VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

UMĚLÁ NEURONOVÁ SÍŤ PRO GENEROVÁNÍ VĚROHODNÉHO PALEO-ARTU

A GENERATIVE NEURAL NETWORK FOR ACCURATE PALEO-ART

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Kryštof Havrda

AUTHOR

VEDOUCÍ PRÁCE — Ing. Dominik Řičánek

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Kryštof Havrda ***ID:*** 230070

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Umělá neuronová síť pro generování věrohodného paleo-artu

POKYNY PRO VYPRACOVÁNÍ:

1.Proveďte rešerši modelů umělých neuronových sítí a vyberte nejvhodnější pro rekonstrukci.
2.Vytvořte trénovací množinu pro vybranou síť.
3.Naučte vybranou neuronovou síť na zvolených datech.
4.Porovnejte vygenerované rekonstrukce tvorů s profesionálními rekonstrukcemi.

DOPORUČENÁ LITERATURA:

Hierarchical Text-Conditional Image Generation with CLIP Latents, https://doi.org/10.48550/arXiv.2204.06125

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 22.5.2024

Vedoucí práce: Ing. Dominik Řičánek

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Abstrakt práce v originálním jazyce

KLÍČOVÁ SLOVA

Klíčová slova v originálním jazyce

ABSTRACT

Překlad abstraktu (v angličtině, pokud je originálním jazykem čeština či slovenština; v češtině či slovenštině, pokud je originálním jazykem angličtina)

KEYWORDS

Překlad klíčových slov (v angličtině, pokud je originálním jazykem čeština či slovenština; v češtině či slovenštině, pokud je originálním jazykem angličtina)

HAVRDA, Kryštof. *Umělá neuronová síť pro generování věrohodného paleo-artu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024, 39 s. Bakalářská práce. Vedoucí práce: Ing. Dominik Řičánek

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Kryštof Havrda
VUT ID autora:	230070
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Umělá neuronová síť pro generování věrohodného paleo-artu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Dominiku Řičánkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Tato bakalářská práce se zabývá umělými neuronovými sítěmi. Tyto sítě dosáhly v současné době značného rozvoje a neustále se vyvíjí a zdokonalují. Od primitivních klasifikátorů, které nebyly schopné řešit složitější úlohy, se postupně vyvinuly tak, že v současné době umí vytvořit výstupy, které jsou v podobě reálného obrázku, a to na základě slovního zadání. Umělá neuronová síť je jeden z výpočetních modelů používaných v oblasti umělé inteligence. Vzorem je chování neuronů v mozku, od toho je odvozen i název. Skládá se z umělých neuronů, které se zakládají na biologickém neuronu. Umělé neurony jsou podobně jako biologické neurony vzájemně propojeny synaptickými vazbami a navzájem si tak předávají signály a transformují je pomocí aktivačních přenosových funkcí. Neuron má libovolný počet vstupů, ale pouze jeden výstup.

Umělé neuronové sítě nacházejí uplatnění v řadě odvětví, protože dokáží pracovat například s akustickými (rozpoznávání řeči) i elektrickými signály (EKG, EEG) nebo i s pixely (generování obrazu). Právě možnost generování obrazu je velmi důležitá pro tuto práci, jelikož se snažím doučit generativní model názvy a vzhled již vymřelých druhů aby bylo možné takto doučený model následně použít jako inspiraci pro umělce, kteří se zabývají tvorbou paleo-artu.

V první části práce se seznámíme s vývojem modelů umělých neuronových sítí a to především proto, abychom si utvořili obrázek o možnostech jednotlivých známých druhů sítí a o tom jak se zlepšovali možnosti jejich použití. Dostaneme se od základů umělé inteligence až k pokročilým moderním modelům umělých neuronových sítí, které umí generovat velmi věrohodné reálné výstupy. V tomto přehledu uvidíme možnosti i nedostatky některých dosud známých umělých neuronových sítí. Budeme se zde také soustředit na porovnání výstupů, a to především u pokročilých umělých neuronových sítí. Právě jedna z těchto pokročilých sítí bude předmětem této práce. Jde o Stable Difusion.

V druhé části této práce se tedy budu už soustředit na samotné programové řešení. Zde nejprve musím zvážit, kterou neuronovou síť bude možné použít pro generování obrázků dle zadání této bakalářské práce. Vyberu skupinu pokročilých sítí difúzní modely a poté vyberu konkrétní generativní model z již volně dostupných naučených modelů. Vysvětlím, proč jsem nevytvořit vlastní model.

Pro práci je také velmi důležité seznámit se s programovým řešení neuronových sítí.

Následně popíšu trénování vlastního generativního modelu, jak použít kód pro doučování difúzních modelů a jak je model potřeba naučit pro mé zadání. Cílem učení neuronové sítě je nastavit síť tak, aby dávala co nejpřesnější výsledky. Proto se v další části práce zabývám právě tím, jak model správně učit, že je potřeba

vytvořit dataset (trénovací množinu) a jakými kroky postupovat. Vybraný model, který je naučený na jiná data musím doučit mnou zvolená data. Pro vytvoření modelu musím vybrat druhy prehistorických tvorů, které se budu snažit modelem generovat. Popisuji postup, jak jsem vybíral vhodné obrázky pro trénovací datasety. ¨ Poté popisuji práci na úpravě kódu, aby byl použitelný pro moje účely. Následuje doučování modelu a podrobnější popis kolizí a cesty k uspokojivým vygenerovaným výstupům i s příklady obrázků pro lepší představivost a porovnání vývoje práce. Trénuji výstupy a snažím se získat co nejlepší rekonstrukce.

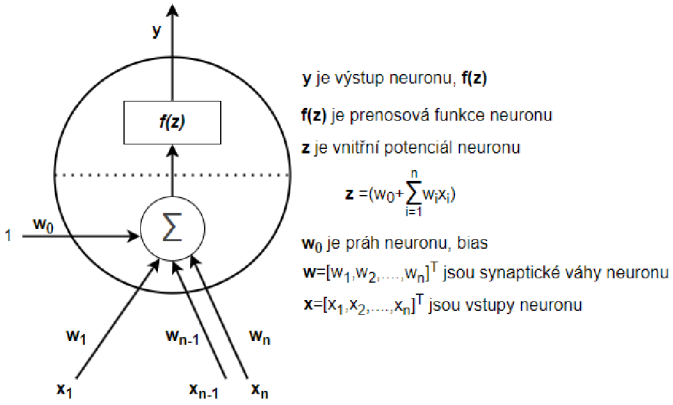
Tím se dostávám k poslední kapitole, kde porovnávám rekonstrukce tvorů generované mnou naučeným modelem s rekonstrukcemi profesionálních generativních modelů a hodnotím celou moji práci.

1 Teoretická část

V následující sekci se podíváme na jednotlivé komponenty, které dohromady tvoří generativní modely, jako například difuzní modely (DM).

1.1 Perceptron

Perceptron je jedním z nejjednodušších typů umělých neuronových sítí. Jedná se o jednovrstvou neuronovou síť, kterou lze použít pro klasifikační úlohy za předpokladu že třídy dat jsou lineárně separabilní. Hlavní myšlenkou perceptronu je napodobení způsobu, jakým pracuje biologický neuron v mozku.



