



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská



Moderní metody robustního strojového učení

Modern methods of robust machine learning

Diplomová práce

Autor: **Bc. Pavel Jakš**
Vedoucí práce: **Mgr. Lukáš Adam, Ph.D.**
Konzultant: **Ing. Pavel Strachota, Ph.D.**
Akademický rok: 2023/2024

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jakš** Jméno: **Pavel** Osobní číslo: **491912**
Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**
Zadávací katedra/ústav: **Katedra matematiky**
Studijní program: **Matematická informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Moderní metody robustního strojového učení

Název diplomové práce anglicky:

Modern methods of robust machine learning

Pokyny pro vypracování:

1. Nastudování literatury v oblasti metrik vizuální podobnosti a tvorby adversariálních vzorků.
2. Implementace vybraných metrik vizuální podobnosti včetně (aproximace) Wassersteinovy vzdálenosti a její porovnání s jinými implementacemi.
3. Využití naimplementovaných metod vizuální podobnosti pro tvorbu adversariálních vzorku na větším vzorku dat.
4. Porovnání naimplementovaných metrik mezi sebou a s veřejně dostupnými knihovnami na větším vzorku dat.
5. Zpřístupnění kódu na veřejném úložišti včetně dokumentace a vzorových příkladů použití knihovny.

Seznam doporučené literatury:

- [1] N. Akhtar, A. Mian, N. Kardan and M Shah, Advances in adversarial attacks and defenses in computer vision: A survey. IEEE Access 9, 2021, 155161-155196.
- [2] W. Eric, F. Schmidt, Z. Kolter, Wasserstein adversarial examples via projected sinkhorn iterations. International Conference on Machine Learning, PMLR, 2019.
- [3] J. Rauber, R. Zimmermann, M. Bethge, W. Brendel, Foolbox: A Python toolbox to benchmark the robustness of machine learning models. Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning, 2017.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Lukáš Adam, PhD. Ministerstvo životního prostředí

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

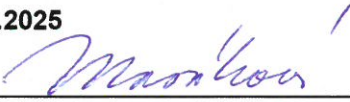
Ing. Pavel Strachota, Ph.D. katedra matematiky FJFI

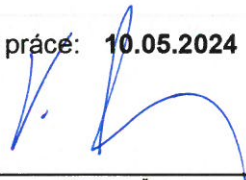
Datum zadání diplomové práce: **31.10.2023**

Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **31.10.2025**


Mgr. Lukáš Adam, PhD.
podpis vedoucí(ho) práce

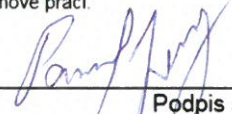

prof. Ing. Zuzana Masáková, Ph.D.
podpis vedoucí(ho) ústavu/katedry


doc. Ing. Václav Čuba, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.


Datum převzetí zadání


Podpis studenta

Poděkování:

Chtěl bych zde poděkovat především svému školiteli panu doktoru Adamovi za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé diplomové práce. Dále děkuji svému konzultantovi panu doktoru Strachotovi za jeho odborné rady.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 10. května 2024

Bc. Pavel Jakš

Název práce:

Moderní metody robustního strojového učení

Autor: Bc. Pavel Jakš

Studijní program: Matematická informatika

Druh práce: Diplomová práce

Vedoucí práce: Mgr. Lukáš Adam, Ph.D., Ministerstvo životního prostředí České republiky, Vršovická 1442/65, Vršovice, 100 10 Praha 10

Konzultant: Ing. Pavel Strachota, Ph.D., Katedra matematiky, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze, Trojanova 13, 120 00 Praha 2.

Abstrakt: V oblasti strojového učení se vyskytl problém existence tak zvaných adversariálních vzorků. Jedná se o jev, kdy i malá změna vstupu nějakého modelu strojového učení způsobí velikou změnu výstupu, což je ve většině případech nežádoucí. V této práci se potom věnujeme problematice metrik vizuální podobnosti, a to právě v kontextu existence a tvorby adversariálních vzorků v oblasti klasifikace obrázků. Naším cílem je osvětlit, jakým způsobem taková metrika vizuální podobnosti ovlivní proces tvorby adversariálních vzorků a jejich podobu.

Klíčová slova: Adversariální vzorky, CW útok, l_p norma, metrika vizuální podobnosti, neuronová síť, PSNR, robustní strojové učení, SSIM, Wassersteinova metrika.

Title:

Modern methods of robust machine learning

Author: Bc. Pavel Jakš

Abstract: There exists a problem in the field of machine learning called adversarial examples. This is a phenomenon where even a small change of the input to a machine learning model causes a big difference in the model output, which is unwanted in most cases. In this work we study the questions concerning visual similarity metrics and that in the context of existence and crafting of adversarial examples in the problem of image classification. Our goal is to enlighten the way how such a visual similarity metric affects the crafting process of adversarial examples and their final look.

Key words: Adversarial examples, CW attack, l_p norm, neural network, PSNR, robust machine learning, SSIM, visual similarity metric, Wasserstein metric.

Obsah

Úvod	8
1 Metriky vizuální podobnosti	9
1.1 Metriky indukované klasickými normami	10
1.2 MSE a RMSE	10
1.3 Peak signal-to-noise ratio	11
1.4 Wassersteinova vzdálenost	11
1.5 Structural similarity index measure	12
2 Detaily pro implementaci metrik vizuální podobnosti	14
2.1 Metriky založené na klasických normách	14
2.2 Modifikace Wassersteinovy vzdálenosti	14
2.2.1 Sinkhornova metrika	15
2.2.2 Duální Sinkhornova metrika	17
2.2.3 Řešení optimalizačního problému duální Sinkhornovy metriky	17
2.2.4 Výsledná implementace pro obrázky	19
2.3 Structural dissimilarity	19
3 Adversariální vzorky a jejich tvorba	20
3.1 Úvod do problematiky klasifikačních neuronových sítí	20
3.2 Adversariální vzorky	21
3.3 Tvorba adversariálních vzorků	22
3.4 Robustnost neuronové sítě	22
3.4.1 Přístup knihovny Foolbox	23
3.4.2 Přístup knihovny RobustBench	23
4 Implementace metrik vizuální podobnosti	25
4.1 Veřejné úložiště kódu	25
4.2 Koncepty pro použití kódu	25
4.3 Příklad použití metrik vizuální podobnosti	26
4.4 Testování naimplementovaných metrik	27
5 Porovnání metrik vizuální podobnosti	28
5.1 Časová náročnost výpočtu	28
5.2 Porovnání implementací s implementacemi veřejných knihoven	29
5.2.1 Aproximace Wassersteinovy vzdálenosti	29
5.2.2 DSSIM	29

6	Výsledky tvorby adversariálních vzorků	31
6.1	Detaily implementace adversariálního útoku	31
6.2	Srovnání útoků pro různé metiky	31
6.3	Adversariální útok za použití metriky indukované l_2 normou	33
6.4	Adversariální útok za použití metriky indukované l_1 normou	33
6.5	Adversariální útok za použití metriky vizuální podobnosti DSSIM	33
6.5.1	Velikost výpočetního podokna 5	33
6.5.2	Velikost výpočetního podokna 10	33
6.5.3	Velikost výpočetního podokna 20	33
6.5.4	Velikost výpočetního podokna jako velikost celého obrázku	39
6.6	Adversariální útok za použití aproximace Wassersteinovy vzdálenosti	39
	Závěr	44
	Literatura	45

Úvod

V této práci se věnujeme problematice metrik vizuální podobnosti, a to v kontextu strojového učení, konkrétně v oblasti existence a tvorby adversariálních vzorků. Zkoumáme, jaký vliv má volba takové metriky na tvorbu a podobu adversariálních vzorků.

Jelikož vizuální vjemy, tedy obrázky, reprezentujeme v počítači jako tenzory, lze na ně nahlédnout okem matematika, a to i na rozdíly mezi dvěma obrázky. Proto přejímáme klasické zavedení pojmu metrika, které vyjadřuje, jak jsou si dva objekty v jistém metrickém prostoru vzdáleny, či naopak jak jsou si blízko, případně jak jsou si podobny.

Od této matematické definice přecházíme k relaxovanému pojmu metriky vizuální podobnosti, který zde pro jeho širší formálně nedefinujeme. Podaří se nám ovšem zahrnout pod tento pojem i formálně nemetrické vzdálenosti jako je například index *DSSIM* (*structural dissimilarity*), který je šitý obrázkům na míru a při svém výpočtu nenahlíží na obrázky, jako na tenzory, jejichž složky jsou na sobě nezávislé, jako to například činí metriky založené na l_p normách.

Ohledně samotné problematiky existence adversariálních vzorků lze říci, že se jedná o jev, který je obecně typický pro metody strojového učení. My jsme se rozhodli jej zkoumat v kontextu *neuronových sítí*. Ve zkratce lze říci, že se jedná o fakt, že malá změna vstupu do modelu strojového učení zapříčiní velkou změnu výstupu tohoto modelu. Pro definici pojmu malá změna vstupu potom používáme právě ony metriky vizuální podobnosti, neboť se pohybujeme na poli počítačového vidění. Pro uvedení termínu velká změna výstupu na pravou míru využijeme povahu námi řešeného klasifikačního problému, tedy velká změna výstupu odpovídá změně v rozhodnutí klasifikátoru.

Proto vzniklo odvětví robustního strojového učení, neboť takové chování algoritmů strojového učení je ze zřejmých důvodů nežádoucí. Toto odvětví se pak snaží existenci adversariálních vzorků zabránit, a to různými úpravami samotných algoritmů strojového učení, resp. v modifikaci dat, na kterých se daný model učí.

Na základě těchto vědomostí se potom ptáme, zda má vliv volba metriky vizuální podobnosti na tvorbu adversariálních vzorků.

Stěžejní součástí této práce potom je i samotná implementace uvedených metrik vizuální podobnosti, tak aby mohly být tyto metriky použity pro tvorbu adversariálních vzorků. Tyto metriky jsou potom zpřístupněny na veřejném úložišti včetně dokumentace.

Kapitola 1

Metriky vizuální podobnosti

Metrika vizuální podobnosti je nástroj, který umožňuje, jak napovídá sám název, měřit, jak jsou si dva vizuální vjemy podobné. Pod vizuálním jevem zde v kontextu strojového učení myslíme strojově zpracovatelný vizuální vjem, tedy obrázek. Tedy obecně se jedná o tenzor z množiny $\mathbb{M}^{C \times W \times H}$, kde \mathbb{M} je podmnožina reálných čísel, za kterou volíme například množinu $\{0, 1, \dots, 255\}$, tedy diskrétní hodnoty pixelů, které lze reprezentovat pomocí 8 bitů, nebo třeba za \mathbb{M} volíme interval $[0, 1]$. Parametry C, W, H potom reprezentují po řadě počet kanálů obrázku, šířku obrázku a výšku obrázku. V praxi se nejčastěji setkáme se šedotónovými obrázky, potom $C = 1$, nebo s obrázky typu *RGB*, kde $C = 3$ a jednotlivé kanály reprezentují po řadě červenou, zelenou a modrou barvu. Alternativní přístup k obrázku je potom reprezentace pomocí tenzorů $\mathbb{M}^{W \times H \times C}$, kdy měníme pořadí indexace jednotlivých prvků tenzoru. V této práci potom užíváme první z konvencí. Důvodem je, že tato konvence je vlastní knihovně *PyTorch*, která je zde užita. Máme tedy ujasněno slovo *vizuální* z termínu *metriky vizuální podobnosti*.

Pod pojmem *podobnost* si potom představme to, jaké společné rysy dva takové obrázky mají. Může se jednat o vyobrazení stejného objektu či o informaci, kterou nesou. Nebo prostě blízkost ve smyslu pohledu na obrázky jako na dva tenzory.

Nakonec rozved' me pojem metrika. Slovo metrika v první řadě vyjadřuje propojení vzdálenosti nebo podobnosti ve výše uvedeném smyslu s jedním konkrétním reálným číslem. Metrika je tedy zobrazení, které na vstupu bere dva prvky stejné množiny a vrací číslo, které vyjadřuje, jak moc si jsou tyto dva objekty blízko či jak jsou si tyto dva objekty podobné. Metriku lze ovšem formálně matematicky definovat, aby dávala v konečném důsledku vzniknout topologii, což je nástroj, kterým vybavíme-li libovolnou množinu, můžeme nakládat s pojmy jako je okolí bodu, otevřená množina či kompaktnost. Pod pojmem metrika na prostoru X si tedy každý matematik představí zobrazení $\rho : X \times X \rightarrow [0, +\infty)$ splňující

1. $\rho(x, y) = 0 \iff x = y \quad \forall x, y \in X$,
2. $\rho(x, y) = \rho(y, x) \quad \forall x, y \in X$,
3. $\rho(x, z) \leq \rho(x, y) + \rho(y, z) \quad \forall x, y, z \in X$.

