLIP

Stable Diffusion korsti CLIP tokenizator koji omogućava vezu između teksta i slike. Tokenizator prvo pretvori riječi u računalu razumljiv oblik (uglavnom brojeve) te ih podijeli na tokene. Jedna riječ se ne pretvara nužno u jedan token jer tokenizator može pretvoriti samo one riječi na kojima je treniran. Tako da će na primjer riječ "plavipas" podijeliti u dva tokena "plavi" i "pas".

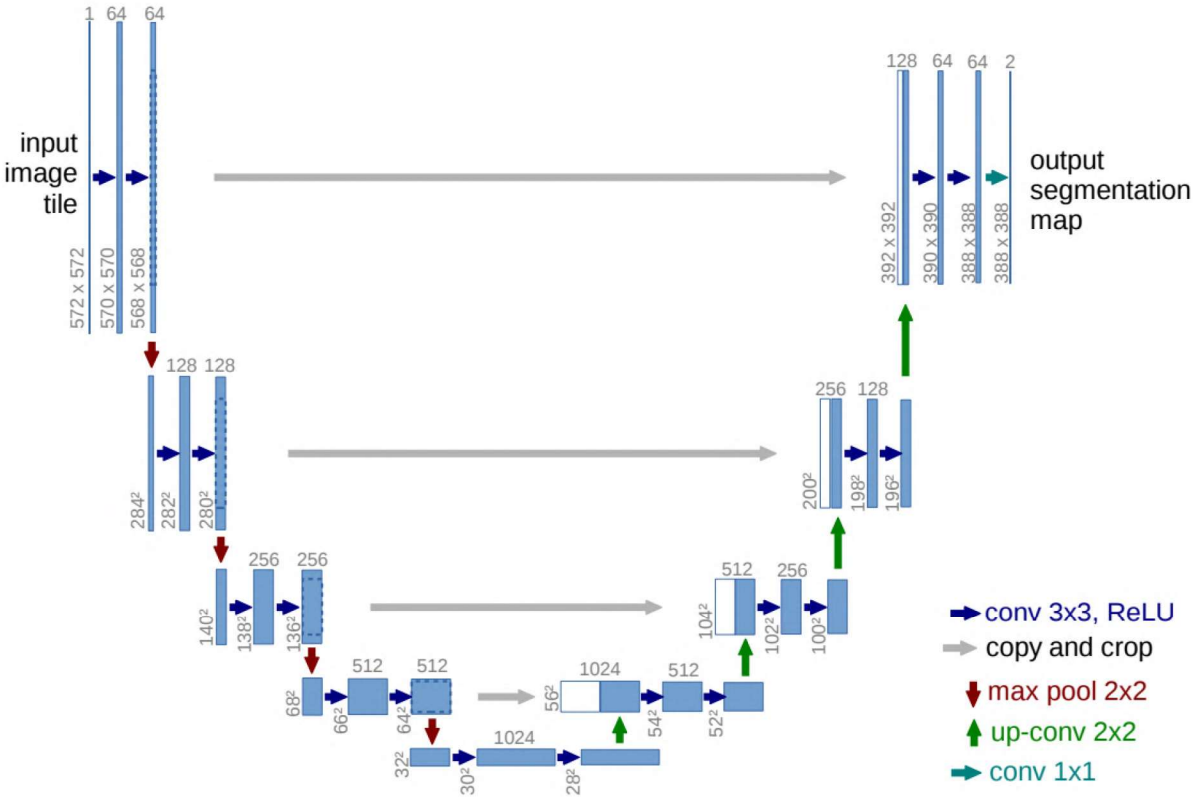
Nakon što se cijeli prompt podijeli na tokene, oni se šalju u transformator teksta koji dalje manipulira prediktorom šuma, u našem slučaju U-Net.



Slika 8: Prikaz obrade teksta prije U-Net-a [2]

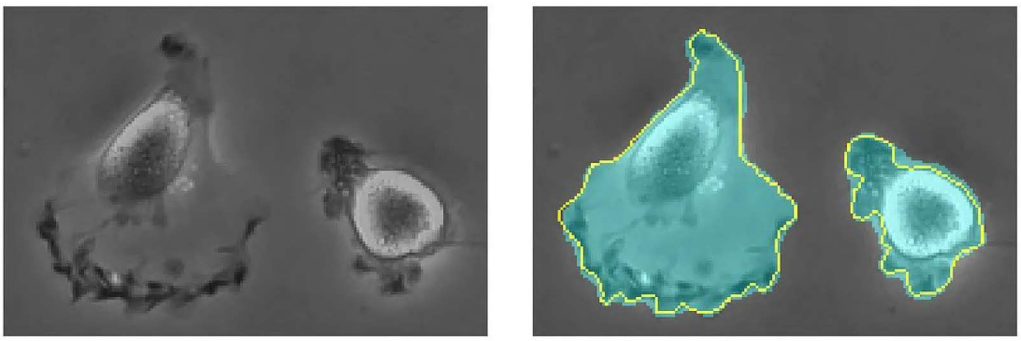
2.5. U-Net arhitektura

Svrha U-Net arhitekture je šum pretvoriti u informacije iz kojih se mogu izvući slikovni podaci koje kasnije dekoder koristi kako bi konstruirao gotovu sliku. Prvobitna svrha U-Net arhitekture bila je biomedicniska segmentacija slike, ali se u zadnje vrijeme počela koristiti i u generativnim modelima.