Obr. 1.1: Perceptron s n vstupy a jedním prahem

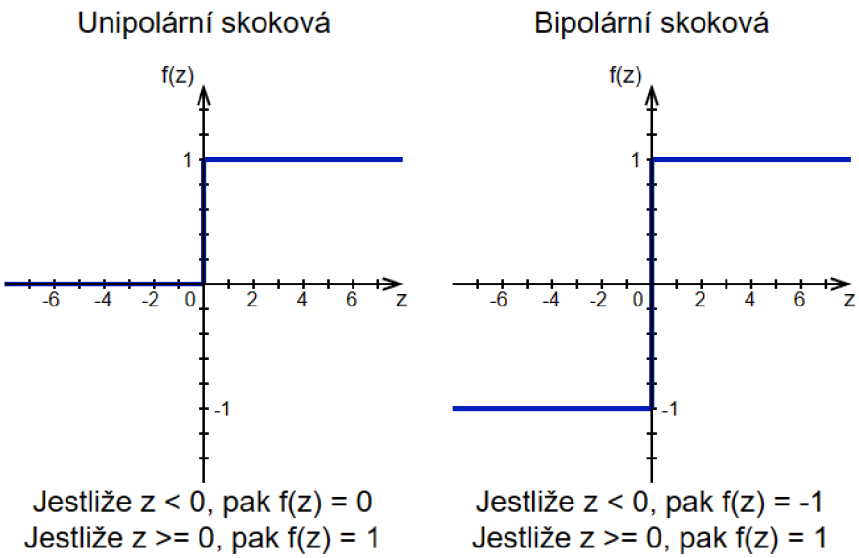
1.1.1 Popis perceptronu

Vstupní vrstva

Vstupní vrstva přijímá vstupní hodnoty. Každý vstup se vynásobí odpovídající váhou. Tyto vážené vstupy se poté sečtou.

Přenosová funkce

Součet vážených vstupů je pak předán přenosové funkci, která na základě výsledné sumy určí jaký bude výstup perceptronu. Mezi přenosové funkce pro perceptrony patří skoková funkce unipolární nebo bipolární a s prahem nebo bez prahu.



Obr. 1.2: Příklad skokových přenosových funkcí, které se používají v perceptronu a nevyužívají práh neuronu (bias)

Výstup

Výstupem perceptronu je výstup přenosové funkce. Pro unipolární skokovou přenosovou funkci bude výstup 0 nebo 1, pro bipolární skokovou přenosovou funkci bez prahu bude výstup -1 nebo 1, a pro bipolární skokovou přenosovou funkci s prahem může být výstup -1, 0 nebo 1.

Učení

Perceptron se učí změnou vah na základě chyby mezi výstupem z neuronu a požadovaným výstupem.

Rovnice pro výpočet nových hodnot vah w_i pro daný tréninkový vzor pro každé $i : 1 <= i <= n$:

pokud y=0 a $y_p = 1$ $w_i(t+1) = w_i(t) + x_i$

pokud y=1 a $y_p = 0$ $w_i(t+1) = w_i(t) - x_i$

pokud $y = y_p$ $w_i(t+1) = w_i(t)$

x_i - vstup neuronu, w_i - váha neuronu

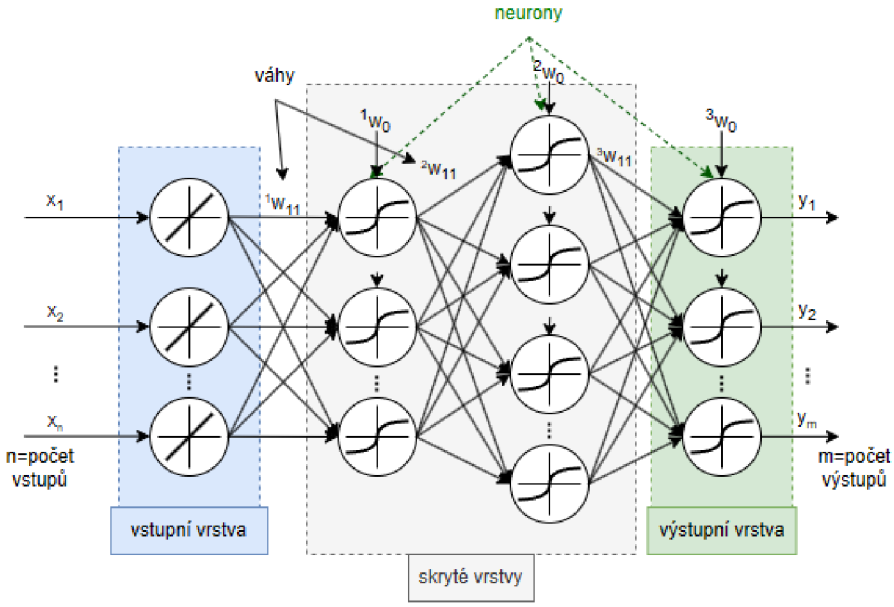
y - výstup neuronu, n - počet vstupů

x_p - požadovaný výstup neuronu, t - iterace učení

Proces učení opakovaně prochází trénovací sadu dat, dokud model nedojde k vahám, které přesně klasifikují trénovací data.

1.2 Vícevrstvá perceptronová síť (MLP)

Vícevrstvá perceptronová síť je typ umělé neuronové sítě, která se skládá z několika vrstev neuronů, vstupní vrstvy, jedné nebo více skrytých vrstev a výstupní vrstvy.



Obr. 1.3: Vícevrstvá perceptronová síť s n vstupy a m výstupy

1.2.1 Popis MLP

Vstupní vrstva

Vstupní vrstva se skládá ze vstupních neuronů, které mají lineární přenosovou funkci. Každý neuron reprezentuje jeden parametr vstupních dat.

Skryté vrstvy

Skryté vrstvy jsou vrstvy neuronů mezi vstupní a výstupní vrstvou. Každý neuron ve skryté vrstvě přijímá vstupy od neuronů v předchozí vrstvě, každý vstup vynásobí příslušnou váhou a vážený součet propustí přes přenosovou funkci. Skryté vrstvy umožňují síti naučit se komplexní reprezentaci vstupních dat.

Výstupní vrstva

Výstupní vrstva se skládá z výstupních neuronů, které vytvářejí konečný výstup sítě. Počet výstupních neuronů závisí na povaze úlohy kterou řešíme. V případě, že například chceme klasifikovat více tříd, použijeme pro klasifikaci čísel od 0 do 9

celkem 10 výstupních neuronů, kde každý reprezentuje pravděpodobnost příslušnosti k danému číslu.

Přenosová funkce

Každý neuron, kromě vstupních neuronů, obvykle aplikuje přenosovou funkci na vážený součet svých vstupů, aby se do sítě vnesla nelinearita. Mezi běžně používané přenosové funkce patří sigmoida, tanh a funkce ReLU (Rectified Linear Unit).

Učení

MLP se trénují pomocí algoritmů učení s učitelem, který se jmenuje algoritmus backpropagation. Backpropagation spočívá v iterativním nastavování vah spojů mezi neurony v síti tak, aby se minimalizoval rozdíl mezi požadovanými výstupy a skutečnými výstupy sítě.

Loss Function (Ztrátová funkce)

Během trénování se rozdíl mezi požadovanými výstupy a skutečnými výstupy kvantifikuje pomocí Loss funkcí, jako je cross-entropy loss pro klasifikační úlohy, nebo Mean Squared Error (MSE) pro regresivní úlohy. Cílem trénování je minimalizovat tuto loss funkci.[1] [2]

Backpropagation

Backpropagation je iterativní gradientní algoritmus učení, který minimalizuje čtverce chybové funkce E. Chyba sítě E je rozdíl mezi skutečnými a požadovanými hotnotami výstupů neuronové sítě pro danou tréninkovou množinu. Jakmile provedeme výpočet ve výstupní vrstvě, šíříme chybu zpětně sítí, vrstvu po vrstvě. Klíčovým výpočtem při zpětném průchodu je určení gradientů pro každou váhu a bias v síti. Tento gradient nám říká, o kolik by se měla každá váha/bias upravit, aby se minimalizovala chyba při dalším průběhu sítě. K efektivnímu výpočtu tohoto gradientu se iterativně používá řetězové pravidlo. K efektivním úpravám vah sítě během trénování se používá optimalizační technika zvaná gradientní sestup. Gradientní sestup je iterační algoritmus pro nalezení lokálního minima vícerozměrné funkce.