Taková metrika může být na lineárním prostoru V nad číselným tělesem (pro naše účely zůstaňme nad \mathbb{R}) snadno zadána pomocí normy, která je buď indukována skalárním součinem v případě pre-Hilbertových prostorů, nebo dána vlastnostmi, že se jedná o zobrazení $\|\cdot\| : V \rightarrow [0, +\infty)$ a splňuje:

1. $\|x\| = 0 \iff x = 0 \quad \forall x \in V$,
2. $\|\alpha x\| = |\alpha| \cdot \|x\| \quad \forall \alpha \in \mathbb{R}, \forall x \in V$,
3. $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in V$.

Metriku potom získáme z normy následující konstrukcí:

$$\rho(x, y) = \|x - y\|,$$

tedy vzdálenost dvou vektorů je dána normou rozdílu vektorů. Snadno lze nahlédnout, že takto zadané zobrazení je metrika. S metrikami, které jsou tzv. indukované normami dle předchozího se setkáme.

1.1 Metriky indukované klasickými normami

Vzhledem k tomu, že obrázky, které jsou středem naší pozornosti, lze reprezentovat jako tenzory o rozměrech $C \times W \times H$, kde C , W a H jsou jako výše, tak lze tyto tenzory použít jako vstup pro L^p normy. Pro $p \in [1, +\infty)$ je L^p norma z $f \in L_p(X, \mu)$ definována vztahem:

$$\|f\|_p = \left(\int_X |f|^p d\mu \right)^{\frac{1}{p}}.$$

Pro naše obrázky lze za X vzít $\{1, \dots, C\} \times \{1, \dots, W\} \times \{1, \dots, H\}$ a za μ počítací míru. Potom naše L^p norma přejde v l_p normu, která má pro naše obrázky, tedy tenzory $x \in \mathbb{R}^{C \times W \times H}$, tvar:

$$\|x\|_p = \left(\sum_{i=1}^C \sum_{j=1}^W \sum_{k=1}^H |x_{i,j,k}|^p \right)^{\frac{1}{p}}. \quad (1.1)$$

Této definici se potom vymyká l_∞ norma, která má tvar pro tenzor $x \in \mathbb{R}^{C \times W \times H}$:

$$\|x\|_\infty = \max_{i \in \{1, \dots, C\}} \max_{j \in \{1, \dots, W\}} \max_{k \in \{1, \dots, H\}} |x_{i,j,k}|. \quad (1.2)$$

1.2 MSE a RMSE

Vzdálenosti, které mají blízko k metrikám indukovaným l_2 normou, jsou *MSE* (z anglického *Mean Squared Error*) a *RMSE* (z anglického *Root Mean Squared Error*). Pro tenzory $x, \tilde{x} \in \mathbb{R}^{C \times W \times H}$ mají definici:

$$\text{MSE}(x, \tilde{x}) = \frac{1}{CWH} \sum_{i=1}^C \sum_{j=1}^W \sum_{k=1}^H |x_{i,j,k} - \tilde{x}_{i,j,k}|^2 \quad (1.3)$$

$$\text{RMSE}(x, \tilde{x}) = \left(\frac{1}{CWH} \sum_{i=1}^C \sum_{j=1}^W \sum_{k=1}^H |x_{i,j,k} - \tilde{x}_{i,j,k}|^2 \right)^{\frac{1}{2}} \quad (1.4)$$

Jedná se vlastně o transformaci metriky založené na l_2 normě. Platí totiž:

$$\text{MSE}(x, \tilde{x}) = \frac{1}{CWH} \|x - \tilde{x}\|_2^2 \quad (1.5)$$

$$\text{RMSE}(x, \tilde{x}) = \frac{1}{\sqrt{CWH}} \|x - \tilde{x}\|_2 \quad (1.6)$$

Tyto metriky vizuální podobnosti potom sehrávají roli, chceme-li porovnávat rozdíly mezi obrázky napříč obrázky různých rozměrů. Samozřejmě to ale neznamená, že získáme vzdálenost dvou obrázků, které mají každý jiný rozměr.

1.3 Peak signal-to-noise ratio

Vzdálenost označená zkratkou *PSNR* z anglického *peak signal-to-noise ratio* vyjadřuje vztah mezi obrázkem $x \in \mathbb{R}^{C \times W \times H}$ a jeho pokažením $\tilde{x} \in \mathbb{R}^{C \times W \times H}$, což je obrázek, který má zásadě nést stejnou informaci, ale je poškozený, a to ať už rozmazáním nebo šumem. Cílem této metriky vizuální podobnosti je potom kvantitativně vyjádřit právě míru šumu. Definice je následující:

$$\text{PSNR}(x, \tilde{x}) = 10 \cdot \log_{10} \left(\frac{l^2}{\text{MSE}(x, \tilde{x})} \right), \quad (1.7)$$

$$= 20 \cdot \log_{10} \left(\frac{l}{\text{RMSE}(x, \tilde{x})} \right), \quad (1.8)$$

kde l je dynamický rozsah obrázků, tedy rozdíl mezi maximální možnou hodnotou pixelů a minimální možnou hodnotou pixelů. Jedná se tedy o transformaci metriky *MSE*. Samotná hodnota *PSNR* ovšem není metrická vzdálenost. Vždyť budou-li se obrázky x a \tilde{x} blížit k sobě, hodnota $\text{PSNR}(x, \tilde{x})$ poroste do nekonečna, neboť jmenovatel argumentu v logaritmu v definici (1.7) jde k nule, a to zprava, tudíž argument logaritmu jde do $+\infty$ a tedy i logaritmus roste do $+\infty$. Toto odpovídá tomu, že šum v pokažení \tilde{x} je nulový.

1.4 Wassersteinova vzdálenost

Bud' (M, ρ) metrický prostor. Zvolme $p \in [1, +\infty)$. Potom máme *Wassersteinovu p -vzdálenost* mezi dvěma pravděpodobnostními mírami μ a ν na M , které mají konečné p -té momenty, jako:

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y) \sim \gamma} \rho(x, y)^p \right)^{\frac{1}{p}}, \quad (1.9)$$

kde $\Gamma(\mu, \nu)$ je množina všech sdružených pravděpodobnostních měr na $M \times M$, které mají po řadě μ a ν za marginální pravděpodobnostní míry [6].

Jak to souvisí s obrázky? Přes dopravní problém. Pod pravděpodobnostní distribucí μ či ν na X si lze představit rozložení jakési hmoty o celkové hmotnosti 1. Sdružená rozdělení $\gamma \in \Gamma(\mu, \nu)$ potom odpovídají transportnímu plánu, kde $\gamma(x, y) dx dy$ vyjadřuje, kolik hmoty se přesune z x do y . Tomu lze přiřadit nějakou cenu c , totiž kolik stojí přesun jednotkové hmoty z x do y : $c(x, y)$. V případě *Wassersteinovy vzdálenosti* za cenu dosadíme $c(x, y) = \rho(x, y)^p$, tedy p -tou mocninu vzdálenosti mezi x a y . Potom cena celkového dopravního problému s transportním plánem γ bude:

$$c_\gamma = \int c(x, y) \gamma(x, y) dx dy \quad (1.10)$$

a optimální cena bude:

$$c = \inf_{\gamma \in \Gamma(\mu, \nu)} c_\gamma. \quad (1.11)$$

Po dosazení:

$$c = \inf_{\gamma \in \Gamma(\mu, \nu)} \int c(x, y) \gamma(x, y) dx dy \quad (1.12)$$

$$= \inf_{\gamma \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y) \sim \gamma} c(x, y) \quad (1.13)$$

$$= \inf_{\gamma \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y) \sim \gamma} \rho(x, y)^p \quad (1.14)$$

$$= W_p(\mu, \nu)^p \quad (1.15)$$

Dostáváme tedy interpretaci, že p -tá mocnina *Wassersteinovy vzdálenosti* odpovídá ceně dopravního problému.

Pro obrázky má tato konstrukce následující uplatnění: Obrázky je třeba chápat jako diskrétní pravděpodobnostní rozdělení, proto je třeba je normalizovat, aby součet prvků tenzoru obrázku byl roven 1. Pak střední hodnota v definici *Wassersteinovy vzdálenosti* přejde ve váženou sumu cen, tedy p -tých mocnin vzdáleností mezi jednotlivými pixely.

Jak je to barevnými obrázky, tedy s obrázku, které mají více než jeden kanál? Zde lze uplatnit následující dva přístupy:

1. Normovat celý obrázek na jedničku, tedy všechny kanály dohromady, a tím pádem i definovat vzdálenost mezi jednotlivými kanály,
2. Normovat každý kanál zvlášť na jedničku, počítat *Wassersteinovu metriku* pro každý kanál zvlášť a následně vybrat nějakou statistiku výsledných vzdáleností, např. průměr.

1.5 Structural similarity index measure

Zkratka *SSIM* pochází z anglického *structural similarity index measure*. Tato metrika se při výpočtu indexu dvou obrázků x a \tilde{x} dívá na podokna, ze kterých vybere jisté statistiky a z nich vytvoří index pro daná podokna obrázků. Potom se jako celkový index bere průměr přes tato okna. Uved' me vzorce pro výpočet indexu *SSIM* pro případ, že máme jediné okno, které splývá s obrázkem, které pro jednoduchost zvolme jednobarevné, tedy černobílé. Označme $N = W \times H$ počet pixelů v obrázku a indexujme prvky matice obrázku jediným číslem. Potom definujeme pro obrázky x a \tilde{x} následující:

$$\begin{aligned}\mu_x &= \frac{1}{N} \sum_{i=1}^N x_i, \\ \mu_{\tilde{x}} &= \frac{1}{N} \sum_{i=1}^N \tilde{x}_i, \\ \sigma_x^2 &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2, \\ \sigma_{\tilde{x}}^2 &= \frac{1}{N-1} \sum_{i=1}^N (\tilde{x}_i - \mu_{\tilde{x}})^2, \\ \sigma_{x\tilde{x}} &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(\tilde{x}_i - \mu_{\tilde{x}}).\end{aligned}$$

Tedy proměnné μ_x a $\mu_{\tilde{x}}$ odpovídají průměru, σ_x^2 a $\sigma_{\tilde{x}}^2$ rozptylu a $\sigma_{x\tilde{x}}$ kovarianci. Potom definujeme:

$$\text{SSIM}(x, \tilde{x}) = \frac{(2\mu_x\mu_{\tilde{x}} + C_1)(2\sigma_{x\tilde{x}} + C_2)}{(\mu_x^2 + \mu_{\tilde{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\tilde{x}}^2 + C_2)}, \quad (1.16)$$

kde C_1, C_2 jsou konstanty pro stabilitu dělení volené kvadraticky úměrně dynamickému rozsahu. Tato výsledná statistika je potom součinem tří metrik, které jsou definovány jako metrika jasů

$$l(x, \tilde{x}) = \frac{2\mu_x\mu_{\tilde{x}} + C^{(1)}}{\mu_x^2 + \mu_{\tilde{x}}^2 + C^{(1)}}, \quad (1.17)$$

metrika kontrastu

$$c(x, \tilde{x}) = \frac{2\sigma_x\sigma_{\tilde{x}} + C^{(2)}}{\sigma_x^2 + \sigma_{\tilde{x}}^2 + C^{(2)}}, \quad (1.18)$$

a metrika struktury

$$s(x, \tilde{x}) = \frac{\sigma_{x\tilde{x}} + C^{(3)}}{\sigma_x\sigma_{\tilde{x}} + C^{(3)}}. \quad (1.19)$$

Volba konstant je potom

$$C^{(1)} = C_1, \quad (1.20)$$

$$C^{(2)} = C_2, \quad (1.21)$$

$$C^{(3)} = \frac{C_2}{2}. \quad (1.22)$$

SSIM je pak metrikou vizuální podobnosti kombinující informaci o podobnosti jasu, kontrastu a struktury. Můžeme si povšimnout, že $\text{SSIM}(x, \tilde{x})$ není metrická vzdálenost. Budou-li obrázky stejné, nevychází 0, nýbrž 1 [7]. Může se také stát, že SSIM vrátí zápornou hodnotu, která může vzniknout členem $\sigma_{x\tilde{x}}$. Jak volíme celkový SSIM pro barevné obrázky? Jako průměr přes kanály.

Kapitola 2

Detaily pro implementaci metrik vizuální podobnosti

V minulé kapitole jsme viděli přehled metod, jak přistoupit k porovnávání dvou různých obrázků. Předvedli jsme, jak vyčíslit rozdíl mezi dvěma obrázky. Ne vždy se ovšem jedná o metriku ve smyslu matematickém, což pro tvorbu adversariálních vzorků je záhodno, a ne vždy lze takovou vzdálenost přímočaře spočítat. Proto uved' me, je-li to nutné, příslušné úkroky stranou, které nám umožní hledat adversariální vzorky, a to pokud možno v krátkém čase.

Poznatky z této kapitoly nám potom pomáhají k vlastní implementaci metrik vizuální podobnosti v programovacím jazyce *Python* [20] za použití knihovny *PyTorch* [21].