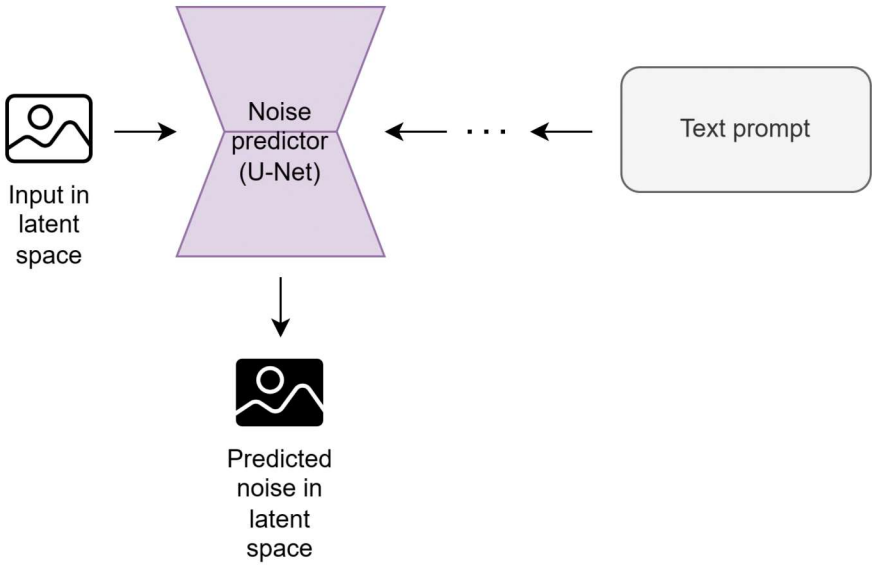
Slika 9: U-Net arhitektura [5]

Arhitektura je simetrična i ima oblik slova "U". Prvi dio postepeno smanjuje ulaz (originalnu sliku) i izvlači značajke (eng. features), a drugi dio proširuje međurezultate sve dok izlaz nije iste dimenzije kao i ulaz. Tim procesom smo dobili semantičku segmentaciju koja svakom pikselu slike odredi klasu kojoj pripada.



Slika 10: Semantička segmentacija [5]

Konkretno u Stable Diffusion modelu U-Net koristimo za predviđanje šuma i na temelju njega konstruiramo jednu iteraciju slike.



Slika 11: Predikcija šuma uvjetovana tekstom [2]

Nova iteracija slike se računa na način da od prijašnje iteracije oduzmemo šum koji je predvidio U-Net.

Slika 12: Konstrukcija nove iteracije slike [2]

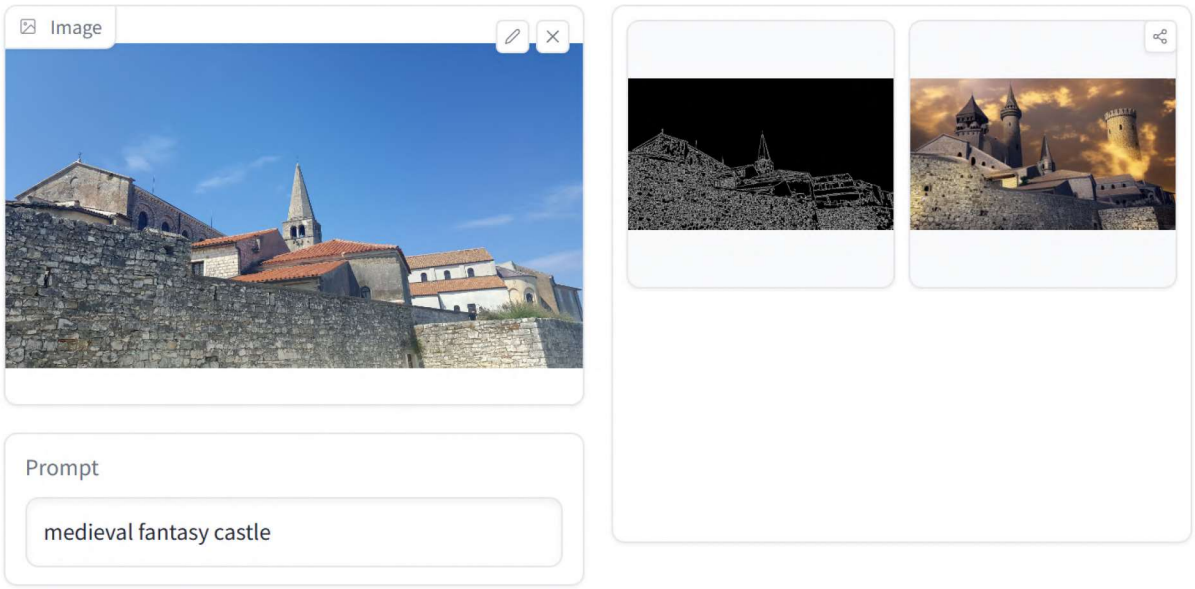
3. Generiranje slike iz teksta

U ovom dijelu rada ćemo ukratko objasniti alate koje ćemo koristiti i napraviti nekoliko primjera generiranja i manipularanja slikama kako bismo dobili što bolji rezultat.

3.1. ControlNet

ControlNet je skup modela koji omogućuju precizniju kontrolu Stable Diffusion modela.