Rovnice Backpropagation:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \tag{1.1}$$

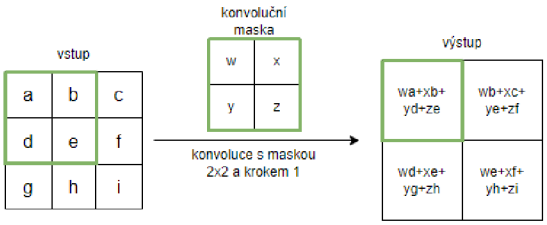
net_j - vstup do aktivační funkce
o_j - výstup z aktivační funkce
w_{ij} - jsou váhy modelu

1.3 Konvoluční neuronové sítě [CNN]

Konvoluční neuronové sítě jsou speciální třidou neuronových sítí, které se specializují na zpracování dat které mají strukturu mřížky jako jsou obrázky. Používají se třeba pro aplikace které mají za úkol klasifikovat objekty na fotkách.[3] CNN se skládá z několika častí, konvolučních vrstev, pooling vrstev a plně propojených vrstev.

Konvoluční vrstva

Používá masku(jádro/kernel) s kterou prochází vstupní obraz. Tato maska aplikuje na vstup matematickou operaci konvoluce. Konvoluce obrazového vstupu probíhá tak, že vezmeme masku například 2x2 a přiložíme ji na vstupní obraz. Každá hodnota pixelu vstupu se pronásobí s příslušnou vahou masky a provede se suma těchto součinů, výsledek této operace nám dá jeden pixel na výstupu.



Obr. 1.4: Příklad konvoluce pro vstupní obrázek velikosti 3x3 a konvoluční masku velikosti 2x2 která se pohybuje s krokem jedna

Výhodou konvoluce oproti jednoduché neuronové síti je to, že pokud máme obrázek o rozměrech 256x256, tak u neuronové sítě by jsme potřebovali měnit 65536 vah a biasů, kdežto konvoluční vrstva využívá stejnou masku pro procházení celého obrázku, tudíž pro masku velikosti 3x3 musíme měnit pouze 9 vah a jeden bias.

Konvoluce extrahuje různé rysy vstupního obrázku. Mezi rysy, které jsou pro člověka lehce pochopitelné, patří třeba výstup pro masku která detekuje hrany. Při reálném použití si CNN sama mění váhy v masce tak, aby došla k požadovaným výsledkům. To způsobí to, že pokud se člověk podívá na výstupy jednotlivých vrstev, tak mu nemusí dávat smysl.

Rovnice pro výpočet redukce rozměrů po provedení konvoluce:

$$(N - M)/stride + 1 \tag{1.2}$$

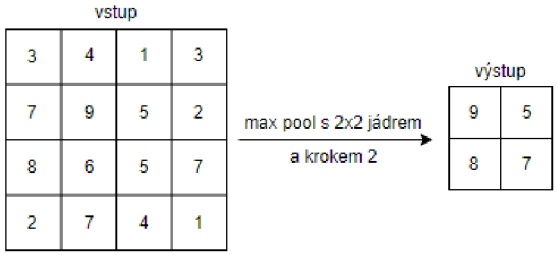
Kde N je délka hrany buněk ve vstupu, M je velikost masky a stride je velikost kroku masky. Příklad: Máme obraz o velikosti 7x7, masku velikosti 3x3 a stride 1:

$$(7 - 3)/1 + 1 = 5 \tag{1.3}$$

Výsledkem bude obraz velikosti 5x5.

Pooling vrstva

Pooling vrstva provádí redukci vstupu a tím přichází o část jeho informační hodnoty, U většiny CNN se jedná o max-pooling, který používá masku 2x2 s krokem 2, čímž se sníží velikost na čtvrtinu velikosti vstupu. Je možné využít i jiného rozměru masky a jiného kroku.



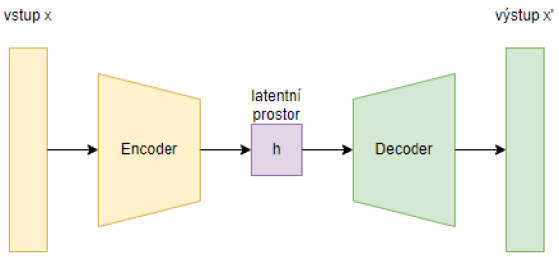
Obr. 1.5: Příklad max-poolingu s velikostí jádra 2x2 které se pohybuje s krokem 2. Z obrázku je vidět že při každém kroku max-pooling vybere nejvyšší hodnotu v dané oblasti a promítne ji na výstup

Plně propojené vrstvy

Plně propojené vrstvy mají stejnou funkci jako MLP, které se snaží klasifikovat do jednotlivých tříd.

1.4 Autoenkodér

Jedná se o druh neuronové sítě, která se skládá ze dvou části encodéru a decodéru. Úkolem encodéru je vzít vstup x (může se jednat o jakýkoliv typ dat: text, 2D nebo 3D obraz/video atd.) a tyto data namapovat do latentního prostoru. Dekodér pak bere tento latentní prostor a snaží se rekonstruovat původní data x takže na výstupu dostaneme data x'. Cílem autoenkodéru je provedení rekonstrukce vstupních dat co možná nejpřesněji. Pro učení autoenkodéru se používá loss funkce $= ||x - x'||^2$. [3]



Obr. 1.6: Autoenkodér se skládá z Enkodéru a Dekodéru

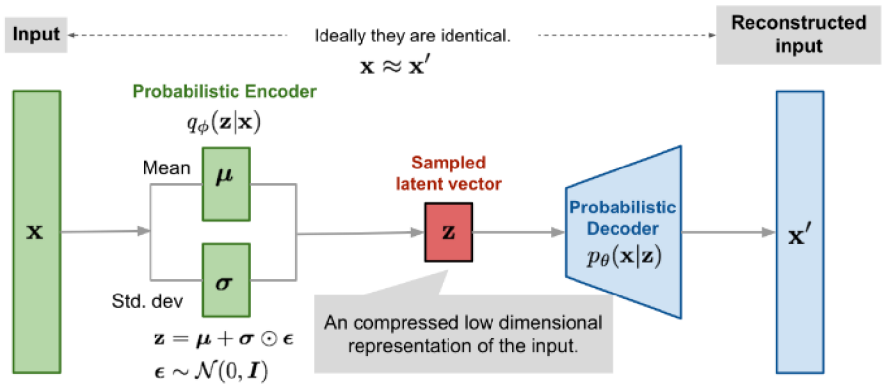
Při učení autoenkodéru se snažíme aby latentní prostor získal užitečné vlastnosti, nejen aby autoenkodér kopíroval data ze vstupu na výstup. Můžeme toho dosáhnout tak, že latentní prostor h má menší rozměr než vstup x, autoenkodér je poté nucen naučit se nejvíce podstatné vlastnosti trénovacích dat.[3]

1.5 Variační autoenkodér (VAE)

Variační autoenkodér neboli VAE je řízený model, který využívá naučenou aproximativní inferenci a lze jej trénovat čistě metodami založenými na gradientu.[3]

VAE je autoenkodér, jehož kódovací rozložení je během trénováni regulováno, aby bylo zajištěno, že jeho latentní prostor má dobré vlastnosti, které nám umožní generovat některá nová data. Termín "variační"navíc pochází z úzkého vztahu, který existuje mezi regularizační a variační inferenční metodou ve statistice.[4]

1.5.1 Jak funguje VAE



Obr. 1.7: variačního autoenkodérový model s vícerozměrným Gaussovým předpokladem [5]

Pro vytvoření vzorku z modelu VAE nejprve vybere vzorek z z rozdělení kódu $p_{model}(z)$. Vzorek je poté prohnán diferencovatelnou generátorovou sítí g(z). Nakonec je z rozdělení $p_{model}(x; g(z)) = p_{model}(x|z)$ vybrán vzorek x. Během trénování se však k získání z používá přibližná odvozovací síť (nebo kodér) $q(z|x)$ a na $p_{model}(x|z)$ se pak pohlíží jako na dekódovací síť. Klíčovým poznatkem variačních autoenkoderů je, že je lze trénovat maximalizací variační dolní meze $L(q)$ spojené s datovým bodem x.

$$L(q) = E_{z\text{-}q(z|x)} log\ p_{model}(x|z) - D_{KL}(q(z|x)||p_{model}(z)) \tag{1.4}$$

V rovnici 1.1 rozpoznáme první člen jako rekonstrukční logaritmus pravděpodobnosti, který se vyskytuje v jiných autoenkodérech. Druhý člen se snaží dosáhnout

toho, aby se přibližné následné rozdělení q(z | x) a priorita modelu p model(z) navzájem přiblížily.[3]

1.6 Transformátory

Transformátory jsou neuronové sítě skládající se z jiných neuronových sítí. Jimiž jsou tokenizér, attention layer a MLP. Tokenizér je použit na vstupu transformátoru. Attention layer a MLP se střídavě opakují dokud nedojdeme k požadovanému výsledku.