2.1 Metriky založené na klasických normách

Implementovat klasické l_p normy je snadné, a tedy i metriky jimi indukované. MSE a RMSE jsou též snadné na implementaci. Vlastně i PSNR. Metriku vizuální podobnosti PSNR je třeba ovšem ošetřit, neboť, jak již bylo poznamenáno, budou-li se obrázky x a \tilde{x} blížit k sobě, hodnota $\text{PSNR}(x, \tilde{x})$ poroste do nekonečna. Proto zkusme vzít konstrukci, kde prohodíme roli dynamického rozsahu l (peak signal) s rolí šumu (noise), dostaneme tedy, co lze nazvat noise to peak signal ratio (NPSR):

$$\text{NPSR}(x, \tilde{x}) = 20 \cdot \log_{10} \left(\frac{\text{RMSE}(x, \tilde{x})}{l} \right), \quad (2.1)$$

$$= -\text{PSNR}(x, \tilde{x}). \quad (2.2)$$

Při dvou obrázcích blížících se k sobě bude tedy NPSR klesat, a to neomezeně.

2.2 Modifikace Wassersteinovy vzdálenosti

Abychom mohli s Wassersteinovou metrikou nakládat například v počítači, je nutné tuto metriku spočítat. Podíváme-li se do definice (1.9), znamená to vyřešit optimalizační problém. Byť bychom se omezili hledání vzdáleností dvou vektorů o rozměru q , měli bychom problém s časovou složitostí nejlépe $O(q^3 \log q)$ [8]. Bohužel takovou výpočetní kapacitu, která by toto zvládla rychle pro námi používané obrázky, nemáme k dispozici. Proto se podívejme, jak Wassersteinovu vzdálenost spočítat rychleji, byť za cenu ztráty přesnosti.

V následujících odstavcích textu uvažujme, že chceme spočítat vzdálenost mezi diskrétními rozděleními, jejichž pravděpodobnostními vektory jsou $\mu \in \mathbb{R}^q$ a $\nu \in \mathbb{R}^q$, kde $q \in \mathbb{N}$. Dále označme 1_q vektor, jehož hodnoty v každé souřadnici jsou rovny 1 a má dimenzi udanou v indexu. Platí tedy $\mu^T 1_q = \nu^T 1_q = 1$ a $\forall i \mu_i \geq 0$ a $\forall j \nu_j \geq 0$. Označme jako $U(\mu, \nu)$ množinu všech matic $P \in \mathbb{R}^{q \times q}$, $P_{i,j} \geq 0$ takových, že $P 1_q = \mu$ a $P^T 1_q = \nu$. Jako matici C označme zadanou matici cen, která splňuje, že reprezentuje metriku. To znamená, že $C_{i,j} \geq 0$, $C_{i,j} = 0 \iff i = j$, $C_{i,j} = C_{j,i}$ a $C_{i,k} \leq C_{i,j} + C_{j,k}$. Tedy prvek matice $C_{i,j}$ určuje metrickou vzdálenost mezi prvkem i a prvkem j . Potom lze napsat:

$$W(\mu, \nu) \equiv W_1(\mu, \nu) = \min_{P \in U(\mu, \nu)} \langle P, C \rangle, \quad (2.3)$$

kde $\langle P, C \rangle = \sum_{i,j=1}^q P_{i,j} C_{i,j}$. Toto je důsledkem volby $p = 1$. To že infimum v definici (1.9) přechází v minimum je dáno tím, že $U(\mu, \nu)$ je kompaktní podmnožina $\mathbb{R}^{q \times q}$. Dále poznamenejme, že $U(\mu, \nu)$ je také konvexní množina, což vychází z linearitě jejích hraničních podmínek a faktu, že dvě nezáporná čísla se vždy sečtou opět na nezáporné číslo.

2.2.1 Sinkhornova metrika

Začneme s mírnou úpravou původního optimalizačního problému definujícího Wassersteinovu vzdálenost: Pro $\alpha > 0$ definujme jakési α okolí rozdělení $\mu\nu^T$ (sdružené pravděpodobnostní rozdělení s marginálními μ a ν , ve kterém μ a ν jsou nezávislá rozdělení) ve smyslu *Kullback-Leiblerovy divergence*

$$U_\alpha(\mu, \nu) = \{P \in U(\mu, \nu) | KL(P || \mu\nu^T) \leq \alpha\}. \quad (2.4)$$

Připomeňme definici Kullback-Leiblerovy divergence:

$$KL(\tilde{P} || \hat{P}) = \sum_{i=1}^q \sum_{j=1}^q \tilde{P}_{i,j} \log \frac{\tilde{P}_{i,j}}{\hat{P}_{i,j}}.$$

Pro dané $P \in U(\mu, \nu)$ lze na kvantitu $KL(P || \mu\nu^T)$ nahlédnout jako na informaci mezi veličinami s rozděleními μ a ν . Tedy $U_\alpha(\mu, \nu)$ vybírá ta rozdělení, která nesou malou vzájemnou informaci mezi μ a ν (ve smyslu menší než α).

Potom lze definovat Sinkhornovu metriku pomocí zúžení definičního oboru jako

$$W^\alpha(\mu, \nu) = \min_{P \in U_\alpha(\mu, \nu)} \langle P, C \rangle. \quad (2.5)$$

Jelikož

$$(\forall P \in U_\alpha(\mu, \nu)) (P \in U(\mu, \nu)),$$

tak jistě

$$W^\alpha(\mu, \nu) \geq W(\mu, \nu).$$

Sinkhornova metrika je tedy horním odhadem Wassersteinovy metriky.

Zde se tedy při hledání optimálního řešení $P^* = \operatorname{argmin}_{P \in U_\alpha(\mu, \nu)} \langle P, C \rangle$ omezíme na ta řešení, která nesou dostatečně malou vzájemnou informaci mezi rozděleními μ a ν . Na výraz $KL(P || \mu\nu^T)$ lze ale

nahlédnout též následovně:

$$KL(P||\mu\nu^T) = \sum_{i=1}^q \sum_{j=1}^q P_{i,j} \log \frac{P_{i,j}}{\mu_i \nu_j}, \quad (2.6)$$

$$= \sum_{i=1}^q \sum_{j=1}^q P_{i,j} \log P_{i,j} - \sum_{i=1}^q \sum_{j=1}^q P_{i,j} \log \mu_i - \sum_{i=1}^q \sum_{j=1}^q P_{i,j} \log \nu_j, \quad (2.7)$$

$$= \sum_{i=1}^q \sum_{j=1}^q P_{i,j} \log P_{i,j} - \sum_{i=1}^q \mu_i \log \mu_i - \sum_{j=1}^q \nu_j \log \nu_j, \quad (2.8)$$

$$= -H(P) + H(\mu) + H(\nu), \quad (2.9)$$

kde H značí entropii. Odtud můžeme vidět, že podmínka $KL(P||\mu\nu^T) \leq \alpha$ znamená i to, že požadujeme po sdruženém rozdělení P , aby mělo velkou entropii. Tedy:

$$H(P) \geq H(\mu) + H(\nu) - \alpha. \quad (2.10)$$

Abychom osvětlili toto omezení definičního oboru hledání P^* , stačí si uvědomit, že na základě principu maximální entropie [22] je jednodušší vybírat ta rozdělení, která mají entropii co největší. Avšak dodejme, že tato myšlenka je v protikladu s klasickým výsledkem lineárního programování, jehož je v našem případě původní Wassersteinův problém instancí. Lineární programování totiž ukazuje, že řešení dopravního problému je takové rozdělení P , které je velmi podobné deterministickému, tedy, že entropie tohoto rozdělení je velmi nízká.

Zkusme nyní nahlédnout na optimalizační problém v (2.5):

$$W^\alpha(\mu, \nu) = \min_{P \in U(\mu, \nu)} \langle P, C \rangle, \quad (2.11)$$

$$KL(P||\mu\nu^T) - \alpha \leq 0. \quad (2.12)$$

Toto je instance konvexního optimalizačního problému, neboť účelová funkce je lineární, tedy i konvexní, dále Kullback-Leiblerova divergence je vzhledem ke svému prvnímu argumentu opět konvexní, což plyne z konvexity funkce $x \ln x$. To lze snadno ověřit výpočtem druhé derivace této funkce. A nakonec sám definiční obor je konvexní množina, jak již bylo poznamenáno. Dále uvažme, že $\mu\nu^T \in U(\mu, \nu)$ a zároveň $KL(\mu\nu^T||\mu\nu^T) = 0$. Z tohoto vyplývá podle [23], že pro tento problém platí silná dualita.

Lagrangeova funkce tohoto problému vypadá následovně:

$$\mathcal{L}(P, \lambda) = \langle P, C \rangle + \lambda (KL(P||\mu\nu^T) - \alpha). \quad (2.13)$$

Na základě teorie optimalizace potom hledáme:

$$\min_{P \in U(\mu, \nu)} \max_{\lambda \geq 0} \mathcal{L}(P, \lambda), \quad (2.14)$$

což díky silné dualitě znamená, že hledáme:

$$\max_{\lambda \geq 0} \min_{P \in U(\mu, \nu)} \mathcal{L}(P, \lambda), \quad (2.15)$$

Tedy pro $\alpha > 0$ existuje $\lambda > 0$ takové, že $W^\alpha(\mu, \nu) = \min_{P \in U(\mu, \nu)} \mathcal{L}(P, \lambda)$. Což je ekvivalentní s tím, že existuje $\xi > 0$ takové, že $W^\alpha(\mu, \nu) = \min_{P \in U(\mu, \nu)} \mathcal{L}(P, \frac{1}{\xi})$. Když ještě rozepíšeme Lagrangeovu funkci při uvážení (2.9):

$$W^\alpha(\mu, \nu) = \min_{P \in U(\mu, \nu)} \left(\langle P, C \rangle - \frac{1}{\xi} H(P) \right) + \frac{1}{\xi} (H(\mu) + H(\nu) - \alpha). \quad (2.16)$$

2.2.2 Duální Sinkhornova metrika

Přejděme nyní od Sinkhornovy metriky k tzv. duální Sinkhornově metrice. Ta je pro pevně zvolené $\xi > 0$ definována následovně:

$$W^\xi(\mu, \nu) = \langle P^\xi, C \rangle, \quad (2.17)$$

$$\text{kde } P^\xi = \operatorname{argmin}_{P \in U(\mu, \nu)} \left(\langle P, C \rangle - \frac{1}{\xi} H(P) \right), \quad (2.18)$$

kde $H(P)$ je opět entropie pravděpodobnostního rozdělení P . Jedná se tedy ve výsledku o regularizovaný dopravní problém, který je inspirován úvahami uvedenými v sekci o Sinkhornově metrice.

Vliv parametru ξ je potom následující: Máme-li ξ dostatečně velké, je jeho převrácená hodnota blízká nule, a proto se řešení regularizovaného problému bude blížit Wassersteinově metrice. Tato úprava Wassersteinovy metriky je, jak se přesvědčíme, mnohem lépe vyčíslitelná. Nejdříve se ovšem podívejme na intuici za touto úpravou.

Přechod od Wassersteinovy metriky k Sinkhornově spočíval v restrikci definičního oboru, kde hledáme (v řeči dopravního problému) dopravní plán P , na takové dopravní plány, které mají velkou entropii. Tohoto lze ovšem vlastně docílit i jiným způsobem, a to použitím regularizace v samotné účelové funkci optimalizačního problému. Proto duální Sinkhornova metrika, u které máme to štěstí, že ji lze spočítat relativně rychle.

Optimalizační problém duální Sinkhornovy metriky je opět konvexní, což vychází z konkávnosti entropické funkce, která je zde vzata s opačným znaménkem a z již komentované konvexity definičního oboru.

2.2.3 Řešení optimalizačního problému duální Sinkhornovy metriky

Jelikož Lagrangeova funkce daného optimalizačního problému vypadá následovně:

$$L(P; m, n) = \sum_{i,j} P_{i,j} C_{i,j} - \frac{1}{\xi} H(P) + \sum_i m_i \left(\sum_j P_{i,j} - \mu_i \right) + \sum_j n_j \left(\sum_i P_{i,j} - \nu_j \right),$$

kde m a n jsou vektory Lagrangeových multiplikátorů, tak derivace této Lagrangeovy funkce dle $P_{i,j}$ bude vypadat následovně:

$$\frac{\partial L}{\partial P_{i,j}} = C_{i,j} + \frac{1}{\xi} \log P_{i,j} + \frac{1}{\xi} + m_i + n_j.$$

Položme nyní tento gradient roven nule a vyjádřeme $P_{i,j}$:

$$\begin{aligned} P_{i,j} &= \exp(-\xi C_{i,j} - 1 - \xi m_i - \xi n_j) \\ &= \exp\left(-\xi m_i - \frac{1}{2}\right) \exp(-\xi C_{i,j}) \exp\left(-\xi n_j - \frac{1}{2}\right) \end{aligned}$$

Označme $u \in \mathbb{R}^q$, pro který platí $u_i = \exp\left(-\xi m_i - \frac{1}{2}\right)$ a $v \in \mathbb{R}^q$, pro který platí $v_j = \exp\left(-\xi n_j - \frac{1}{2}\right)$, dále jako $K \in \mathbb{R}^{q \times q}$ matici, pro kterou platí $K_{i,j} = \exp(-\xi C_{i,j})$. Dostáváme tedy, že řešení optimalizačního problému nalezneme ve tvaru součinu tří matic:

$$P = \operatorname{diag}(u) K \operatorname{diag}(v), \quad (2.19)$$

kde diag přiřadí vektoru čtvercovou matici, která má vektor v v argumentu na diagonále. S přihlédnutím k faktu, že matice K je vzhledem k proměnným Lagrangeovy funkce nezávislá a konstantní, vidíme že

hledáme vektory u a v . Tyto vektory jsou potom transformacemi vektorů Lagrangeových multiplikátorů m a n .