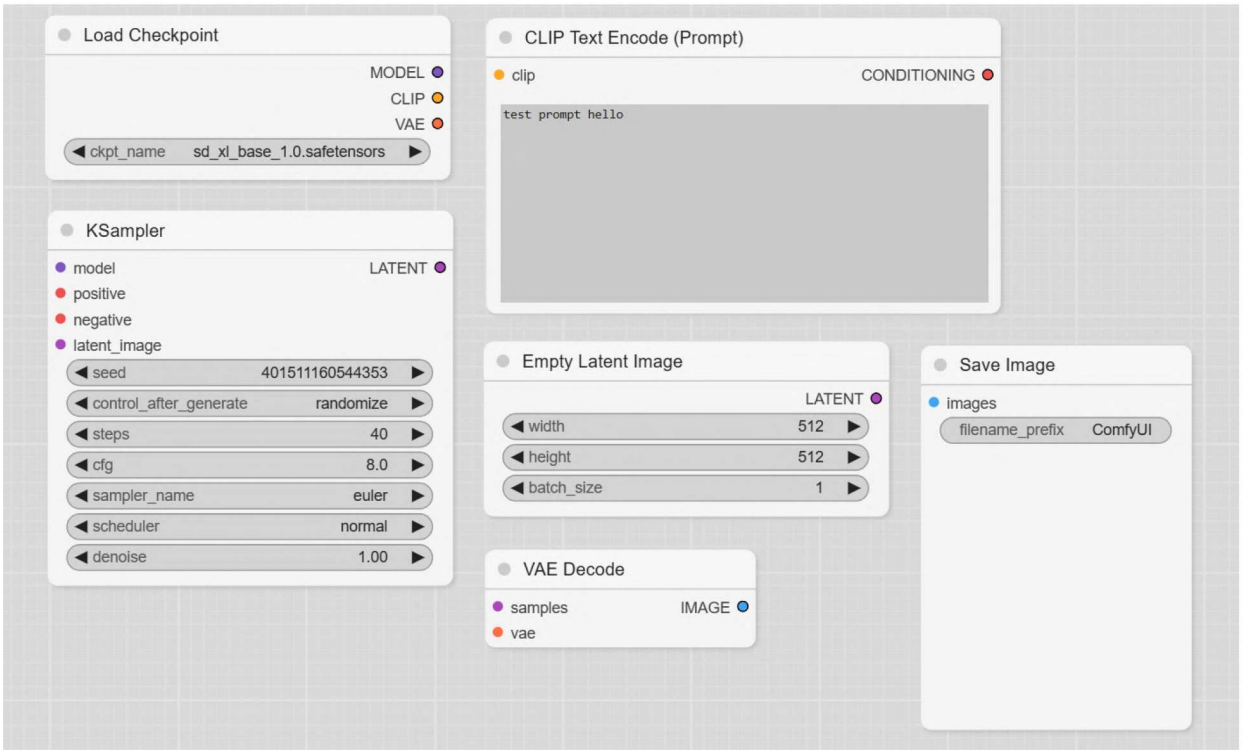
Do sada smo koristili samo tekst za kondicioniranje Stable Diffusion modela. ControlNet modeli dodaju nekoliko novih načina za kondicioniranje kao što su detekcija rubova, detekcija ljudskih poza, detekcija dubine, kontrola pomoću skice, itd.



Slika 13: Primjer kontrole modela s detekcijom rubova

3.2. ComfyUI

Primjeri u nastavku će biti dizajnirani u programu ComfyUI grafičkom sučelju u obliku grafa/dijagrama toka sa čvorovima, bez pisanja koda. Ovaj program omogućuje kreiranje kompleksnih procesa koji uključuju razne modele i alate za generiranje i obradu slike.



Slika 14: ComfyUI sučelje

3.3. Generiranje slike

Kako bismo na najjednostavniji način generirali sliku potrebno nam je nekoliko stvari. Prvo moramo učitati Stable Diffusion model. U ovom radu ćemo koristiti Stable Diffusion XL 1.0 model.

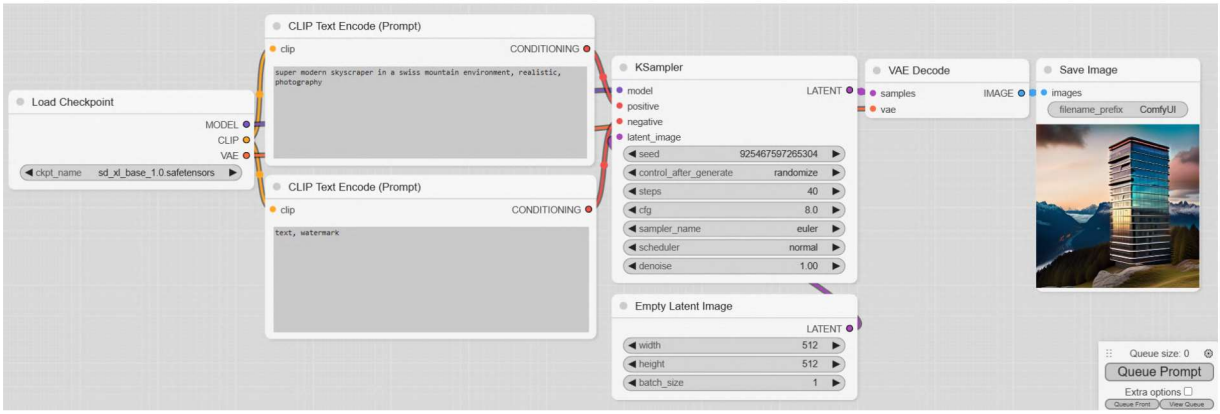
Za učitavanje koristimo čvor *Load Checkpoint* koji za izlaze ima *Model* koji predstavlja model za predikciju šuma, *CLIP* jezični model i *VAE* koji pretvara sliku iz prostora piksela u latentni prostor.