Tokenizér

Transformátory využívají tokenizér, aby si vstupní data rozdělili na menší části takzvané tokeny. V případě textového vstupu si text rozdělí na části slov, v případě obrazového vstupu si může daný obraz rozdělit na skupiny pixelů. Tokenům je poté přiřazena číselná hodnota, v případě rozdělování textu může tokenizér přidělit každému tokenu embedding vektor čísel z embedding matice. Tyto vektory poté tokenizér předává do attention vrstvy. Tokenizér je nutnou součástí transformátorů jelikož zajišťuje převod z textu na čísla s kterými neuronové sítě pracují.

Attention vrstva

Tato vrstva upravuje jednotlivé vektory v závislosti na ostatních vektorech. Embedding vektory jednotlivých tokenů se vynásobí s maticí vah W_Q, tím získáme takzvaný query vektor Q_i. Query vektor si můžeme představit jako otázku. Zároveň se Embedding vektory vynásobí s maticí vah W_K, čímž získáme key vektor K_i, který si můžeme představit jako odpovědi na otázky query vektoru. Rovnice pro Self-attention:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1.5}$$

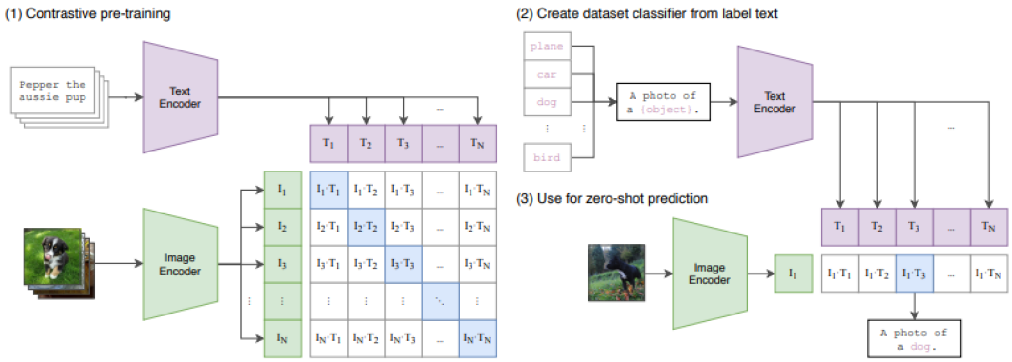
Softmax je funkce která zajištuje, že suma hodnot bude rovna jedné. d_k reprezentuje rozměr query a key vektorů.[6] Vektor hodnot V vznikne vynásobením Embeding vektorů s maticí vah W_V.

1.7 State of the art v generování obrázků

V této kapitole si probereme pokročilé generativní modely. Tyto modely už lze s uspokojivým výsledkem využít pro generování věrohodného obrázku.

1.7.1 CLIP (Contrastive Language-Image Pretraining)

Příchod modelu CLIP od OpenAI způsobil nárůst text-to-image modelů. CLIP je naučen na velkém množství dat tvaru obrázek + popis obrázku, které byli stažené z internetu. Když již naučenému CLIPu dáme obrázek a nějaký popis obrázku, CLIP nám ohodnotí s jakou pravděpodobností k danému obrázku patří tento popis.[7][8] CLIP byl vytvořen, abychom se vyhnuli několika hlavním problémům se kterými se



Obr. 1.8: CLIP společně trénuje encodér pro obraz i text, aby správně předpovídal příslušnost obrázku k danému popisu. [7]

setkáváme u modelů hlubokého učení. Mezi tyto problémy patří například nákladné vytváření použitelných datasetů, nebo velmi specifické zaměření modelů, které fungují dobře pouze na datech, která jsou podobná datům na kterých se model učil. Nákladné vytváření použitelných datasetů bylo vyřešeno stahováním veřejně dostupných dat z internetu. Místo specifického zaměření u modelů hlubokého učení, kde docházelo k tomu, že uměly pracovat pouze s daty na kterých se učily, můžeme CLIP použít na různé úlohy klasifikace bez potřeby doučení na nových trénovacích datech. Pro generování obrázků využíváme CLIP propojený s některým z generativních modelů (VAE, GAN, difuzní modely, atd.). Propojení CLIP a difúzního modelu využívá takzvaný unCLIP.[7][8]

1.7.2 Difuzní model

Difuzní modely jsou generativními modely které se používájí pro generování dat, které se podobají datům, na kterých byl difůzní model trénován. Difuzní modely k tréninkovým datům postupně přidávají šum a následně se učí jak obrátit tento proces a odstranin šum z dat tak aby znovu vytvořili původní data. Difuzní modely v současné době poskytují vysoce kvalitní obrazy (State-of-the-Art).[9] [10]

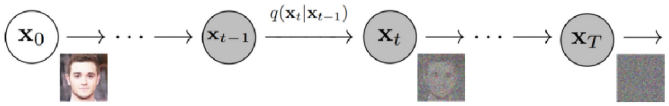
Obr. 1.9: Difuzní model postupné přidání šumu a nasledné odebrání při opačném postupu[9]

Obr. 1.10: Přiklady obrazku generovaních difuznimi modely [10]

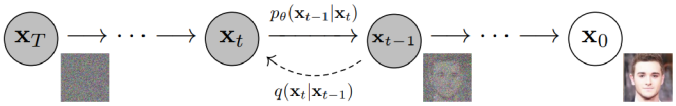
Jak funguje difuzní model

Difůzní model používá markovský řetězec aby postupně mapoval šum do latentního prostoru, který je přidáván k datům. S cílem získat další stav $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ kde $\mathbf{x}_1, ..., \mathbf{x}_T$ jsou latentní proměnné se stejnou dimenzí jako $\mathbf{x}_0$.[10]

Na konci tohoto procesu je obraz převeden na čistý Gaussův šum. Cílem trénování difuzního modelu je naučit se opačný proces: $p_\theta(x_{t-1}|x_t)$ Postupem zpět po tomto řetězci můžeme generovat nová data.[10]



Obr. 1.11: Markovův řetězec zobrazený pro obrazová data[10]



Obr. 1.12: Markovův řetězec zobrazený pro obrazová data v opačném směru[10]

Trénování difuzního modelu

Difuzní model se trénuje nalezením reverzních Markovových přechodů, které maximalizují pravděpodobnost trénovacích dat. Trénování spočívá v minimalizaci L_{vlb} (Variational uper Bound což je záporná hodnota ELBO(Evidence Lower Bound)) záporné logaritmické pravděpodobnosti.[10]

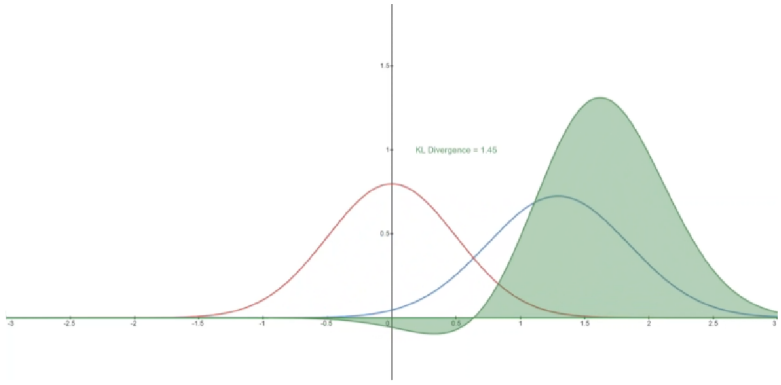
$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L_{vlb} \qquad [10]\ (1.5)$$