Podle Sinkhornovy věty [19] potom platí, jelikož matice K je čtvercová a má kladné prvky, že existují matice $D_1, D_2 \in \mathbb{R}^{q \times q}$, které jsou diagonální s kladnými prvky na diagonále, pro které platí, že matice $Q = D_1 K D_2$ je dvojité stochastická, tedy že $\sum_i Q_{i,j} = \sum_j Q_{i,j} = 1$. Tyto matice jsou jednoznačně určené až na multiplikativní konstantu, kterou lze jednu z matic vynásobit a druhou vydělit.

Získáme-li tedy matici Q , můžeme její řádky vynásobit prvky vektoru rozdělení μ a její sloupce prvky vektoru rozdělení ν . Tím získáme matici $z U(\mu, \nu)$, tedy z našeho definičního oboru, napsanou ve tvaru:

$$\begin{aligned} R &= \text{diag}(\mu) Q \text{diag}(\nu), \\ &= \text{diag}(\mu) D_1 K D_2 \text{diag}(\nu) \end{aligned}$$

Když tuto matici R položíme za hledanou matici P splníme kritérium (2.19), jelikož matice $\text{diag}(\mu) D_1$ a matice $\text{diag}(\nu) D_2$ jsou diagonální. Je ovšem otázkou, jestli je toto kritérium nejen nutné, nýbrž i postačující. Ano, je i postačující, a to z důvodu konvexity optimalizační úlohy. Tedy, najdeme-li Q nalezneme i hledané P .

K hledání této matice potom slouží algoritmus Sinkhornových iterací pevného bodu. Tento algoritmus je založen na postupném škálování řádků, resp. sloupečků původní matice (v našem případě matice K) tak, aby se řádky, resp. sloupce sečetly na 1. Detaily k tomuto algoritmu a rozbor jeho konvergence lze pak nalézt v [24]. Ale jelikož původní problém nespočívá v nalezení matice, jež je dvojité stochastická, nýbrž jež se nachází v $U(\mu, \nu)$, tak lze tyto iterace ještě upravit, aby výsledkem byla hledaná matice P , potažmo její Frobeniův součin s maticí cen, jež je hodnotou duální Sinkhornovy metriky.

V [8] potom lze nalézt onen algoritmus pro výpočet duální Sinkhornovy metriky: Na vstupu algoritmus dostává pravděpodobnostní rozdělení μ a ν , jejichž vzdálenost je hledaná, dále matici cen C a regularizační parametr ξ .

1. $\tilde{q} = \sum_{i=1}^q 1_{\mu_i \neq 0}$
2. $\tilde{\mu} \in \mathbb{R}^{\tilde{q}}, \tilde{\mu}_i = \mu_{j_i}$, tj. do proměnné $\tilde{\mu}$ uložíme právě nenulové prvky μ .
3. $\tilde{C} \in \mathbb{R}^{\tilde{q} \times q}, \tilde{C}_{i,j} = C_{i_{k,j}}$, tj. do proměnné \tilde{C} uložíme příslušné řádky matice cen.
4. $K = \exp(-\xi \tilde{C})$ - jako matici K vezmeme matici, která vznikne po prvcích jako exponenciála matice $-\xi \tilde{C}$.
5. $u = 1_{\tilde{q}} / \tilde{q}$, tj. do proměnné u uložíme rovnoměrné rozdělení délky \tilde{q} .
6. $\hat{K} = (1. / \tilde{\mu}) * K$
7. Opakujme: $u = 1. / (\hat{K}(\nu. / (K^T u)))$ - dokud není dosaženo vhodné zastavovací kritérium.
8. $v = \nu. / (K^T u)$.
9. $W^\xi(\mu, \nu) = u((K * \tilde{C})v)$.

V algoritmu výše potom $./$, resp. $*$ znamená dělení, resp. násobení ve smyslu Hadamardově.

2.2.4 Výsledná implementace pro obrázky

V minulé kapitole jsme nastínili, jak Wassersteinovu metriku chápat jako vzdálenost mezi obrázky. Uvažme pro jednoduchost dva jednokanálové obrázky společné šířky W a společné výšky H . Potom na daný obrázek lze nahlédnout jako na histogram nějakého diskrétního pravděpodobnostního rozdělení. Podmínkou tedy je, že obrázek přeškálujeme, aby součet hodnot všech pixelů byl roven jedné. Dále je třeba z takového obrázku, tedy vlastně matice vytvořit vektor. Udělejme to tak, že za sebe po řadě naskládáme řádky této matice do jednoho dlouhého řádku. Matematicky potom pro formální přesnost provedeme transpozici, abychom získali sloupcový vektor. Získáváme tedy $\mu, \nu \in \mathbb{R}^{W \cdot H}$.

Další krok je na základě rozměrů původních obrázků určit matici cen, kterou budeme používat pro výpočet aproximace Wassersteinovy metriky. Definujme proto matici cen C jako $C \in \mathbb{R}^{W \cdot H \times W \cdot H}$:

$$C_{i,j} = \left\| \begin{bmatrix} i \operatorname{div} W \\ i \bmod W \end{bmatrix} - \begin{bmatrix} j \operatorname{div} W \\ j \bmod W \end{bmatrix} \right\|, \quad (2.20)$$

kde $x \operatorname{div} y$ značí dolní celou část podílu $\frac{x}{y}$. Jedná se tedy o matici, která obsahuje metrickou vzdálenost vždy mezi dvěma body v rovině. Za tuto metrickou vzdálenost volme metriku indukovanou l_2 či l_1 normou. V (2.20) potom počítáme s indexováním od nuly.

Potom lze pro daný regularizační parametr ξ provádět algoritmus popsany v předešlé podsekcí. Bohužel ale v praxi narážíme na limity strojové přesnosti. Při provádění kroku 4 v algoritmu se stává, že ona vypočtená exponenciála je v počítači reprezentována nulou. Dokonce není ojedinělé, že je v počítači nulový celý sloupec vzniklé matice K . Toto způsobuje, že algoritmus selže, neboť se v kroku 7 potom nulou dělí. Proto do výsledné implementace vstupuje ještě další parametr, a to je malá kladná konstanta pro stabilitu dělení, kterou nahradíme každý prvek matice K , který je menší než ona zvolená konstanta. Takto upravený algoritmus potom již vrací aproximaci Wassersteinovy vzdálenosti pro dva zadané obrázky.

2.3 Structural dissimilarity

Nyní potřebujeme z indexu SSIM získat metriku, resp. alespoň aby byla splněna podmínka, že když se dva obrázky blíží k sobě, tak jejich vzdálenost klesá. K tomu může dobře posloužit konstrukce *DSSIM* (structural dissimilarity):

$$\text{DSSIM}(x, \tilde{x}) = \frac{1 - \text{SSIM}(x, \tilde{x})}{2}. \quad (2.21)$$

Máme vlastnost, že

$$\text{DSSIM}(x, \tilde{x}) = 0 \iff x = \tilde{x}, \quad (2.22)$$

která plyne z vlastnosti

$$\text{SSIM}(x, \tilde{x}) = 1 \iff x = \tilde{x}. \quad (2.23)$$

Kapitola 3

Adversariální vzorky a jejich tvorba

Adversariální vzorky jsou fenomén spjatý s algoritmy strojového učení. Jedná se v zásadě o jednoduchou možnost, jak takový algoritmus strojového učení lze rozbít. Ve zkratce lze na adversariální vzorky nahlédnout jako na data taková, že při vstupu do modelu strojového učení produkují aplikované algoritmy daného modelu špatné výsledky, a to i přes to, že se v jistém slova smyslu neliší od dat, které byly použity pro samotné učení. V této kapitole se potom zaměříme na korektnější formulaci toho, čím vlastně adversariální vzorky jsou. Tohoto pak docílíme při výběru úlohy klasifikace obrázků a následném výběru řešení tohoto problému, a to pomocí neuronových sítí.

3.1 Úvod do problematiky klasifikačních neuronových sítí

Pro účely definice pojmu adversariálního vzorku, který je svým charakterem pro tuto práci stěžejní, je potřeba uvést kontext, ve kterém o adversariálních vzorcích vůbec lze mluvit. Jak již bylo uvedeno v úvodu této kapitoly, vybrali jsme jednu z klasických úloh strojového učení, to jest klasifikaci obrázků. Jedná se o problém, kdy máme nějakou množinu vstupů neboli *vzorků* a máme za úkol ke každému z nich přiřadit třídu, do které tento vzorek náleží. Vlastně máme za úkol rozdělit danou množinu obrázků na předem známý počet podmnožin, které nazýváme třídy.

Mějme tedy za úkol klasifikovat dané obrázky do m tříd. Např. mějme za úkol na základě šedotónového obrázku s číslicí říci, jaká že číslice je na daném obrázku vyobrazená. Máme-li dostatečný počet vzorků, o kterých víme, do jaké třídy náleží, můžeme využít různých metod strojového učení. V úvodu této kapitoly jsme potom nastínily použití neuronových sítí pro řešení této úlohy.

Neuronová síť je potom v zásadě univerzální aproximátor spojitých zobrazení z kompaktních podmnožin \mathbb{R}^n do \mathbb{R}^m v našem případě. Dále požadujeme od neuronové sítě, aby měla předem danou strukturu, typicky potom strukturu složeného zobrazení, kde dílčí zobrazení jsou co do výpočtu jednoduchá. Stěžejní pak je pro neuronovou síť fakt, že tato dílčí zobrazení, ze kterých neuronová síť sestává, mají v předpisu parametry. Tyto parametry potom podléhají změnám při procesu učení neuronové sítě. Pro více informací o možných architekturách neuronových sítí lze nahlédnout do [1].

Tedy, vytvoříme zobrazení $F_\theta : X \rightarrow Y$, kde za X bereme množinu všech možných vzorků, v případě klasifikace číslic na obrázku právě všechny možné obrázky příslušného rozměru. Dále za Y bereme množinu všech diskrétních pravděpodobnostních rozdělení na třídách, tedy v případě klasifikace číslic to může být:

$$Y = \left\{ y \in \mathbb{R}^{10} \mid \forall i = 1, \dots, 10 : y_i \geq 0 \wedge \sum_{i=1}^{10} y_i = 1 \right\}. \quad (3.1)$$

Pro úplnost, číslo 10 v definici (3.1), protože máme deset číslic. Kdybychom bývali zůstali u problému klasifikace do m tříd, namísto čísla 10 bychom měli m .

F_θ je potom daný model neuronové sítě parametrizovaný pomocí parametrů θ . Vhodné parametry θ se potom volí pomocí procesu *učení*, což je řešení následujícího optimalizačního problému:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta), \quad (3.2)$$

kde J je funkce parametrů určující, jak moc se neuronová síť má-li dané parametry θ mýlit. Za funkci J se standardně volí agregace typu průměr dílčí ztrátové funkce L , která určuje, jak moc se neuronová síť mýlí na konkrétním vzorku.

K tomu je tedy potřeba mít trénovací datovou sadu sestávající z dostatečného počtu vzorků se správnými odpověďmi, tedy značkami. Budiž trénovací datová sada označena $\mathbb{T} = (\mathbb{X}, \mathbb{Y})$, kde $\mathbb{X} = (x^{(i)} \in X)_{i=1}^N$ a $\mathbb{Y} = (y^{(i)} \in Y)_{i=1}^N$. Potom:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N L(y^{(i)}, F_\theta(x^{(i)})). \quad (3.3)$$

Jelikož prvky Y jsou diskrétní pravděpodobnostní distribuce, za dílčí ztrátovou funkci lze vzít *křížovou entropii*:

$$L(y, \hat{y}) = - \sum_{i=1}^m y_i \log(\hat{y}_i). \quad (3.4)$$

Proces učení potom je řešení optimalizačního problému (3.2), který se typicky řeší pomocí stochastického gradientního sestupu, resp. pomocí algoritmů od stochastického gradientního sestupu odvozených.

Poslední objekt, který si zde v této sekci kapitoly zadefinujeme, je samotná klasifikace. Jedná se o funkci $C : Y \rightarrow \{1, 2, \dots, m\}$ definovanou přepisem:

$$C(y) = \underset{i \in \{1, \dots, m\}}{\operatorname{argmax}} y_i. \quad (3.5)$$

Definici v (3.5) lze interpretovat tak, že na základě celého pravděpodobnostního rozdělení $y \in Y$ volíme za třídu, kterou y reprezentuje, tu s největší pravděpodobností.