Idući čvor koji koristimo je *CLIP Text Encode* koji prompt transformira u uvjete za prediktor šuma pomoću CLIP modela. Koristimo dva takva čvora. Jedan predstavlja "pozitivan prompt", a drugi "negativan prompt". Model će pokušati što bliže izgenerirati ono što se nalazi u pozitivnom prompt-u i izbjegavati ono što se nalazi u negativnom prompt-u.

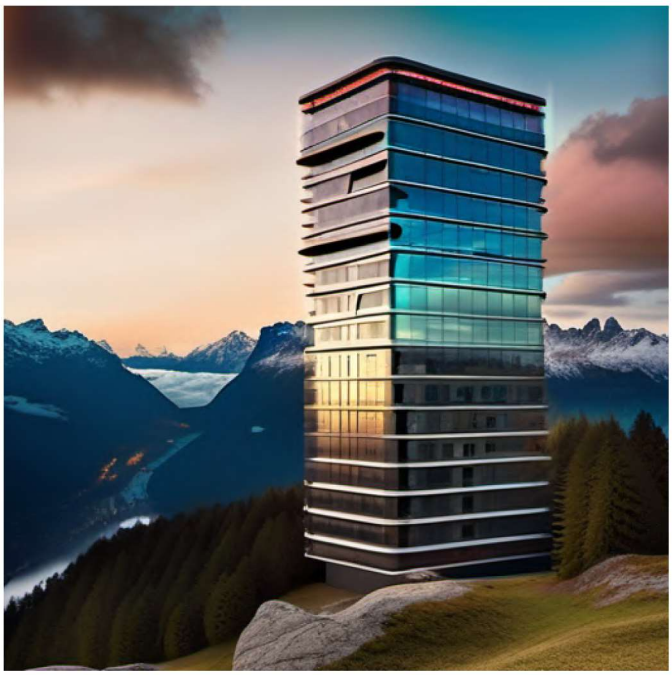
Empty latent image čvor predstavlja početnu sliku u latentnom prostoru koju prosljeđujemo u *KSampler*.

Glavni dio samog generiranja slike je *KSampler* čvor, on uklanja šum iz početne latentne slike uvjetovan tekstom. U ovom čvoru imamo nekoliko parametara, a najbitniji su:

1. *steps* - broj koraka uklanjanja šuma
2. *sampler name* - algoritam koji koristimo
3. *scheduler* - kontrolira na koji način se u svakom koraku uklanja šum
4. *denoise* - broj koji predstavlja koliko šuma se uklanja u svakom koraku



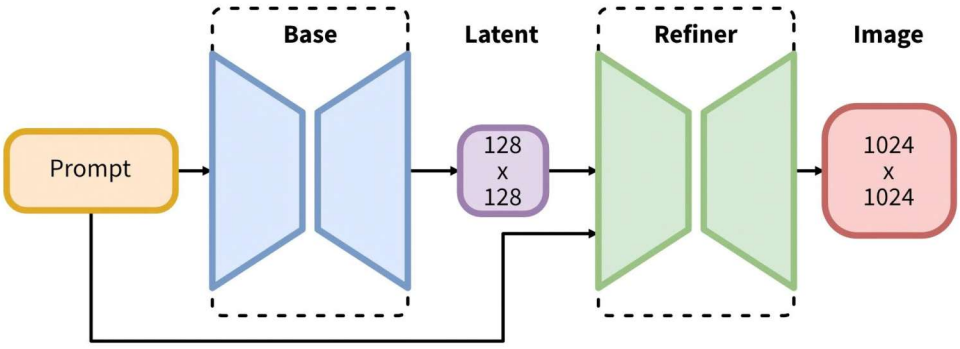
Slika 15: Workflow za jednostavno generiranje slike



Slika 16: Rezultat generiranja slike

3.4. Usporedba baznog i refiner modela

Stable Diffusion XL dolazi u dva oblika, *base* i *refiner*. Često bazni model generira slike lošije kvalitete zbog čega je nastao *Refiner* model. *Refiner* je odgovoran za uklanjanje zadnjih 15 do 20 posto šuma što poboljšava kvalitetu slike jer je specifično treniran za to.



Slika 17: Stable Diffusion XL Refiner arhitektura

Slika 18: Workflow baznog i refiner modela

(a) bazni model

(b) refiner model

Slika 19: Usporedba baznog i refiner modela

3.5. Embeddings

Embeddings omogućuju dodatnu kontrolu modela pomoću ključnih riječi. Rezultat je sličan podešavanju (eng. *fine-tuning*) modela, ali prednost je to što nove ključne riječi ne mijenjaju model nego se tokeniziraju (pretvore u broj) i u *embedding* vektor. Tehnika tekstualne inverzije (eng. *textual inversion*) nakon toga pronađe *embedding* vektor koji najbolje opisuje novu ključnu riječ čime je omogućena dodatna kontrola.

Sada ćemo usporediti rezultat sa i bez *embedding*-a. Za ovaj primjer ćemo koristiti Stable Diffusion XL 1.0 i *embedding* pod imenom *Bad Hands*. To je negativni *embedding* što znači da pokušava model odmaknuti od generiranja "loših ruku".

Slika 20: Usporedba specijaliziranog modela i embedding-a s baznim modelom

(a) bazni model bez embedding-a (b) specijalizirani model s embedding-om

Slika 21: Usporedba rezultata embedding-a i modela

Možemo primijetiti da su ruke na drugoj slici daleko bolje izgenerirane.