Je zde snaha přepsat L_{vlb} v podobě Kullback-Leiblerových (KL) divergencí. KL Divergence je asymetrická statistická míra vzdálenosti, která udává, jak moc se jedno pravděpodobnostní rozložení P liší od referenčního rozložení Q. Je zde snaha vyjádřit L_{vlb} formou KL divergencí, protože přechodová rozdělení v markovském řetězci jsou Gaussova rozložení a KL divergence mezi Gaussovými rozloženími má konečný tvar.[10]

KL divergence

Matematický tvar KL divergence pro spojité rozložení je následující:

$$D_K L(P||Q) = \int_{-\infty}^{\infty} \log(\frac{p(x)}{q(x)})\, dx \tag{1.6}$$



Obr. 1.13: KL divergence, proměnného rozdělení P (modrá) od referenčního rozdělení Q (červená). Zelená křivka označuje funkci v rámci integrálu který je uveden v rovnici (1.6) KL divergence a celková plocha pod křivkou představuje hodnotu KL divergence P od Q v daném okamžiku.[10]

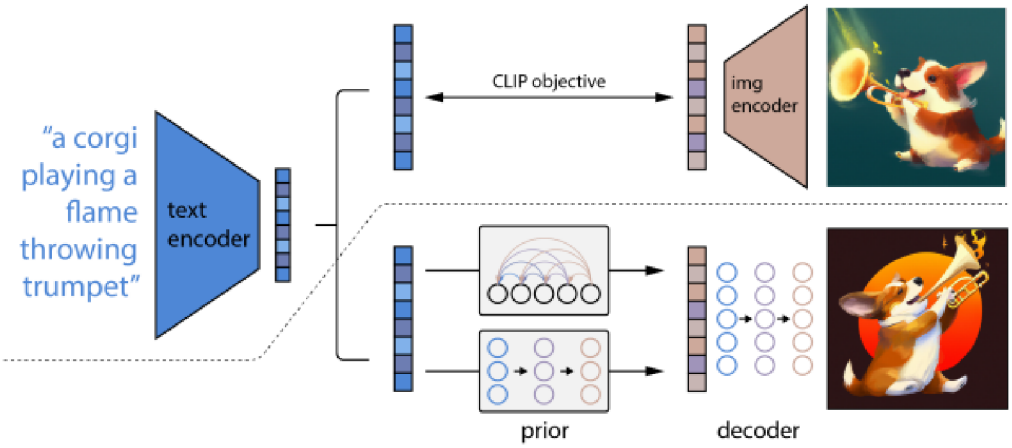
1.7.3 unCLIP

Model unCLIP propojuje vlastnosti CLIPu a difuzních modelů. Embedingy, které CLIP vytváří z textu, mají žádoucí vlastnosti a jsou vyladěné tak, že dosahují state-of-the-art kvality pro různé úlohy z oblasti počítačového vidění a jazykových modelů. Ve stejné době se objevily i difuzní modely, které posunuly kvalitu generování obrázků o krok kupředu.[11]

Metoda použitá pro unCLIP

Soubor trénovacích dat se skládá z dvojic (x, y) obrázků x a jim odpovídajících popisků y. Jeli dán obrázek x, tak z_i a z_t jsou jeho obrazové a textové embedingy CLIPu. Model unCLIP je navržen tak, aby vytvářel obrázky z popisků pomocí dvou komponent:[11]

- prior $P(z_i|y)$ který produkuje CLIP embeddings obrázků z_i podmíněných popisky y.[11]
- dekodér $P(x|z_i, y)$ který vytváří obrazy x podmíněné vloženými embeddingy obrázků CLIPu z_i (a volitelně textové popisky y).[11]



Obr. 1.14: Přehled na vysoké úrovni projektu unCLIP. Nad přerušovanou čarou je znázorněn proces tréninku CLIP, při kterém se učí společnou reprezentaci textu a obrázku v latentním prostoru. Pod přerušovanou čarou je znázorněn model pro generování obraz z textu: textový embeding CLIPu je nejprve vložen do autoregresního nebo difuzního prioru, aby vytvořil embedding obrazu a poté je tento embedding použit k nastavení difúzního dekodéru, který vytvoří konečný obraz. Model CLIP je během trénování priorů a dekodéru zmrazen.[11][9]

Dekodér nám umožňuje vytvářet obrazy vzhledem k jejich obrazovým embeddingům clipu, zatímco prior nám umožňuje naučit se samotné embeddingy obrázků. Spojením těchto dvou složek získáme generativní model $P(x|y)$ obrázků x vzhledem k popiskům y:[11]

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)P(z_i|y). \tag{1.7}$$

První rovnost platí, protože z_i je deterministickou funkcí x. Druhá rovnost platí, protože platí pravidlo o derivaci složené funkce. Může se tedy vzorkovat z pravého podmíněného rozdělení $P(x|y)$ tak, že nejprve odebereme vzorek z_i pomocí prioru a poté odebereme vzorek x pomocí dekodéru.[11]

Dekodér

Pro vytvoření obrázků podmíněných obrazovými embedingy CLIPu je použit difuzní model.[11] Ačkoli by bylo možné vzorkovat přímo z podmíněného rozdělení dekodéru, předešlé práce využívající difuzních modelů ukázaly, že použití vodítek na základě podmíněných informací zvyšuje kvalitu vzorků.[11]

Prior

I když dekodér může invertovat obrazové embeddingy CLIPu z_i k vytvoření obrazů x. Tento model ještě potřebuje prior, který vytvoří z_i z popisků obrázku y, aby bylo možné generovat obrázky z textových popisků. Pro prior je možné použít dvě ruzné třídy modelů.:[11]

- Autoregresní (AR) prior: embeddingy obrazu CLIP z_i se převedou na posloupnost diskrétních hodnot a předpovídá se autoregresivně na základě popisku y.[11]
- Difuzní prior: Spojitý vektor z_i je přímo modelován pomocí Gaussova difuzního modelu na základě popisku y.[11]

Kromě popisku můžeme prior podmínit pomocí textového embedingu CLIPu z_t, protože se jedná o deterministickou funkci popisku. [11]

1.7.4 DALL · E

DALL-E je verze GPT-3 s 12 miliardami parametrů, která je naučená generovat obrázky z textových popisů pomocí datové sady dvojic text-obrázek. Ukázalo se, že má rozmanité schopnosti, včetně vytváření antropomorfizovaných verzí zvířat a objektů, kombinování nesouvisejících pojmů věrohodnými způsoby, vykreslování textu a použití transformací na existující obrázky.[12]

Schopnosti

V OpenAI zjistili , že DALL-E je schopen vytvořit věrohodné obrazy pro velké množství vět, které zkoumají kompoziční strukturu jazyka. [12]

1.7.5 Midjourney

Midjourney je nezávislá výzkumná laboratoř, která vytváří vlastní program umělé inteligence pod stejným názvem, který vytváří obrázky z textových popisů, podobný DALL · E od OpenAI. Midjourney využívá algoritmů strojového učení a je natrénováno na velkém počtu dat. Momentálně je tento nástroj na tvorbu obrázků z textu dostupný pouze prostřednictvím platformy Discord.