3.2 Adversariální vzorky

Jsme-li vybaveni metrikou ρ na prostoru vzorků X , lze přistoupit ke korektnímu zadefinování adversariálních vzorků.

Nejprve však definujme vzorek *benigní*. Jedná se o takový vzorek $x \in X$, že $C(F_\theta(x)) = C(y)$, kde y je pravděpodobnostní distribuce na třídách reprezentující třídu vzorku x . Je to tedy správně klasifikovaný vzorek.

Adversariální vzorek potom je takový vzorek \tilde{x} , že k němu existuje benigní vzorek x tak, že vzorek \tilde{x} je podobný vzorku x , ale vzorek \tilde{x} je špatně klasifikovaný. Podobnost lze matematicky vyjádřit způsobem, že vzdálenost \tilde{x} od x je malá ve smyslu:

$$\rho(x, \tilde{x}) \leq \kappa, \quad (3.6)$$

kde κ je pevně zvolený číselný práh. Špatnou klasifikaci lze pak vyjádřit jako

$$C(F_\theta(\tilde{x})) \neq C(F_\theta(x)). \quad (3.7)$$

Problematika adversariálních vzorků představuje v oboru strojového učení úskalí, neboť vhodně zvolený adversariální vzorek dokáže v praxi, kde algoritmy strojového učení mohou sehrávat roli při automatizaci bezpečnostně kritických úkolů, daný algoritmus rozbít.

Zatím jsme adversariální vzorky představili pouze jako teoretický koncept. V následujících sekcích textu se podívejme na jejich konkrétní tvorbu a v jedné z následujících kapitol i na praktickou ukázkou takových adversariálních vzorků.

3.3 Tvorba adversariálních vzorků

Dostaneme-li benigní vzorek x s pravdivou značkou y a máme-li za úkol k němu pro daný klasifikátor najít adversariální vzorek \tilde{x} , mějme v první řadě na mysli, že chceme, zachovat podobnost x a \tilde{x} , tak aby oba vzorky měly stejnou třídu reprezentovanou značkou y . Tedy jedna část úkolu bude minimalizovat $\rho(x, \tilde{x})$, tak aby $\rho(x, \tilde{x}) \leq \kappa$. V další části úkolu máme na výběr ze dvou možností.

Jedna možnost spočívá v tom, že si vybereme falešnou značku \tilde{y} , která reprezentuje jinou třídu než y . Dále se budeme snažit pohybovat s \tilde{x} tak, abychom $F_\theta(\tilde{x})$ přiblížili k \tilde{y} , budeme tedy minimalizovat $L(\tilde{y}, F_\theta(\tilde{x}))$, kde L je dílčí ztrátová funkce na množině značek. Tento princip se pak zazývá cílený adversariální útok.

Druhý přístup spočívá v myšlence, že chceme klasifikaci $F_\theta(\tilde{x})$ co nejvíce vzdálit od původní správné značky, tedy že chceme maximalizovat $L(y, F_\theta(\tilde{x}))$. Případně lze uvážit minimalizaci $-L(y, F_\theta(\tilde{x}))$. Tato úvaha potom vede na necílený adversariální útok.

Jak potom tato dvě kritéria (podobnost adversariálního vzorku k benignímu a špatná klasifikace) složíme dohromady? Opět máme na výběr ze dvou možností. Uveďme je na příkladu necíleného adversariálního útoku.

První přístup spočívá maximalizaci $L(y, F_\theta(\tilde{x}))$ za podmínky, že $\rho(x, \tilde{x}) \leq \kappa$. Tedy:

$$\tilde{x} = \underset{\hat{x}}{\operatorname{argmax}} L(y, F_\theta(\hat{x})), \text{ kde } \rho(x, \hat{x}) \leq \kappa. \quad (3.8)$$

Tento problém lze snadno řešit pro volbu $\rho(x, \tilde{x}) = \|x - \tilde{x}\|_\infty$ pomocí metod jako je *PGD* ([16]; zkratka anglického *projected gradient descent*).

Druhý přístup spočívá regularizaci vazebné podmínky problému (3.8). Tedy:

$$\tilde{x} = \underset{\hat{x}}{\operatorname{argmin}} \rho(x, \hat{x}) - \lambda \cdot L(y, F_\theta(\hat{x})). \quad (3.9)$$

Tento přístup se nazývá metoda *CW* (*Carlini-Wagner*, [10]).

Jak nastavit parametr λ ? Bud' můžeme parametr λ chápat jako hyper-parametr, tedy jako něco, co musíme ručně ladit, nebo lze hledat optimální hodnotu parametru λ vhodně vybraným kritériem. Myšlenka je potom následující: Chtějme najít adversariální vzorek co nejblíže původnímu benignímu vzorku. Potom v optimalizačním problému (3.9) potřebujeme, aby byl kladen větší důraz na první člen $\rho(x, \hat{x})$, tedy aby λ bylo co nejmenší. Zároveň ale potřebujeme, aby výsledek byl nesprávně klasifikován.

3.4 Robustnost neuronové sítě

Předvedli jsme jev existence adversariálních vzorků a z jeho vlastní povahy je zřejmé, že se jedná o nežádoucí jev. Na tento fenomén potom odpovídá *robustní strojové učení*, které se ve své podstatě snaží při učení neuronové sítě danou neuronovou sít' naučit tak, aby, pokud možno, k existenci adversariálních vzorků nedocházelo. Obor robustního strojového učení potom nabízí řadu metod, které k tomu mohou dopomoci.

Jednou z takovýchto metod je úprava účelové funkce optimalizované při učení dané neuronové sítě v následujícím smyslu. Vyjděme z předpokladu, že trénovací datová sada obsahuje takové vzorky, že jsou správně klasifikovány příslušnou značkou nejen samotné vzorky, ale i jejich bezprostřední kulová okolí s poloměrem κ , a to vzhledem k nějaké metrice ρ . Potom lze problém hledání optimálních parametrů neuronové sítě $\hat{\theta}$ přeformulovat následovně:

$$\hat{\theta} = \operatorname{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^N \max_{\rho(\tilde{x}, x^{(i)}) \leq \kappa} L(y^{(i)}, F_{\theta}(\tilde{x})). \quad (3.10)$$

Existuje samozřejmě více metod robustního strojového učení. Otázkou ovšem je, jak takové metody zařadit mezi ostatní. Tak jinak, jak lze srovnat danou metodu robustního strojového učení s ostatními metodami? Potřebujeme proto ideálně číselné vyjádření toho, jak si na tom daná neuronová síť, která byla učena daným algoritmem robustního strojového učení, stojí ve smyslu náchylnosti na přítomnost jevu adversariálních vzorků.

3.4.1 Přístup knihovny Foolbox

Jednou z programovacích knihoven, která se snaží osvětlit tuto tematiku a přinést nástroj pro výše nastíněné měření robustnosti modelu neuronové sítě, je knihovna *Foolbox* [4]. Její přístup spočívá v implementaci pěti základních stavebních kamenů pro tvorbu adversariálních vzorků. Jsou to:

- *Model*, implementace rozhraní, které zajišťuje kompatibilitu knihovny s populárními knihovnami strojového učení (jako je i *PyTorch*);
- *kritérium*, totiž pravidlo, podle kterého se rozhoduje, zda daný vzorek je adversariální či nikoliv;
- *metrická vzdálenost*, to jest funkce, která vyjadřuje velikost perturbace potenciálního adversariálního vzorku (rozdíl $\tilde{x} - x$, užijeme-li zavedeného značení);
- *algoritmus útoku*, způsob, jakým budou potenciální adversariální vzorky vyráběny;
- samotná *adversariální perturbace*, což je výsledek algoritmu útoku.

Za komentář stojí, jaká že kritéria mohou být užita k určení adversariality vzorku. S jedním jsme se již setkali v minulých kapitolách, totiž kritérium *nesprávné klasifikace*, které spočívá ve vyhodnocení vzorku jako adversariálního právě při určení modelu, že daný vzorek je v jiné třídě než původní vzorek, podle kterého je vzorek adversariální tvořen. Nemusíme zůstat pouze u tohoto kritéria. Další kritéria lze odvodit při hlubším studiu pravděpodobnostního rozdělení, které model produkuje. Takže např. kritérium *top-k nesprávné klasifikace* spočívá v tom, že vzorek je adversariální, pokud původní třída není mezi k nejpravděpodobnějšími třídami.

Za zmínku též stojí fakt, že knihovna *Foolbox* implementuje celou řadu různých adversariálních útoků.

3.4.2 Přístup knihovny RobustBench

Další programovací knihovnou, která se věnuje tématu robustnosti neuronových sítí je knihovna *RobustBench* [5]. Tato knihovna jde o krok dál než knihovna *Foolbox*, neboť její snahou je vyvinout jednotnotný test, který pro všechna nastavení produkuje jediné číslo, které lze tudíž hladce porovnat s ostatními výsledky. Těchto testů je několik druhů, totiž pro datové sady *CIFAR-10*, *CIFAR-100* [13] a *ImageNet* [14]. Následně podle zkoumané metriky jsou testy pro l_{∞} či l_2 útoky. Výsledné číslo se potom

nazývá *robustní úspěšnost* (z angl. *robust accuracy*), jehož vyhodnocení spočívá ve vyčíslení průměrné úspěšnosti klasifikace poškozených vzorků zkoumaným modelem v daném nastavení experimentu. Tyto poškozené vzorky jsou postupně generovány procesem *AutoAttack* [15], který spočívá v postupném provádění čtyř typů adversariálních útoků. Nejprve benigní vzorky projdou úpravou v podobě *projected gradient descent (PGD)* [16] s adaptivní velikostí kroku a ztrátou křížové entropie. Dále vzorky, které zůstanou správně klasifikované projdou obdobně procesem PGD, ale s jinou ztrátovou funkcí, a to ztrátou rozdílu podílů hodnot funkce logit. Poté je proveden *cílený FAB útok* [17], následně *black-box square attack* [18]. Následně se výsledky agregují v již zmíněnou adversariální úspěšnost. Tím pádem lze metody robustního strojového učení mezi sebou porovnávat.

Kapitola 4

Implementace metrik vizuální podobnosti

Součástí této práce je potom i samotná implementace metrik vizuální podobnosti uvedených a popsáných v předchozích kapitolách. Tato implementace je pak koncipována tak, aby bylo možné naimplementované metriky použít pro tvorbu tzv. adversariálních vzorků. Toto téma pak bude více osvětleno v jedné z následujících kapitol. Lze však nastínit, že se jedná o jednu z úloh strojového učení.

Pro úlohy strojového učení je potom stěžejní, že je mnohdy lze provádět v paralelním nastavení. Takovým úkolem potom může například být stochastický gradientní sestup, resp. od něho odvozené numerické algoritmy učení neuronových sítí. Pro tuto úlohu je totiž typické počítat výstup té samé neuronové sítě pro více vzorků, přičemž výstup neuronové sítě pro daný vzorek je nezávislý na běhu pro jiný vzorek či jakékoliv jiné informaci ohledně postupu neuronové sítě v instanci jiného vzorku. Proto je snadné tuto úlohu paralelizovat.

S tímto na mysli vznikla programovací knihovna *PyTorch* [21] pro prostředí programovacího jazyka *Python* [20]. Použití knihovny *PyTorch* je potom pro tuto práci klíčové. Proto i metriky vizuální podobnosti, kterými se tato práce z nemalé části zabývá, jsou implementovány za využití prostředků, které tato knihovna nabízí, a principů, které tato knihovna prosazuje.

4.1 Veřejné úložiště kódu

Pro zpřístupnění naimplementovaných metrik vizuální podobnosti bylo zvoleno úložiště zdrojových kódů GitHub, které je kompatibilní se systémem správy verzí Git. Samotný repozitář, který kromě samotného textu této práce obsahuje taktéž i onu implementaci metrik vizuální podobnosti, lze nalézt na webové adrese <https://github.com/pavel-jaks/robust-metrics>.

Repozitář obsahuje adresář *TeX*, v němž se nachází tento text. Dále repozitář obsahuje stěžejní adresář *code*, ve kterém se nachází v adresáři *metrics* soubor *metrics.py*. V tomto souboru je potom samotná implementace metrik vizuální podobnosti.

4.2 Koncepty pro použití kódu

Pro použití implementace metrik vizuální podobnosti, která je součástí této práce, je stěžejní mít osvojeny některé programovací koncepty. V první řadě se jedná o zvládnutí programovacího jazyka *Python*. Tento programovací jazyk je potom dynamicky typovaným jazykem, ve kterém se mísí několik paradigmat programování. Mezi tato paradigmata patří objektově orientované programování, dále pak proceduální či funkcionální programování.

Pro knihovnu PyTorch, která je využita v implementaci metrik vizuální podobnosti, je potom klíčové ono objektově orientované programování. Z tohoto důvodu jsou všechny metriky napsány jako třídy, které jsou podděny od abstraktní třídy, jež je součástí knihovny PyTorch, totiž od třídy *Module*, která se v rámci knihovny PyTorch nachází v modulu s názvem *nn*. Tato třída implementuje speciální metodu programovacího jazyka Python, jež se jmenuje `__call__`. Její implementace nějakou třídou potom opravňuje uživatele dané třídy k zacházení s instancí této třídy jako s funkcí. Toto je praktika knihovně PyTorch vlastní.