Još ćemo pokazati koliko se može poboljšati kvaliteta slike kombinacijom specijaliziranog modela i *embedding*-a. Koristimo *embedding* pod imenom *Fast Negative Embedding* koji je napravljen za model *Realistic Vision*. Ova kombinacija bi trebala dati daleko realističnije slike.

Slika 22: Usporedba specijaliziranog modela i embedding-a s baznim modelom

(a) bazni model bez embedding-a

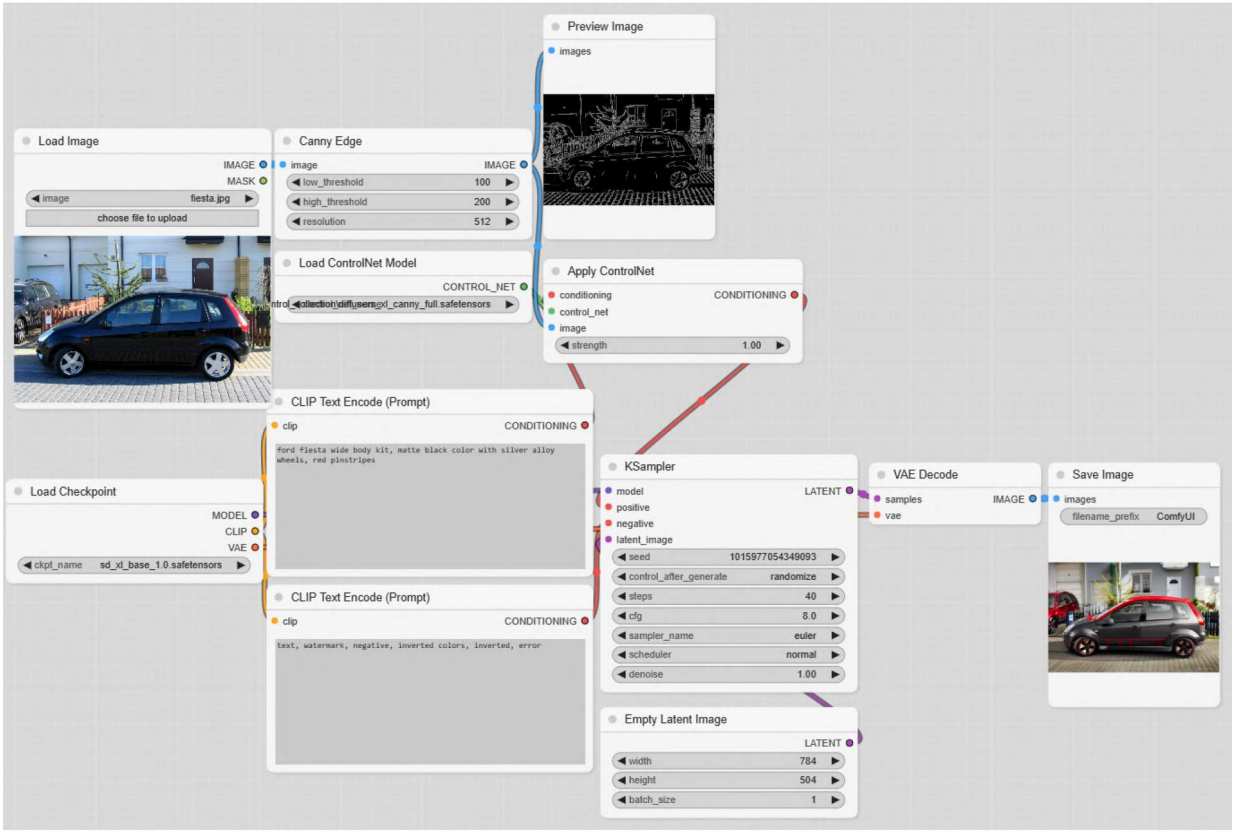
(b) specijalizirani model s embedding-om

Slika 23: Usporedba rezultata embedding-a i modela

3.6. ControlNet Canny

ControlNet Canny je model koji na temelju detektiranih rubova u početnoj slici kondicionira Stable Diffusion.

Kako bismo mogli koristiti ControlNet modele moramo instalirati ControlNet dodatak. U tom dodatku dobijemo nove čvorove od kojih su najbitniji čvorovi za predprocesuiranje slike kao što su detekcija rubova, dubine, poza, itd., čvor *Load ControlNet Model* koji učitava ControlNet model kojeg želimo koristiti i *Apply ControlNet* koji iz dobivene predprocesuirane slike i prompt-a stvara uvjete za Stable Diffusion.