Porovnání DALL · E a Midjourney

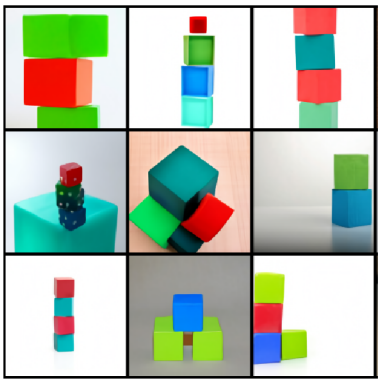
V OpenAI zjistili že DALL-E je schopen nakreslit více kopií objektu, když je k tomu vyzván. Když je vyzván, aby nakreslil podstatná jména, pro která existuje více významů, například "glasses", "chips"a "cups", někdy nakreslí obě interpretace v závislosti na použitém tvaru množného čísla.[12] Vidíme ale, že jednotlivé obrázky

(a) DALL · E[12]

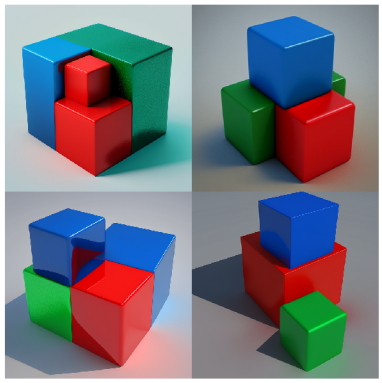
(b) Midjourney

Obr. 1.15: Řízení atributů-zadání: "a collection of glasses is sitting on a table"[12]

vždy obsahují pouze jednu z interpretací kdežto MJ v jednom obrázku zobrazuje obě interpretace daného slova



(a) DALL · E[12]



(b) Midjourney

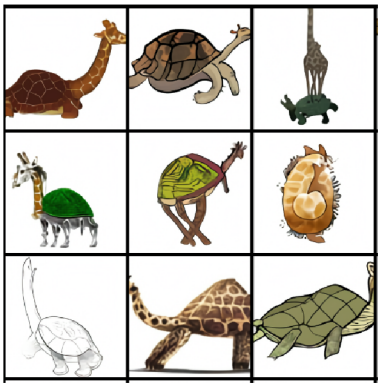
Obr. 1.16: Kreslení několika oběktů-zadání: "a stack of 3 cubes. a red cube is on top,sitting on a green cube. the green cube is in the middle, sitting on a blue cube. the blue cube is on the bottom"[12]

Současné ovládání více objektů, jejich atributů a prostorových vztahů představuje novou výzvu. Pro správnou interpretaci takových vět musí DALL-E a Midjourney nejen správně skládat jednotlivé části (př. oblečení se zvířetem), ale také vytvářet asociace [př.(čepice, červená), (rukavice, žlutá), (košile, modrá) a (kalhoty, zelená)] aniž by je zaměnil.[12] V obrázku 1.16 vidíme že s tímto druhem zadaní má DALL · E i Midjurney problémy a nedodržuje správné barvy kostek a ve většině případů ani správný počet.

Schopnost kombinovat nesouvisející koncepty při generování obrazů. OpenAI tuto schopnost zkoumá v kontextu umění, a to pro ilustraci zvířecí chiméry. Zjis-

tili, že DALL · E je někdy schopen kombinovat různá zvířata věrohodnými způsoby. Obecně platí, že vlastnosti druhého zvířete uvedeného v titulku bývají dominantní.

Zjistili také, že vložení slovního spojení "professional high quality"před slovo "illustration"a podobné někdy zlepšuje kvalitu a konzistenci výsledků.[12] Z obrázku 1.17b můžeme vypozorovat že i Midjourney kombinuje různá zvířata poměrně obstojně. Navíce vytvořil kvalitnější a realističtější obrázky než DALL · E



(a) DALL · E[12]

(b) Midjourney

Obr. 1.17: Ilustrace neexistujících živočichů-zadání: "a professional high quality illustration of a giraffe turtle chimera. a giraffe imitating a turtle. a giraffe made of turtle"[12]

2 Programové řešení

2.1 Studium modelů a vytipování modelu vyhovujícího pro moji práci

Pro praktickou část bakalářské práce jsem si udělal rešerši modelů umělých neuronových sítí pro generování obrázků. Při porovnávání modelů jsem se soustředil především na jejich dokonalost a praktičnost, na kvalitu výstupu, zda budou vhodné pro moje zadání a jaká je obtížnost implementace. Zvažoval jsem možnosti, z nichž jednou bylo vytvořit vlastní model. Zde je úskalím, že bych potřeboval velmi velké množství dat, na kterých bych model mohl učit. Získání těchto dat by bylo časově velice náročné a tudíž nereálné. Následně bych si musel sám vytvořit celou neuronovou síť, kterou bych používal. Dalším navazujícím problémem je velikost úložiště v počítači pro ukládání velkého množství obrázků. Kvůli trénovacím datům je výhodné použít již naučený model a ten doučit má data. Pro moji práci jsem se nakonec rozhodl použít difúzní modely, protože jde o pokročilé modely, od kterých už lze očekávat uspokojivý výsledek a problém s velkým množstvím dat tím bude odstraněn.

2.2 Seznámení se s programovým řešením neuronových sítí

Pro programové řešení modelů hlubokého učení existuje v Pythonu knihovna PyTorch která umožňuje využívat sílu grafické karty a jiných akcelerátorů. Zároveň umožňuje provádění různých operací s tenzory a vytváření vlastních neuronových síti a následné trénování těchto síti pomocí ztrátových funkcí. Jelikož jsem se s touto knihovnou setkal poprvé právě u této práce, potřeboval jsem pochopit, jak tuto knihovnu používat. S tím mi pomohl tutoriál "Deep Learning with PyTorch: A 60 minute Blitz"[13]

2.3 Trénování vlastního generativního modelu

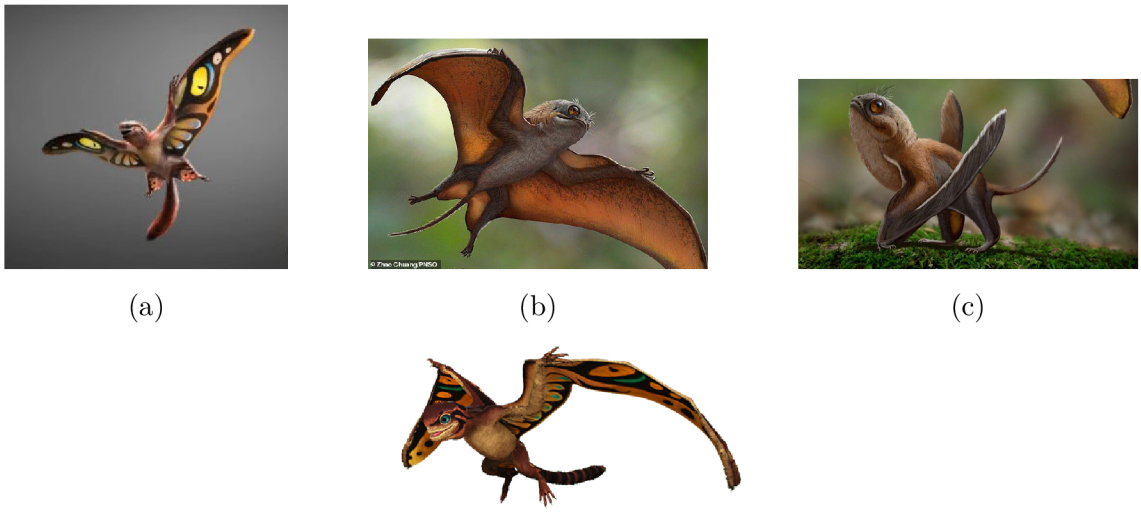
Jelikož jsem se teprve seznamoval s Pythonem a knihovnou PyTorch, bylo by pro mě obtížné vytvořit celou neuronovou síť, která by byla schopná generovat kvalitní obrázky. Nemluvě o tom, že pro vytvoření vlastního modelu bych potřeboval velké množství dat a učení by s mě dostupným výpočetním výkonem ani nebylo možné realizovat. Z těchto důvodů jsem se rozhodl využít kódu pro doučování difuzních

modelů [14], který jsem si upravil tak aby mi vyhovoval. Jako základní model, který doučuji používám "stable-diffusion-2"od stabilityai [15], který je volně dostupný na stránkách huggingface.

2.4 Vytvoření trénovacích datasetů

Pro vytvoření modelu který bude schopen generovat paleo-art jsem si musel vybrat které vymřelé druhy se budu model pokoušet naučit. Jako první druh jsem si vybral rod Sinomacrops bondei z čeledi Anurognathidae dále jsem si vybral ptakoještěry z čeledi Dimorphodontidae a jako třetí jsem si vybral Podřád ptakoještěrů Ornithocheiromorpha. Pro jednoduší schromáždění obrázků jsem použil image scraper od ultralytics [16] který jsem si v google colabu stáhnul na disk a spustil abych si stáhl potřebné obrázky. Po stažení všech obrázků bylo nutné je přetřídit jelikož se mezi nimy objevili obrázky koster nebo ptáků.