Třída *Module* potom v rámci implementace metody `__call__` spouští abstraktní metodu *forward*. Tato metoda je potom přepsána v třídách implementující abstraktní třídu *Module*, tedy i třídou *Metric*, která je базovou abstraktní třídou pro třídy metrik vizuální podobnosti a součástí kódu příslušícího této práci.

Další vlastností knihovny PyTorch je její přístup k masivní paralelizaci. Pro úlohy strojového učení je totiž typické, jak již bylo výše zmíněno, že je chceme provádět na vícero datech zároveň. Proto je pro tuto knihovnu standardem implementovat algoritmy tak, aby byly spustitelné pro celé dávky dat. K tomuto bylo přihlédnuto i v této práci, neboť třídy metrik nepočítají vzdálenost mezi jedinou dvojicí obrázků, nýbrž mezi dávkou dvojic.

Neodmyslitelnou součástí knihovny PyTorch je potom její datový typ *Tensor*, což je třída, která zapouzdřuje v podstatě pole čísel libovolných rozměrů. Tato třída dále implementuje mnoho užitečných metod pro práci s poli čísel a přepisuje chování mnoha operátorů. Aby tedy bylo možné tyto funkcionality použít pro výpočet metrik vizuální podobnosti mezi páry obrázků, je metoda *forward* tříd metrik napsána tak, že jako argumenty přijímá dva objekty právě typu *Tensor*.

Dále implementace metrik vizuální podobnosti umožňuje před samotným výpočtem provést libovolnou transformaci obrázků, která se předává metrice jako parametr konstruktoru. Příkladem může být přeškálování obrázků, aby měly jednotnou velikost.

4.3 Příklad použití metrik vizuální podobnosti

Nyní lze tedy přistoupit k nastínění použití kódu této práce. Po správném importu modulu *metrics* či jeho několika exportovaných objektů, může uživatel zkonstruovat instanci metriky, přičemž zadá transformaci a parametry metrice vlastní. Dále pak snadno použije metriku tak, že proměnnou, ve které je instance metriky uložena, zavolá jako funkci s dvěma parametry, kterými jsou instance třídy *Tensor*.

Uvažme, že chceme spočítat aproximaci Wassersteinovy vzdálenosti mezi náhodnými obrázky velikosti 20×20 . Kód by potom mohl vypadat jako v Listing 4.1.

Listing 4.1: Příklad výpočtu aproximace Wassersteinovy vzdálenosti

```

1 # import knihovny PyTorch
2 import torch

4 # import tridy pro vypocet metriky
5 from metrics import WassersteinApproximation
6
7 # nahodne obrazky
8 # Tensor rozmeru: davka x pocet kanalu x sirka x vyska
9 image1 = torch.rand(1, 1, 20, 20)
10 image2 = torch.rand(1, 1, 20, 20)

12 # instance metriky
13 metric = WassersteinApproximation(regularization=5)
14
15 # samotny vypocet metriky
16 distance = metric(image1, image2)

```

Samotná dokumentace ke všem implementovaným metrikám, tedy i význam jednotlivých parametrů a jejich korespondence s matematickými koncepty metrik vizuální podobnosti, je dostupná v repozitáři této práce v souboru [README.md](#) v adresáři [metrics](#) adresáře [code](#).

4.4 Testování naimplementovaných metrik

Pro účely ověření správnosti implementace vybraných metrik vizuální podobnosti jsou součástí této práce i testy. Tyto testy jsou napsány tak, aby byly kompatibilní s veřejně dostupnou knihovnou *Pytest*. Stěžejní je zejména otestovat metriku *DSSIM* a *aproximaci Wassersteinovy metriky*, neboť kód implementace těchto metrik vizuální podobnosti je relativně dlouhý a je snadné při jeho psaní udělat chybu.

Pro otestování funkčnosti implementací metrik vizuální podobnosti byl zvolen následující přístup. Pro několik obrázků náhodně vybraných z datové sady *MNIST* [11] a jejich poškozené verze, oba velikosti 28×28 (pro aproximaci Wassersteinovy vzdálenosti náhodné obrázky velikosti 5×5) je spočtena daná metrika vizuální podobnosti pomocí implementace, jež je součástí této práce, a pomocí alternativního kódu, který typicky neoperuje nad datovými typy knihovny *PyTorch*. Následně jsou tyto dvě hodnoty srovnány, a pokud se neliší o více než pevně stanovenou toleranci (10^{-5}), tak se test považuje za úspěšně splněný.

Za alternativní kusy kódu, kterými ověřujeme správnost implementace spjaté s touto prací, je přednostně volena implementace těch samých metrik, byť pro jiné datové typy, které jsou k dispozici v jiných veřejně dostupných softwarových knihovnách. Takto byly napsány testy pro metriky *MSE*, *DSSIM* a *PSNR* za pomoci knihovny *scikit-image* [25].

Kód pro testování aproximace Wassersteinovy metriky spočívá v přímém řešení optimalizačního problému (2.18) a následného dosažení jeho řešení do (2.17). K vyřešení tohoto problému konvexní optimalizace je používána knihovna *CVXPY* [26, 27], která na základě datových typů knihovny *NumPy* [28] takové problémy řeší.

Takto byly napsány čtyři testovací funkce, jež ověřují správnost implementací metrik vizuální podobnosti *MSE*, *DSSIM*, *NPSR* a aproximace Wassersteinovy vzdálenosti. Všechny tyto testy byly úspěšně splněny a potvrdily korektnost implementací oněch metrik vizuální podobnosti.

Kapitola 5

Porovnání metrik vizuální podobnosti

V minulých kapitolách jsme představili různé metriky vizuální podobnosti. Tyto metriky se od sebe navzájem liší, a to jak v matematické podstatě, tak ve svých vlastnostech, jako je například výpočetní časová složitost. Proto se v této kapitole podíváme, jak je která metrika časově výpočetně náročná.

5.1 Časová náročnost výpočtu

Některé metriky vizuální podobnosti spočívají v zásadě na sečtení mocnin rozdílů hodnot pixelů v příslušných souřadnicích. Lze proto očekávat, že jejich výpočet bude rychlejší, než například výpočet Wassersteinovy metriky, resp. její duální-Sinkhornovy aproximace, výpočet které potom spočívá v numerickém řešení netriviálního optimalizačního problému.

Udělejme tedy experiment, který spočívá ve spuštění výpočtu dané metriky vizuální podobnosti, resp. naší implementace, a změření, jak dlouho samotný výpočet trval. Použijeme přitom paradigma vlastní knihovně *PyTorch* [21], které spočívá v paralelizaci výpočtu tak, že počítáme vzájemné vzdálenosti více než jedné dvojice obrázků. Chtějme tedy změřit čas výpočtu metrik pro po řadě 10 dvojic, 100 dvojic a 1000 dvojic černobílých obrázků o rozměrech 28×28 , což je rozměr, který odpovídá obrázkům datové sady MNIST [11]. Z této datové sady zároveň jeden obrázek z páru obrázků, jejichž vzdálenost měříme, vybereme. Druhý obrázek v páru bude náhodně zašuměný první obrázek. Takovýto výpočet pak pro každou metriku provedeme desetkrát a vezmeme průměrnou délku výpočtu.

Mezi metriky, jejichž časovou výpočetní náročnost speciálně zkoumáme, jsme zařadili metriku založenou na normách l_1 , l_2 , l_∞ a pseudonormě l_0 . Dále pak několik metrik vizuální podobnosti založených na indexu SSIM. Zařadili jsme sem structural dissimilarity s neomezenou velikostí okna, dále pak structural dissimilarity s velikostmi okna po řadě 5, 10 a 20. A jako poslední zkoumanou metriku jsme použili Wassersteinovu metriku, resp. její aproximaci.

V Tabulce 5.1 jsou potom samotné výsledky tohoto experimentu. Jak lze zpozorovat z dat, metriky založené na l_p normách a pseudonormě l_0 jsou vypočteny takřka okamžitě (z pohledu lidského uživatele) a to odpovídá intuici.

Metrikami náročnějšími na výpočet jsou potom metriky založené na indexu SSIM, kde mimo jiné roli ve výsledném čase hraje i velikost okna použitého při výpočtu. Použijeme-li neomezenou velikost okna, tedy velikost obrázku je velikostí okna, lze nahlédnout, že výpočet je méně časově náročný. Je to logické, neboť namísto výpočtu několika vlastních indexů SSIM mezi výřezy obrázků a jejich následného průměrování, počítáme jen jeden jediný index SSIM.

Poslední změřenou metriku je výpočetně nejnáročnější aproximace Wassersteinovy metriky, jejíž výpočet pro deset párů probíhal zhruba deset sekund. Tato metrika je tedy velmi drahá ve smyslu délky výpočetního času.

Metrika	10 párů	100 párů	1000 párů
l_1	$7,036 \cdot 10^{-4}$ s	$1,902 \cdot 10^{-3}$ s	$1,819 \cdot 10^{-2}$ s
l_2	$1,508 \cdot 10^{-3}$ s	$2,846 \cdot 10^{-3}$ s	$2,183 \cdot 10^{-2}$ s
l_∞	$1,301 \cdot 10^{-3}$ s	$3,312 \cdot 10^{-3}$ s	$2,696 \cdot 10^{-2}$ s
l_0	$9,138 \cdot 10^{-4}$ s	$3,584 \cdot 10^{-3}$ s	$3,134 \cdot 10^{-2}$ s
<i>DSSIM</i>	$7,756 \cdot 10^{-3}$ s	$2,365 \cdot 10^{-2}$ s	$1,673 \cdot 10^{-1}$ s
<i>DSSIM</i> ₅	$7,096 \cdot 10^{-1}$ s	1,018 s	4,242 s
<i>DSSIM</i> ₁₀	$4,808 \cdot 10^{-1}$ s	1,126 s	8,743 s
<i>DSSIM</i> ₂₀	$1,537 \cdot 10^{-1}$ s	$7,804 \cdot 10^{-1}$ s	5,418 s
<i>Wasserstein</i>	10,95 s	92,03 s	841,0 s

Tabulka 5.1: Časy výpočtu jednotlivých metrik v sekundách pro různý počet párů.

5.2 Porovnání implementací s implementacemi veřejných knihoven

Mezi netriviální implementace metrik vizuální podobnosti jistě patří aproximace Wassersteinovy vzdálenosti a metrika DSSIM. Proto se podívejme, jaké přístupy k implementaci zvolili jiné knihovny.

5.2.1 Aproximace Wassersteinovy vzdálenosti

Mezi všemi implementacemi Wassersteinovy vzdálenosti, resp. její aproximace je význačná ta v knihovně *GeomLoss* [29]. Její význačnost spočívá v tom, že je navržena, aby byla kompatibilní s knihovnou PyTorch [21], což je i v našem úsilí. Proto naši implementaci srovnáme s implementací, již lze nalézt právě v knihovně *GeomLoss*.

Knihovna *GeomLoss* bohužel nepodporuje přímý výpočet aproximace Wassersteinovy vzdálenosti mezi dvěma obrázky. Přístup knihovny *GeomLoss* je totiž takový, že počítá vzdálenost mezi dvěma rozděleními pravděpodobnosti, jež jsou odhadnuta na základě realizací náhodných veličin popsanych těmito rozděleními. Této knihovně se tedy předávají tzv. *mraky bodů* (z angl. *cloud of points*). Konverze mezi obrázkem (tedy vlastně histogramem takového rozdělení) a mrakem bodů samozřejmě možná je, ovšem následný výpočet, byť pro malé šedotónové obrázky, je velmi paměťově náročný.

Výhodou implementace knihovny *GeomLoss* ovšem je, že je lépe přizpůsobená běhu na grafické kartě. Toto potom je obrovským plusem, který naše implementace nemá. Vzpomeneme-li si na numerický algoritmus pro výpočet aproximace Wassersteinovy vzdálenosti, který je uvedený ve druhé kapitole, zjistíme, že pro výpočet vzdálenosti mezi vektory μ a ν nejprve určujeme, které prvky vektoru μ jsou nulové, a následně s těmito nulovými prvky již nepočítáme. Nyní si uvědomme, že každý takovýto vstup μ může mít počet nulových prvků různý od ostatních vstupů. Tedy potřebovali bychom do paměti uložit vícerozměrné pole, které ve druhé dimenzi má různou délku. To ale knihovna PyTorch pro svůj datový typ *Tensor*, pomocí kterého vlastně vzdálenosti počítáme, neumožňuje. Proto jsme se uchýlili k výpočtu metrik mezi páry obrázků v sekvenčním módu, tedy v módu, který nedokáže plně využít potenciál akceleratorů grafických karet.

5.2.2 DSSIM

Chceme-li porovnat naši implementaci metriky vizuální podobnosti založené na indexu SSIM, tedy metriky DSSIM, máme k dispozici možnost porovnat ji s implementací, jež byla použita pro testování, totiž s implementací knihovny *scikit-image* [25]. Ta ovšem nepodporuje datové typy vlastní knihovně PyTorch.