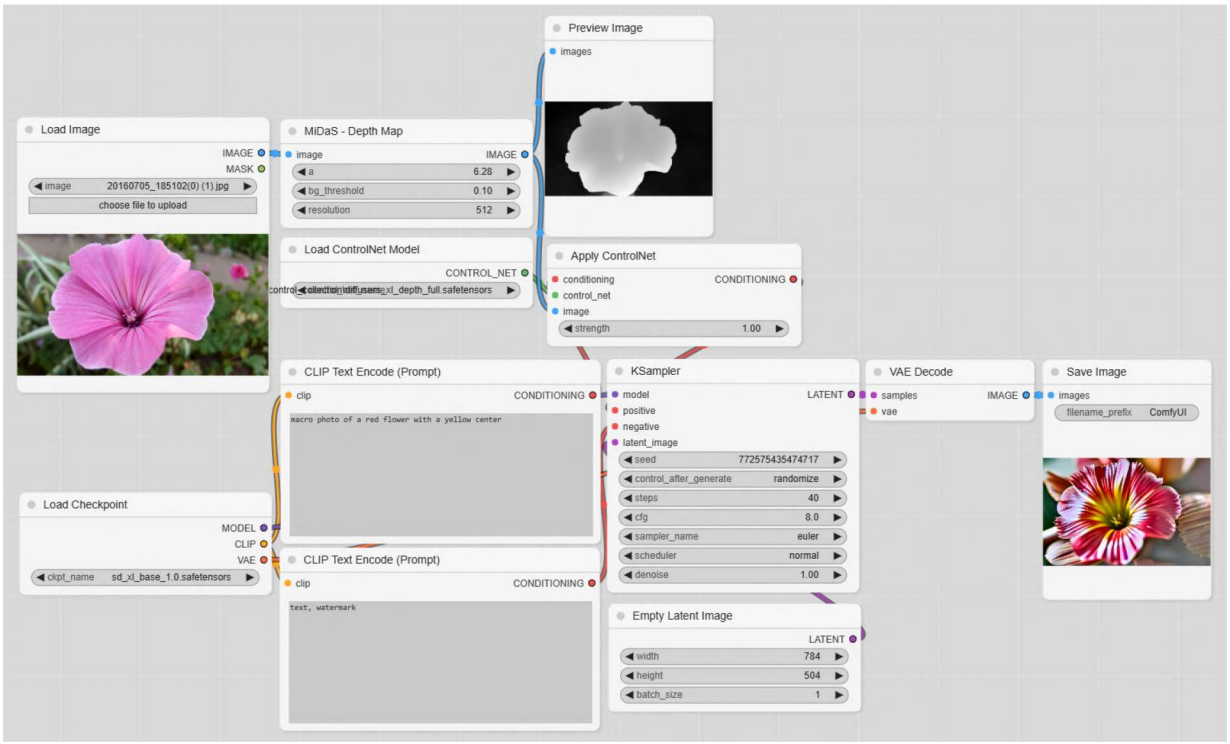
Slika 24: ControlNet Canny workflow

(a) početna slika (b) rezultat

Slika 25: Stable Diffusion kondicioniran s ControlNet Canny

3.7. ControlNet Depth

ControlNet Depth radi na sličan način kao i ControlNet Canny, ali za razliku od detekcije rubova, detektira dubinu objekata u slici.



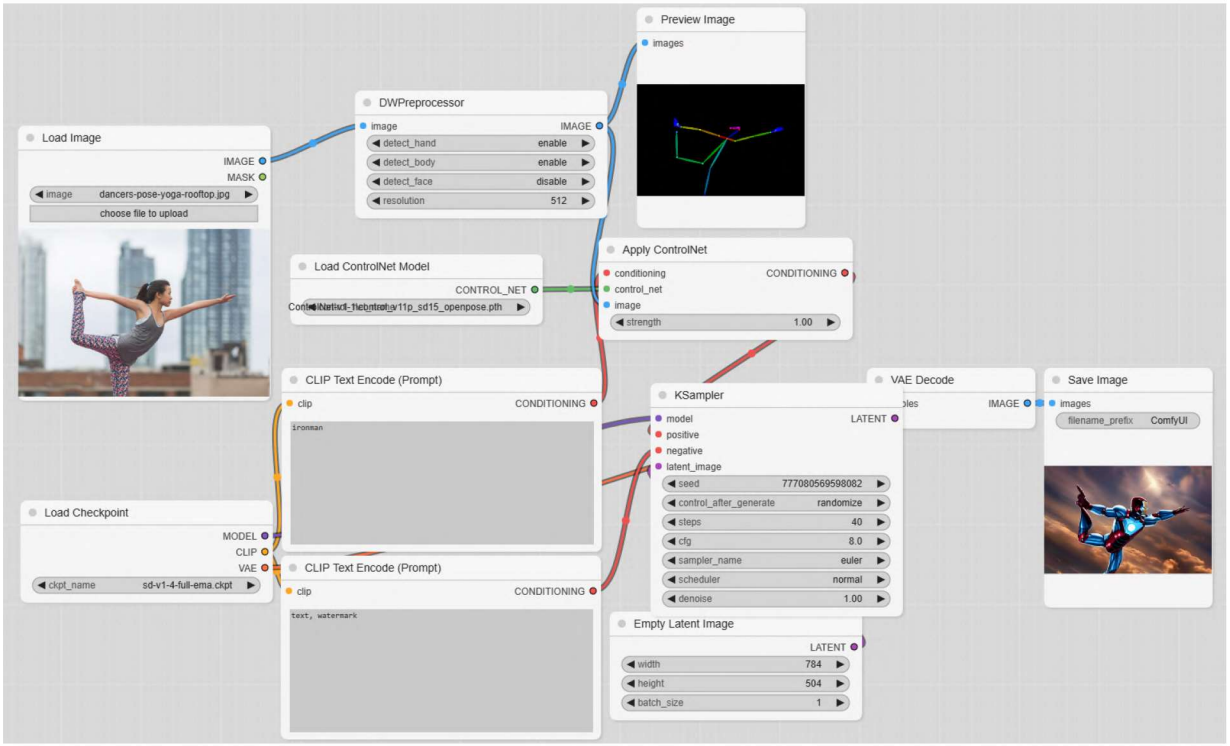
Slika 26: ControlNet Depth workflow

(a) početna slika (b) rezultat

Slika 27: Stable Diffusion kondicioniran s ControlNet Depth

3.8. ControlNet OpenPose

Koncept je isti kao i kod prijašnjih modela, ali ovdje se koristi detekcija ljudskih poza.