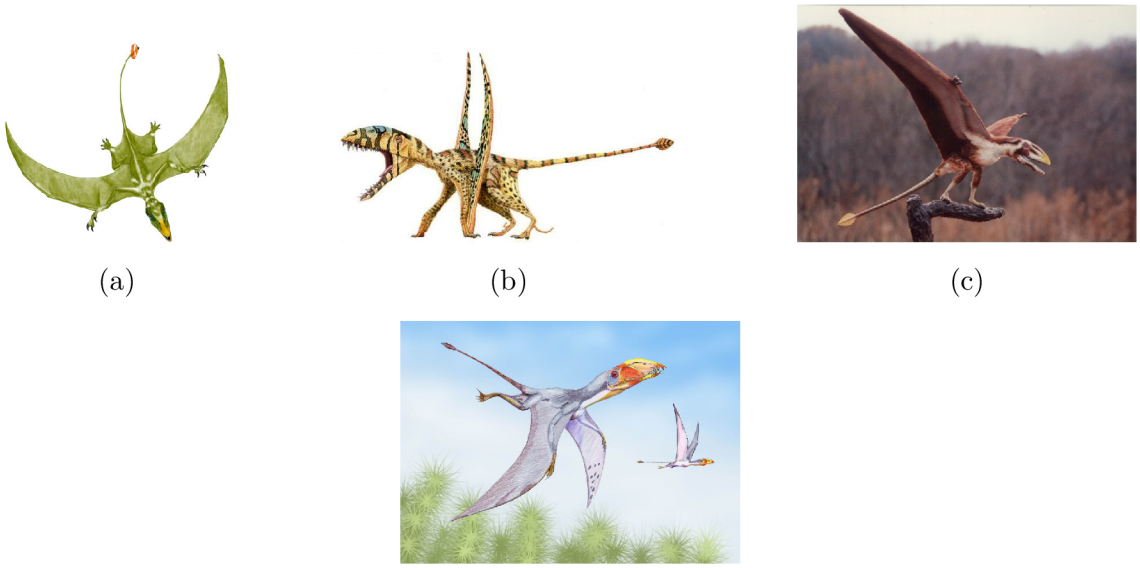
2.4.1 Ukázka obrázků využitých v jednotlivých trénovacích datasetech



(a) (b) (c)

(d)

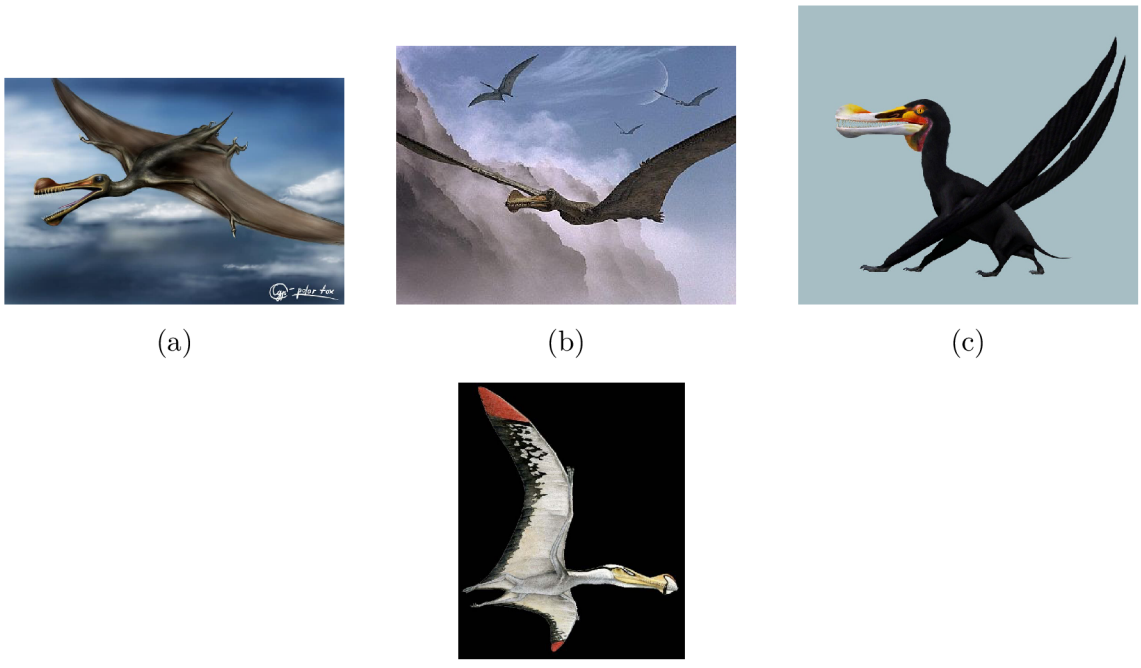
Obr. 2.1: Ukázka trénovacích obrázků pro Sinomacrops bondei



(a) (b) (c)

(d)

Obr. 2.2: Ukázka trénovacích obrázků pro Dimorphodontidae



(a) (b) (c)

(d)

Obr. 2.3: Ukázka trénovacích obrázků pro Ornithocheiromorpha

2.5 Úpravy kódu

Jelikož jsem měl trénovací datasety uložené na google disku, a kód od huggingface pro doučování difuzních modelů [14] načítal trénovací vzory přímo z internetové adresy na které se obrázek nachází. Bylo potřebné upravit kód tak aby byl schopen načíst obrázky přímo z google disku a následně s těmito obrázky pracovat. Pro připojení k disku jsem využil následující kód:

```
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/MyDrive
```

Pro ukládání modelu který byl doučen na má data jsem použil následující kód:

```
  from slugify import slugify
  from huggingface_hub import HfApi, HfFolder, CommitOperationAdd
  from huggingface_hub import create_repo
  from IPython.display import display_markdown
pipe = StableDiffusionPipeline.from_pretrained(
    args.output_dir,
    torch_dtype=torch.float16,
  ).to("cuda")
  os.makedirs("fp16_Sinomacrops_bondei",exist_ok=True)
  pipe.save_pretrained("fp16_Sinomacrops_bondei")
  output_dir = args.output_dir
```

2.6 Použití kódu pro doučení modelu

První věc kterou jsi v kódu musíme nastavit je cesta k již naučenému modelu který chceme použít.

```
pretrained_model_name_or_path = "stabilityai/stable-diffusion-2"
```

Je možné použití i jiného modelu ze stránek Hugging Face [17] nebo také nahrání modelu přímo z google disku. Dále si musíme zvolit jaký popisek si má model spojit s daty která ho budeme učit v mém případě se jednalo o název daného dinosaura. A zda chceme využít možnosti "prior preservation"Která by měla pomoci se zachováním třídy daného konceptu.

```
instance_prompt = "<Sinomacrops␣Bondei>␣Sinomacrops␣Bondei"
prior_preservation = False
prior_preservation_class_prompt = "Pteranodon"
```

Já jsem se rozhodl možnosti využití "prior preservation"nevyužít jelikož je učení poté výpočetně více náročné. Protože mám trénovací data uloženy na disku musím také správně nastavit cestu k nim v mém případě takto:

```
instance_data_dir="./google-images-download/images/sinomacrops",
```

2.6.1 Doučování modelu

Když jsem se seznamoval s kódem od huggingface pro doučování difuzních modelů [14] a testoval jak funguje. Tak jsem zkoušel model učit s nízkým počtem trénovacích kroků, jelikož už i učení s pouhými sto kroky trvalo přibližně deset minut. Brzy jsem ale zjistil že tento přístup je špatný, sice ano model jsem něco málo naučil ale obrázky které jsem dostal na výstupu vypadali jako něco úplně jiné.