Další možností je např. knihovna *pytorch-msssim*. Tato knihovna je specializovaná na výpočet právě indexu SSIM, a to za použití datových typů dostupných v knihovně PyTorch.

Další knihovnou, se kterou lze naši implementaci srovnat, je *pytorch-ignite* [30], která je svou podstatou nadstavba pro trénování neuronových sítí pomocí knihovny PyTorch. Součástí této knihovny je také implementace metriky SSIM.

Po stránce výpočetního času jsou obě knihovny, které jsou napsány nad knihovnou PyTorch, efektivnější. A to zhruba dvacetkrát. Zatímco naše implementace působí pro dávku o tisíci párech obrázků v řádu středních jednotek sekund pro malou velikost podokna, jak implementace knihovny *pytorch-msssim*, tak implementace knihovny *pytorch-ignite* to samé vypočtou za nižší desetiny sekundy.

Kapitola 6

Výsledky tvorby adversariálních vzorků

6.1 Detaily implementace adversariálního útoku

Pro předvedení adversariálního útoku jsme zvolili metodu *CW* s pevným parametrem λ . Pro připomenutí: Metoda útoku *CW* je založena na řešení problému

$$\tilde{x} = \underset{\hat{x}}{\operatorname{argmin}} \rho(x, \hat{x}) - \lambda \cdot L(y, F_{\theta}(\hat{x})). \quad (6.1)$$

Tento problém jsme se rozhodli řešit *znaménkovým gradientním sestupem* (v angl. literatuře uváděný jako *sign gradient descent*), a to s pevným počtem iterací (100) a s pevným krokem 10^{-2} , který je pro obrázky lineárně přeškálované do intervalu $[0, 1]$ akorát dostačující. Jako ukázkovou úlohu jsme zvolili klasifikaci číslic na datové sadě *MNIST* [11] (jednokanálové obrázky o rozměru 28×28), která je v komunitě strojového učení notoricky známá. Pro řešení původního problému klasifikace jsme natrénovali (algoritmem *RMSPProp* [12]) jednoduchou konvoluční neuronovou síť, která na testovací datové sadě dosáhla úspěšnosti 96,9%. Inicializace startovního bodu znaménkového gradientního sestupu probíhala následovně: Inicializace spočívala v náhodné inicializaci každého pixelu v okolí hodnoty 0,5 na základě realizace náhodné veličiny s rovnoměrným rozdělením $U(0, 3; 0, 8)$.

6.2 Srovnání útoků pro různé metiky

Přistupme nyní k vyhodnocení experimentu, který spočíval v následujícím: Pro vybranou metriku vizuální podobnosti a pevně zvolené λ , které se v logaritmu mění lineárně, tj. prochází hodnoty $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$, jsme spustili generování adversariálních vzorků pro 1000 benigních vzorků z datové sady *MNIST* a zjistili pro danou metriku vizuální podobnosti a dané λ průměrnou úspěšnost útoku, dále ale i pro všechny metiky průměrnou l_2 vzdálenost vygenerovaných vzorků od původních benigních vzorků. Jen poznamenejme, že pro aproximaci Wassersteinovy vzdálenosti bylo použito z důvodu výpočetní náročnosti vzorků 200.

V Tabulce 6.1 jsou již uvedena čísla reprezentující výše představenou úspěšnost útoku. V Tabulce 6.2 jsou potom hodnoty odpovídající průměrům l_2 vzdáleností získaných v experimentu. Jelikož metrika vizuální podobnosti DSSIM je založena na indexu SSIM, který je počítán postupně v podoknech a následně agregován, je u názvu této metiky uvedena i velikost tohoto podokna, a to pokud je menší než velikost rozměry obrázku. Dále u aproximace Wassersteinovy vzdálenosti uvádíme i velikost regularizačního parametru ξ .

Metrika	Hodnota parametru λ						
	10^{-3}	10^{-2}	10^{-1}	1	10	10^2	10^3
l_1	0.1 %	0.0 %	0.1 %	0.5 %	89.8 %	95.1 %	99.4 %
l_2	0.4 %	0.1 %	0.1 %	87.9 %	98.1 %	100.0 %	100.0 %
$DSSIM$	2.6 %	82.0 %	94.3 %	98.9 %	99.9 %	100.0 %	100.0 %
$DSSIM_5$	42.9 %	92.3 %	96.4 %	99.5 %	99.9 %	99.9 %	100.0 %
$DSSIM_{10}$	1.5 %	65.6 %	92.6 %	97.3 %	99.9 %	100.0 %	100.0 %
$DSSIM_{20}$	2.4 %	72.2 %	92.4 %	96.9 %	99.9 %	100.0 %	100.0 %
$Wasserstein_5$	19.5 %	24.5 %	90.5 %	96.5 %	99.5 %	100.0 %	100.0 %
$Wasserstein_{10}$	3.0 %	5.0 %	80.5 %	96.5 %	100.0 %	100.0 %	100.0 %
$Wasserstein_{15}$	7.0 %	6.5 %	66.5 %	99.0 %	100.0 %	100.0 %	100.0 %

Tabulka 6.1: Úspěšnost CW útoku v závislosti na zvolené metrice a hodnotě parametru λ

Metrika	Hodnota parametru λ						
	10^{-3}	10^{-2}	10^{-1}	1	10	10^2	10^3
l_1	0.1464	0.1464	0.1463	0.1495	2.043	2.559	2.874
l_2	0.1462	0.146	0.1474	1.567	2.107	2.345	2.545
$DSSIM$	0.2005	1.412	1.958	2.308	2.531	2.748	2.973
$DSSIM_5$	3.557	7.598	8.49	8.98	9.154	9.216	9.251
$DSSIM_{10}$	0.2525	1.71	2.613	3.071	3.412	3.642	3.872
$DSSIM_{20}$	0.1893	1.201	1.939	2.311	2.604	2.848	3.091
$Wasserstein_5$	6.531	6.553	7.248	7.604	7.828	8.031	8.182
$Wasserstein_{10}$	4.359	4.35	5.06	5.575	5.873	6.021	6.208
$Wasserstein_{15}$	2.697	2.721	3.795	4.51	4.754	4.963	5.117

Tabulka 6.2: Průměrná l_2 vzdálenost vzorku z CW útoku v závislosti na zvolené metrice a hodnotě parametru λ

6.3 Adversariální útok za použití metriky indukované l_2 normou

Na Obrázku 6.1 lze spatřit vzorky vygenerované CW útokem za použití metriky indukované l_2 normou. V každém z řádků je potom prvním obrázkem původní benigní vzorek. Následující obrázky v řádku jsou potom obrázky vygenerované CW útokem pro variabilní hodnotu parametru λ . Tato hodnota parametru λ je potom po řadě 10^{-3} , 10^{-2} , 10^{-1} , 1, 10, 100 a 1000. V takovémto formátu potom jsou prezentovány i další obrázky.

Z Obrázku 6.1 si lze odnést jednoznačně potvrzení prezentované matematické intuice ohledně volby parametru λ v CW útok, který spočívá v řešení (3.9). Čím větší je totiž parametr λ , tím menší důraz je kladen na minimalizaci metriky vizuální podobnosti mezi původním benigním vzorkem a generovaným vzorkem. Proto s rostoucí hodnotou parametru λ narůstá míra vizuálního poškození oproti původnímu obrázku.

6.4 Adversariální útok za použití metriky indukované l_1 normou

Na Obrázku 6.2 lze pak spatřit vzorky vygenerované CW útokem, který používal metriku vizuální podobnosti indukovanou l_1 normou. Na první pohled při srovnání s Obrázkem 6.1 lze vidět, že vzorky generované za použití metriky indukované l_1 normou jsou poněkud kostrbatější. Tak jinak, že artefakty v okolí číslic mají výrazně větší rozdíly v hodnotách dvou sousedních pixelů.

6.5 Adversariální útok za použití metriky vizuální podobnosti DSSIM

Podívejme se nyní na vzorky vygenerované CW útokem, ve kterém roli metriky vizuální podobnosti hrála metrika vizuální podobnosti DSSIM. Předvedli jsme, že tato metrika sama o sobě je parametrizovaná velikostí výpočetního podokna. Proto se věnujme různým velikostem výpočetního podokna zvlášť.

6.5.1 Velikost výpočetního podokna 5

Tento komentář se bude věnovat jevu, který vyvstal pro DSSIM útok s parametrem velikosti výpočetního podokna 5. Podle Obrázku 6.3 bývá po tomto útoku okolí číslice silně znečištěno artefakty. Kupodivu samotná číslice a její bezprostřední okolí je relativně nedotknuté. Tento fenomén se vyskytl hlavně u volby velikosti výpočetního podokna 5. Vysvětlení zní jednoduše. Neboť je původní benigní obrázek dál od číslice roven v každém pixelu nule, pak SSIM index příslušných dvou podoken dál od číslice vychází konstantně skoro nula, a to kvůli členu $\sigma_{x\tilde{x}}$ v čitateli ve výrazu v (1.16). Není úplně nula díky konstantě C_2 , která zde vystupuje kvůli dělení. Ta je ovšem dostatečně malá, aby byla převážena druhým členem gradientu.

6.5.2 Velikost výpočetního podokna 10

V Obrázku 6.4 lze potom nahlédnout na vzorky vygenerované CW útokem za použití metriky vizuální podobnosti DSSIM s parametrem velikosti výpočetního podokna 10. Lze vidět, že tyto obrázky jsou mnohem uhlazenější, než obrázky vygenerované za použití metriky DSSIM s velikostí podokna 5.

6.5.3 Velikost výpočetního podokna 20

V Obrázku 6.5 vidíme vzorky vygenerované CW útokem za použití metriky vizuální podobnosti DSSIM s parametrem velikosti výpočetního podokna 20. Je patrné, že tyto obrázky jsou velice podobné těm, jež jsou vygenerované za použití metriky DSSIM s velikostí podokna 10.

Obrázek 6.1: Vzorky generované CW útokem za použití metriky indukované l_2 normou



Obrázek 6.2: Vzorky generované CW útokem za použití metriky indukované l_1 normou



Obrázek 6.3: Vzorky generované CW útokem za použití metriky vizuální podobnosti DSSIM s velikostí výpočetního podokna 5



Obrázek 6.4: Vzorky generované CW útokem za použití metriky vizuální podobnosti DSSIM s velikostí výpočetního podokna 10



Obrázek 6.5: Vzorky generované CW útokem za použití metriky vizuální podobnosti DSSIM s velikostí výpočetního podokna 20

6.5.4 Velikost výpočetního podokna jako velikost celého obrázku

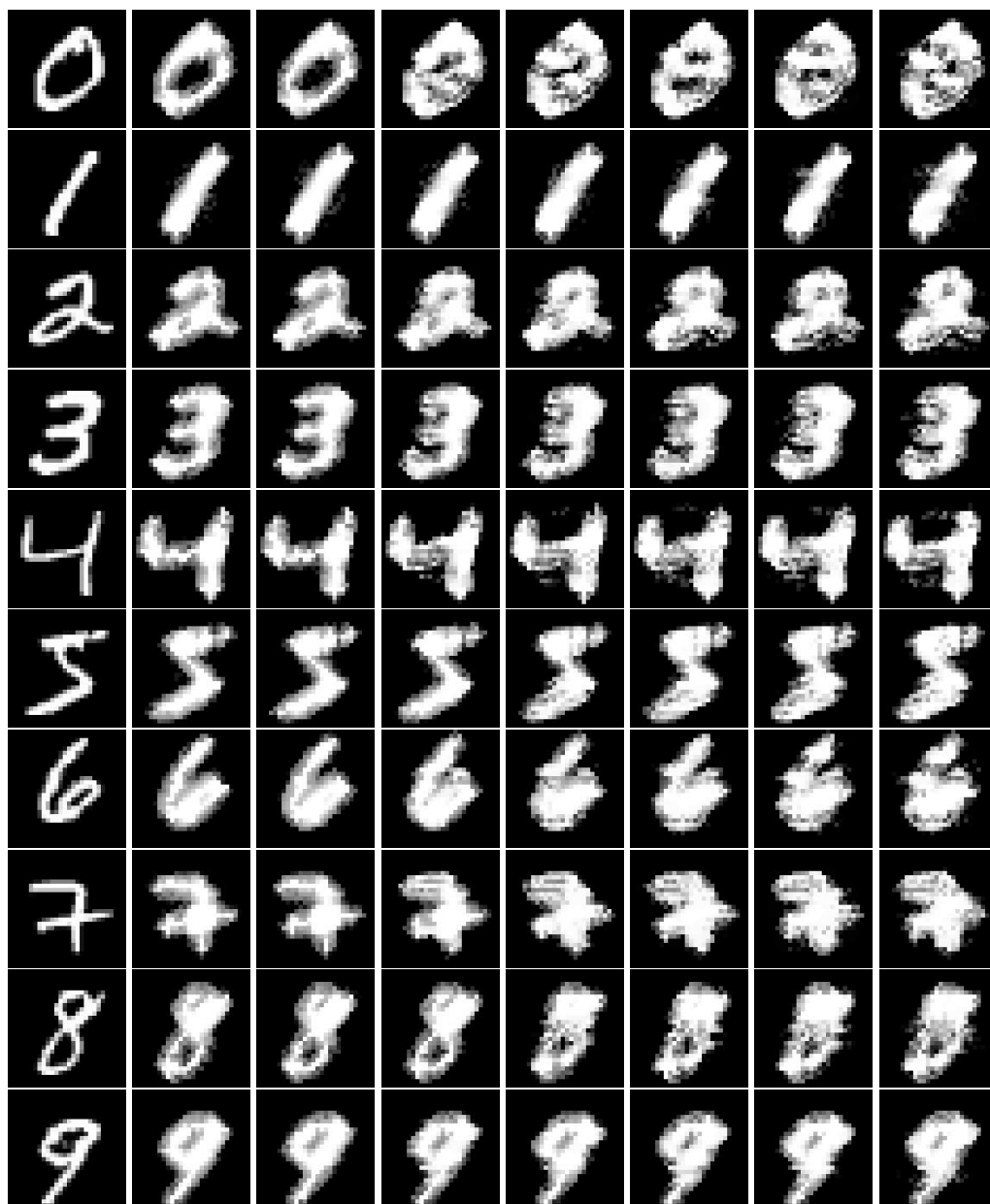
Na Obrázku 6.6 lze spatřit vzorky vygenerované CW útokem za použití metriky vizuální podobnosti DSSIM s neomezeným parametrem velikosti výpočetního podokna, resp. s velikostí 28, jež je velikostí obázků. Obdobně jako v předchozím případě jsou tyto obrázky velice podobné těm, jež jsou vygenerované za použití metriky DSSIM s velikostí podokna 10, a tedy i za použití metriky DSSIM s velikostí výpočetního podokna 20.