Slika 28: ControlNet OpenPose workflow

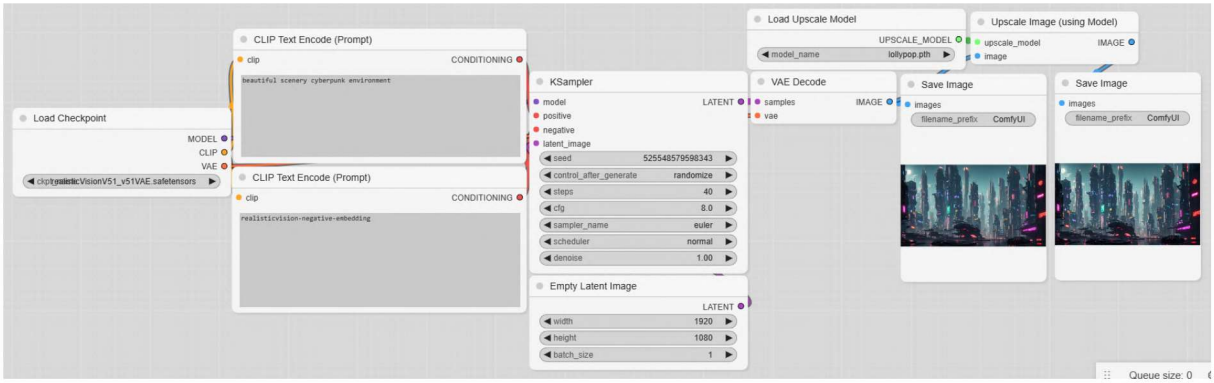
(a) početna slika

(b) rezultat

Slika 29: Stable Diffusion kondicioniran s ControlNet OpenPose

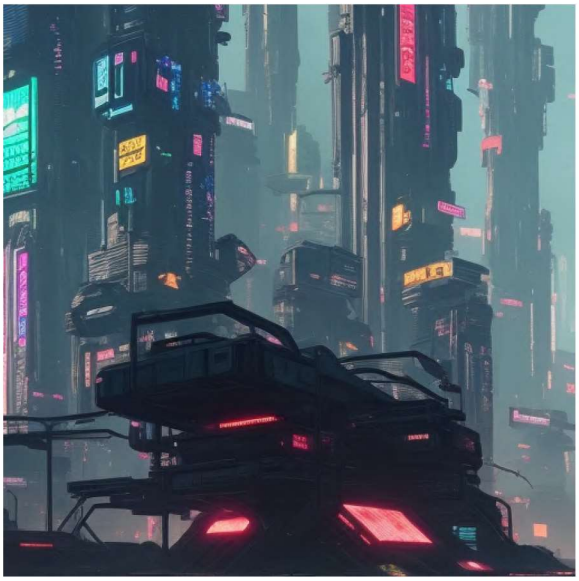
3.9. Modeli za poboljšanje oštrine

Modeli za poboljšanje oštrine mogu daleko poboljšati izgled slike, kao i ubrzati proces generiranja zato što model ne mora generirati sliku u velikoj rezoluciji. Ujedno, što su slike bliže rezoluciji na kojoj je model treniran, to su bolje. ComfyUI dolazi sa nekoliko čvorova koji omogućuju učitavanje modela za poboljšavanje oštrine slike. Za jednostavan primjer ćemo koristiti *Lollipop* model koji ćemo primijeniti samo jednom na generiranu sliku.