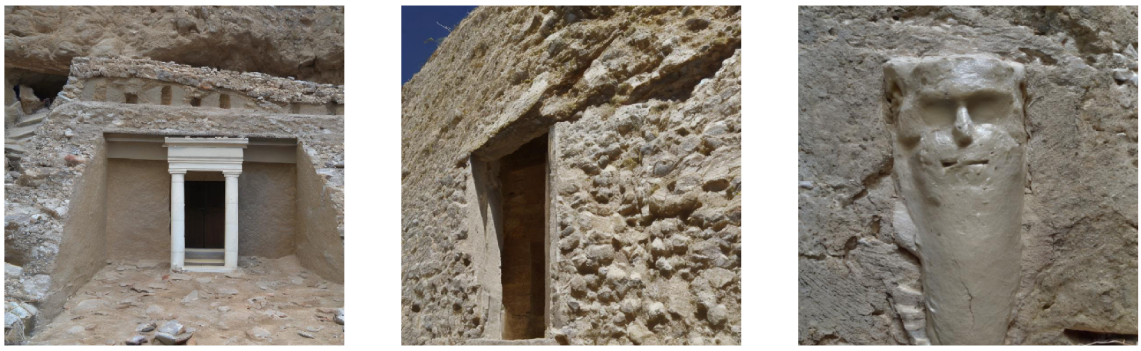
(a) (b) (c)

Obr. 2.4: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při padesáti krocích učení. S použitým promptem: "Sinomacrops bondei"

Vidíme že při pouze padesáti krocích učení dostáváme nesmyslné obrázky lidských postav, budov v poušti nebo třeba lebek a kostí v poušti vis. Obr.2.4. Tyto obrázky dostáváme z toho důvodu že původní model prompt Sinomacrops bondei spojuje právě třeba s budovami v pouští vis. Obr.2.5 a padesát kroků není dostatečné množství aby se model spolehlivě naučil nový koncept.

Překvapivě i když model nedokáže vygenerovat tohoto dinosaura samostatně když zadáme jeho jméno, je již schopen zachytit částečné jeho barvy případně to že má křídla vis. Obr.2.6. I když skutečnost že byli vygenerovány obrázky něčeho co vypadá jako že má křídla může být způsobeno použitím slova "flying"v promptu který byl použit pro vygenerování těchto obrázků.

Jelikož učení s pouze padesáti kroky bylo nedostačující, rozhodl jsem se zvýšit počet kroků učení na 250. v Obr.2.7 můžeme pozorovat značné zlepšení oproti prvním výsledkům které jsme získali při učení pouze s padesáti kroky. Stále ale toto



(a) (b) (c)

Obr. 2.5: Ukázka výstupu modelu před doučením na trénovacích datech. S použitým promptem: "Sinomacrops bondei"

(a) (b) (c)

Obr. 2.6: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při padesáti krocích učení. S použitým promptem: "Sinomacrops bondei flying in the forest"

učení není dostačující jelikož se stále objevuji nesmyslné obrázky u kterých nepozoruji žádnou nebo velmi minimální spojitost s tréninkovými daty vis. Obr.2.8. Pokud správně rozšíříme prompt, který používáme pro generování obrázku můžeme docílit více uspokojivých výsledků a omezit frekvenci s kterou se generují obrázky které nemají žádnou spojitost s našimi daty na kterých jsem model učil.

Výsledné modely jsem doučoval s pěti sty učícími kroky. Výstupy z těchto modelů již poskytovali uspokojivé výsledky pokud byl použit jako promp čistě název daného dinosaura nebo pokud byl tento promp správně rozšířen. V některých případech například když jse jako prompt zvolil "Sinomacrops bondei driving a car"tak jsem získal pouze obrázky aut případně obrázky aut a lidí.

(a) (b) (c)

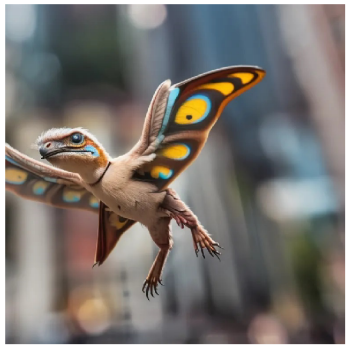
Obr. 2.7: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při 250 krocích učení. S použitým promptem: "Sinomacrops bondei"

(a) (b) (c)

Obr. 2.8: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při 250 krocích učení. S použitým promptem: "Sinomacrops bondei"

(a) "pencil drawn Sinomacrops bondei"

(b) "Sinomacrops bondei drinking watter from the river"



(c) "Sinomacrops bondei flying in the city"

Obr. 2.9: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při 500 krocích učení. S různými prompty

3 Porovnání s ostatními modely

Výstupy mého doučeného modelu jsem se rozhodl porovnat s výstupy modelu DALLE-3 který je dostupný v prohlížeči bing.

(a) (b) (c)

Obr. 3.1: Ukázka výstupu modelu po doučení na trénovacích datech: Sinomacrops bondei při 500 krocích učení. S promptem "Sinomacrops bondei flying in the forest"

(a) (b) (c)

Obr. 3.2: Ukázka výstupu modelu modelu DALLE-3 s promptem "Sinomacrops bondei flying in the forest"

Z obrázků 3.2 a 3.1 mužeme vypozorovat že DALLE-3 vytváří graficky kvalitnější obrázky ale neuvědomuje si jak daný dinosaurus má vypadat. A generuje místo něj motýly nebo jakési modé papoušky.

Zde vidíme že DALLE-3 již generuje obrázky letajících dinosaurů i když se neschodují s názvem daného dinosaura.

(a) (b) (c)

Obr. 3.3: Ukázka výstupu modelu po doučení na trénovacích datech: Dimorphodontidae při 500 krocích učení. S promptem "Dimorphodontidae flying in the forest"

(a) (b) (c)

Obr. 3.4: Ukázka výstupu modelu modelu DALLE-3 s promptem "Dimorphodontidae flying in the forest"

Závěr

Práce seznamuje se základy umělích neuronových sítí a dále pokračuje seznámením s pokročilejšími generativními modely. Popisuje funkce modelu CLIP, difůzních modelů, generování obrazů z textového vstupu pomocí modelu unCLIP. Dále se zmiňuje o pokročilejších modelech jako je DALL · E a Midjourney a poskytuje a porovnává výstupy těchto síti v situaci kdy mají oba modely zadaný stejný textový vstup. Jako formu vstupních kategorizovaných dat, pro učení mého text-to-image modelu, jsem si zvolil obrázky vymřelých dinosaurů. Na kterých svůj model budu následně učit. Model jsem dokázal naučit a výsledky jsem porovnal s jiným generativním modelem.

Literatura

[1] Jason Brownlee. Loss and loss functions for training deep learning neural networks, © 2024. URL: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[2] Jason Brownlee. How to choose loss functions when training deep learning neural networks, © 2024. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] Joseph Rocca. Understanding variational autoencoders (vaes). *Towards Data Science.*, 2019. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[5] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018. URL: https://lilianweng.github.io/posts/2018-08-12-vae/.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. arXiv:1706.03762.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. arXiv:2103.00020.

[8] How does clip text-to-image generation work?, 1.3.2022. URL: https://www.youtube.com/watch?v=-b7xKWeADHQ.

[9] Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[10] Ryan O'Connor. Introduction to diffusion models for machine learning, © 2024. URL: https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/.

[11] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. arXiv:2204.06125.

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, and Scott Gray. Dall · e: Creating images from text, © 2015–2024. URL: https://openai.com/index/dall-e/.

[13] Deep learning with pytorch: A 60 minute blitz, ©2024. URL: https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html.

[14] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[16] Ultralytics. google-images-download. https://github.com/ultralytics/google-images-download?tab=readme-ov-file, 2024.

[17] Hugging face, 2016. URL: https://huggingface.co/models.

Seznam symbolů a zkratek

GAN	generativní adverzní síť
VAE	generativní adverzní síť
GPT-2	Generative Pre-trained Transformer 2
CLIP	Contrastive Language–Image Pre-trainin