6.6 Adversariální útok za použití aproximace Wassersteinovy vzdálenosti

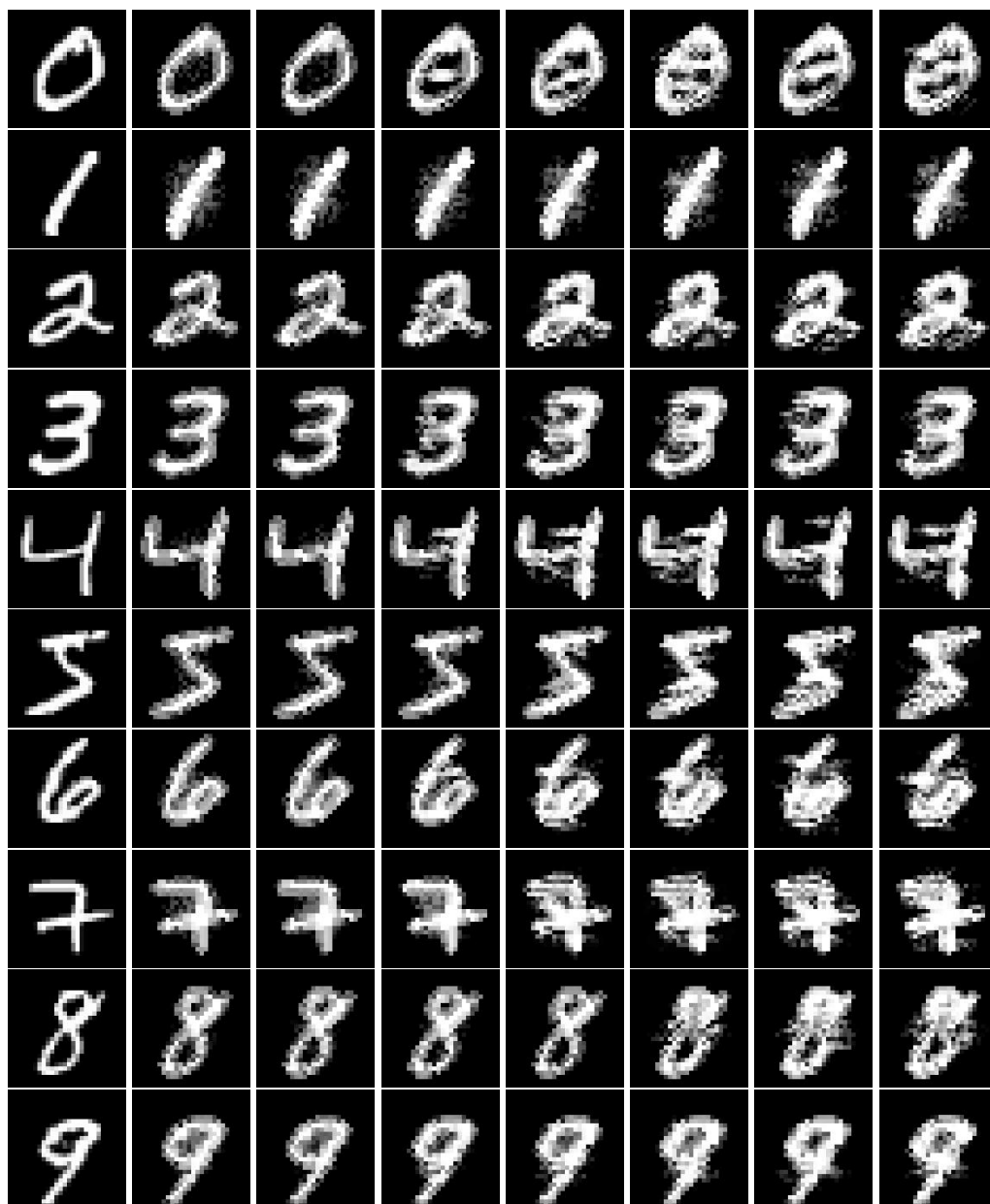
Podíváme-li se po řadě na Obrázky 6.7, 6.8 a 6.9, kde jsou zobrazeny vzorky vygenerované CW útokem za použití aproximace Wassersteinovy vzdálenosti s hodnotami regularizačního parametru ξ po řadě 5, 10 a 15, uvidíme, že tyto vzorky jsou velice odlišné od vzorků předchozích, a to obzvlášť pro hodnotu regularizace 5. Obrázky, které vznikly CW útokem za použití této metriky, jsou totiž jakoby rozmazané. S rostoucím regularizačním parametrem ξ potom tento efekt mizí a jeví se na obrázcích již povědomé artefakty v okolí číslice.



Obrázek 6.6: Vzorky generované CW útokem za použití metriky vizuální podobnosti DSSIM bez omezení velikosti výpočetního podokna



Obrázek 6.7: Vzorky generované CW útokem za použití aproximace Wassersteinovy metricky s hodnotou regularizace 5



Obrázek 6.8: Vzorky generované CW útokem za použití aproximace Wassersteinovy metricky s hodnotou regularizace 10



Obrázek 6.9: Vzorky generované CW útokem za použití aproximace Wassersteinovy metriky s hodnotou regularizace 15

Závěr

V této práci bylo nastíněno téma problematiky adversariálních vzorků z jiného úhlu pohledu než bývá zvykem. Cílem nebylo vyvinout novou metodu pro tvorbu adversariálních vzorků ani najít, jak vhodně naučit model strojového učení tak, aby byl robustní vůči adversariálním útokům. Cílem bylo nahlédnout na tuto problematiku z pohledu různorodosti metrik vizuální podobnosti a poskytnout přehled, jak volba této metriky ovlivňuje tvorbu či podobu adversariálních vzorků.

Představili jsme tedy různé metriky vizuální podobnosti a provedli jejich implementaci v programovacím jazyce *Python* [20]. Dále jsme s pomocí knihovny *PyTorch* [21] natrénováli jednoduchou neuronovou síť pro klasifikaci číslic a tuto síť použili jako cíl námi implementovaného CW útoku s možností volby užití metriky vizuální podobnosti.

Na základě výsledků lze říci, že na volbě metriky vizuální podobnosti záleží. Viděli jsme, že obrázky vygenerované CW útokem za použití aproximace Wassersteinovy vzdálenosti s menším regularizačním parametrem jsou v zásadě rozmazané původní obrázky. Dále jsme nahlédli na fakt, že CW útok využívající metriku vizuální podobnosti DSSIM s velikostí výpočetního podokna 5 vede na značné artefakty v okolí číslice, nikoliv však na poškození čitelnosti samotné číslice. Též se ukázal rozdíl mezi metrikami založenými na l_1 a l_2 normě, neboť obrázky vzniklé CW útokem využívajícím metriku založenou na l_2 normě jsou na pohled uhlazenější.

Nakonec poznamenejme, že riziko spjaté s existencí adversariálních vzorků v kontextu neuronových sítí je třeba brát v potaz, a to obzvlášť při aplikaci těchto technologií v kritických oblastech.

Literatura

- [1] I. Goodfellow, Y. Bengio, A. Courville: *Deep Learning*. MIT Press, 2016.
- [2] N. Akhtar, A. Mian, N. Kardan, M. Shah: *Advances in adversarial attacks and defenses in computer vision: A survey*. IEEE Access 9, 2021, 155161-155196.
- [3] W. Eric, F. Schmidt, Z. Kolter: *Wasserstein adversarial examples via projected sinkhorn iterations*. International Conference on Machine Learning, PMLR, 2019.
- [4] J. Rauber, R. Zimmermann, M. Bethge, W. Brendel: *Foolbox: A Python toolbox to benchmark the robustness of machine learning models*. Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning, 2017.
- [5] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, M. Hein: *RobustBench: a standardized adversarial robustness benchmark*. Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021
- [6] L. Vaserstein, *Markov processes over denumerable products of spaces, describing large systems of automata*. Problemy Peredači Informacii 5, 1969.
- [7] Z. Wang, A. C. Bovik, H. R. Sheikh: *Image Quality Assessment: From Error Visibility to Structural Similarity*. IEEE Transactions on Image Processing, ročník 13, č. 4, April 2004: s. 600–612.
- [8] M. Cuturi, *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*. Advances in Neural Information Processing Systems 26, 2013.
- [9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, *Intriguing properties of neural networks*. arXiv, 2014.
- [10] N. Carlini, D. Wagner, *Towards evaluating the robustness of neural networks*. IEEE Symposium on Security and Privacy (SP), IEEE, 2017.
- [11] Y. Lecun, C. Cortes, C. J. Burges, *The mnist database of handwritten digits*. 1998.
- [12] G. Hinton, *Neural networks for machine learning*. Coursera, video lectures, 2012.
- [13] A. Krizhevsky, G. Hinton: *Learning multiple layers of features from tiny images*. Technical Report, 2009.
- [14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei: *Imagenet: A large-scale hierarchical image database*. IEEE conference on computer vision and pattern recognition, pages 248–255, Ieee, 2009.
- [15] F. Croce, M. Hein: *Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks*. In ICML, 2020.

- [16] A. Mądry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu: *Towards deep learning models resistant to adversarial attacks*. Stat 1050 9, 2017.
- [17] F. Croce, M. Hein: *Minimally distorted adversarial examples with a fast adaptive boundary attack*. ICML, 2020.
- [18] M. Andriushchenko, F. Croce, N. Flammarion, M. Hein: *Square attack: a query-efficient black-box adversarial attack via random search*. ECCV, 2020.
- [19] R. Sinkhorn: *A relationship between arbitrary positive matrices and doubly stochastic matrices*. In The Annals of Mathematical Statistics 35, 876–879, 1964.
- [20] G. van Rossum: *Python tutorial, Technical Report CS-R9526*. Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.
- [21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems 32, 8024–8035, 2019.
- [22] E. T. Jaynes: *Information theory and statistical mechanics*. Phys. Rev. 106, 620–630, 1957.
- [23] M. Slater: *Lagrange Multipliers Revisited*. Cowles Commission Discussion Paper, 1950.
- [24] R. Sinkhorn, P. Knopp: *Concerning nonnegative matrices and doubly stochastic matrices*. Pacific J. Math. 21, 343–348, 1967.
- [25] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, scikit-image contributors: *scikit-image: image processing in Python*. PeerJ 2:e453, 2014.
- [26] S. Diamond, S. Boyd: *CVXPY: A Python-embedded modeling language for convex optimization*. Journal of Machine Learning Research 17, 1-5, 2016.
- [27] A. Agrawal, R. Verschueren, S. Diamond, S. Boyd: *A rewriting system for convex optimization problems*. Journal of Control and Decision 5, 42-60, 2018.
- [28] C. R. Harris, K. J. Millman, S. J. van der Walt, et al.: *Array programming with NumPy*. Nature 585, 357–362, 2020.
- [29] J. Feydy, T. Séjourné, F. Vialard, S. Amari, A. Trounev, G. Peyré: *Interpolating between Optimal Transport and MMD using Sinkhorn Divergences*. The 22nd International Conference on Artificial Intelligence and Statistics, 2681–2690, 2019.
- [30] V. Fomin, J. Anmol, S. Desrozières, J. Kriss, A. Tejani: *High-level library to help with training neural networks in PyTorch*. GitHub repository, GitHub, 2020.