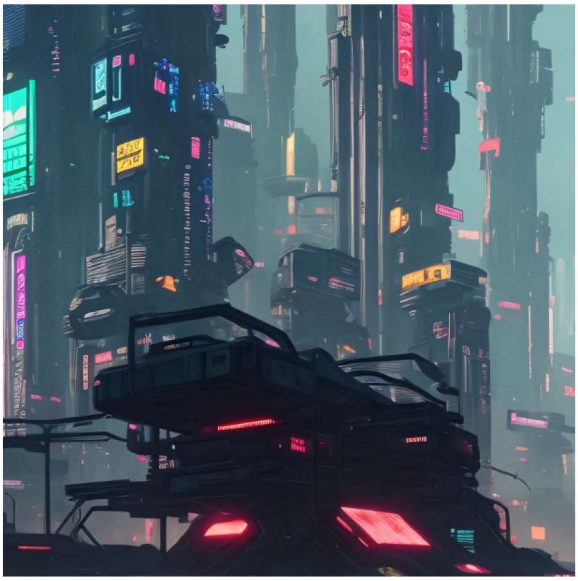


Slika 30: Upscale workflow

Slika 31: 8K rezultat



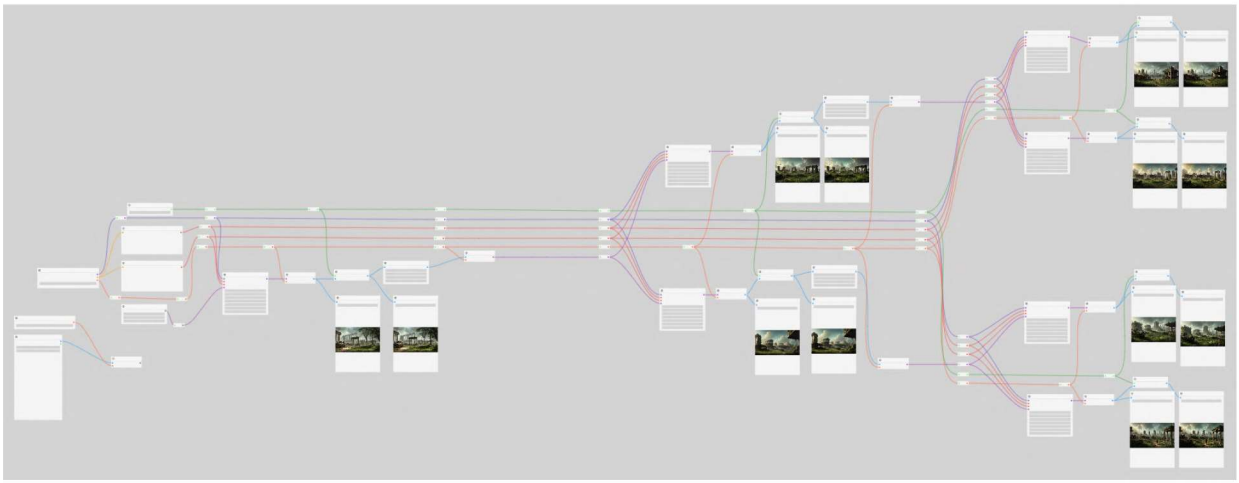
(a) 1080p slika



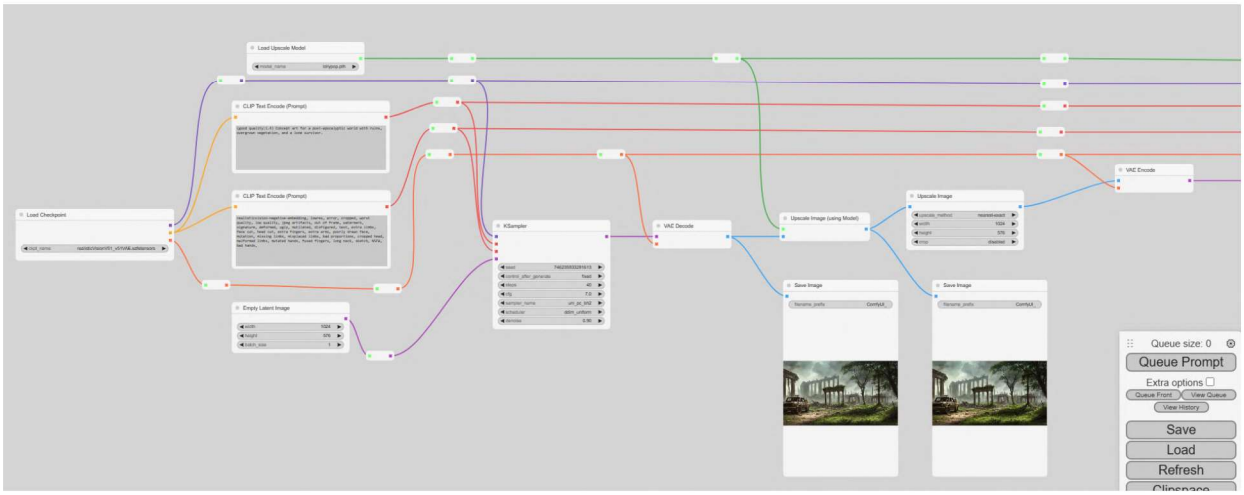
(b) 8K slika

Slika 32: Usporedba normalne i povećane slike

Za kraj ćemo pokazati jedan kompliciraniji workflow za poboljšanje kvalitete slike. Workflow se nakon svakog modela grana na još dva. Prvi model primi tekst i vrati sliku, zatim se ta slika uveća Lollypop modelom, pretvori natrag u latentni prostor manje dimenzije i prosljeđuje se u iduća dva modela. Time dobijemo slike visoke kvalitete i malih varijacija.



Slika 33: Razgranati upscale workflow



Slika 34: Bliži prikaz razgranatog upscale workflow-a

Slika 35: Razgranati upscale slika 1

Slika 36: Razgranati upscale slika 2

Slika 37: Razgranati upscale slika 3

Slika 38: Razgranati upscale slika 4

Slika 39: Razgranati upscale slika 5

Slika 40: Razgranati upscale slika 6

4. Zaključak

U ovom radu smo opisali *text-to-image* modele i objasnili na koji način je došlo do njih. Nakon toga smo preciznije opisali način rada i matematičku pozadinu Stable Diffusion modela i pomoću ComfyUI grafičkog sučelja pokazali samu arhitekturu modela i generiranje slike pomoću njega. Nakon jednostavnog primjera prošli smo kroz neke od najpopularnijih obrada slika, prije i poslije samog generiranja, kako bismo maksimizirali kvalitetu rezultata. Možemo zaključiti da podijela na razne modele koji upravljaju slikom i samim modelom pozitivno utječu na kvalitetu i brzinu generiranja slika.

Literatura

[1] Jonathan Ho, Ajay Jain i Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. URL: https://arxiv.org/abs/2006.11239.

[2] *How does Stable Diffusion work?* 2023. URL: https://stable-diffusion-art.com/how-stable-diffusion-work/.

[3] Elman Mansimov i dr. *Generating Images from Captions with Attention*. 2015. URL: https://arxiv.org/abs/1511.02793.

[4] Norman A. Rink i dr. *Memory-efficient array redistribution through portable collective communication*. 2021. URL: https://arxiv.org/abs/2112.01075.

[5] Olaf Ronneberger, Philipp Fischer i Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. URL: https://arxiv.org/abs/1505.04597v1.

[6] Lilian Weng. *What are Diffusion Models?* 2021. URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.