Diplomová práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Efektivní kódování obrazu založené na učení

Efficient Learning Based Image Coding

Bc. Daniel Šafář

Vedoucí práce: Ing. Karel Fliegel, Ph.D.
Květen 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šafář** Jméno: **Daniel** Osobní číslo: **466299**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Audiovizuální technika a zpracování signálů**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Efektivní kódování obrazu založené na učení

Název diplomové práce anglicky:

Efficient Learning Based Image Coding

Pokyny pro vypracování:

Podejte přehled současných metod pro efektivní kompresi obrazu s využitím metod strojového učení a v kontextu probíhajících standardizačních aktivit JPEG. Vybrané metody implementujte a demonstrujte jejich účinnost na vhodně vybraných obrazových datech. Porovnejte účinnost komprese využívající strojového učení s konvenčními přístupy, včetně analýzy použitelnosti nástrojů pro objektivní hodnocení kvality obrazu.

Seznam doporučené literatury:

[1] Jiang, F., Tao, W., Liu, S., Ren, J., Guo, X., Zhao, D.: An End-to-End Compression Framework Based on Convolutional Neural Networks, IEEE Transactions on Circuits and Systems for Video Technology, 28 (10), 2018.
[2] Punnappurath, A., Brown, M.S.: Learning Raw Image Reconstruction-Aware Deep Image Compressors, IEEE Transactions on Pattern Analysis and Machine Intelligence, 42 (4), 2020.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Karel Fliegel, Ph.D., katedra radioelektroniky FEL

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

Datum zadání diplomové práce: **25.01.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce: **30.09.2022**

Ing. Karel Fliegel, Ph.D.	doc. Ing. Josef Dobeš, CSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	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

Tímto bych rád poděkoval vedoucímu práce Ing. Karlu Fliegelovi, Ph.D. za odborné vedení, hodnotné připomínky a jeho drahocenný čas.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 12. 5. 2021

.....

Abstrakt

Tato diplomová práce se věnuje efektivní kompresi obrazu s využitím strojového učení. Nejprve je představen stručný úvod do neuronových sítí a strojového učení, na který navazuje podrobnější rozbor jejich využití v kompresi obrazu, a to od nejstarších pokusů až po současné aktivity JPEG AI. Součástí teoretické části práce je také přehled dostupných softwarových implementací a nástrojů hlubokého učení. Dále je zařazena kapitola věnující se latentní reprezentaci obrazových dat, která z komprese založené na učení vychází. Významnou částí práce je provedení experimentu porovnávajícího konvenční obrazové kodeky (JPEG, JPEG 2000, HEVC Intra a JPEG XL) a několik metod založených na učení zejména s ohledem na kompresní účinnost. Porovnání je provedeno na vybraných testovacích datech pomocí několika metrik (PSNR, MS-SSIM, VIFP, PSNR-HVS-M, FSIMc a VMAF), přičemž výstupem jsou R–D (Rate–Distortion) křivky a integrální porovnání kompresní účinnosti kodeků pomocí Bjøntegaardovy metriky.

Klíčová slova: Strojové učení, komprese obrazu, neuronové sítě, JPEG AI, latentní reprezentace

Vedoucí práce:
Ing. Karel Fliegel, Ph.D.
České vysoké učení technické v Praze,
Fakulta elektrotechnická,
Katedra radioelektroniky,
Technická 2,
16627 Praha 6

Abstract

This master thesis is devoted to efficient image compression with the use of machine learning. A brief introduction to neural networks and machine learning is presented, then more detailed analysis on the utilization of them in image compression is conducted, namely from the first attempts to the context of ongoing JPEG AI activities. An overview of current available software implementations and deep learning tools is also provided in the theoretical part of the thesis. Then, there is a chapter devoted to latent representation of image data, which is based on learning based image compression. The key part of the thesis is an experiment comparing conventional image codecs (JPEG, JPEG 2000, HEVC Intra and JPEG XL) and several learning based methods with emphasis on compression efficiency. The comparison is performed on selected test data with several metrics (PSNR, MS-SSIM, VIFP, PSNR-HVS-M, FSIMc and VMAF). The output of the experiment are the R–D (Rate–Distortion) curves and integral comparison of compression efficiency based on Bjøntegaard metric.

Keywords: Machine Learning, Image Compression, Neural Networks, JPEG AI, Latent Representation

Title translation: Efficient Learning Based Image Coding

Obsah

Obrázky

Tabulky

Seznam zkratek

AE . . . . . Auto Encoder
ANN . . . . . Artificial Neural Network
BD . . . . . Bjøntegaard Delta
CNN . . . . . Convolutional Neural Network
CPU . . . . . Central Processing Unit
DCT . . . . . Discrete Cosine Transform
DNN . . . . . Deep Neural Network
DPCM . . . . . Differential Pulse Code Modulation
DSLR . . . . . Digital Single-Lens Reflex camera
FF . . . . . Feed Forward
FHD . . . . . Full High Definition
FPGA . . . . . Field Programmable Gate Array
FSIMc . . . . . Feature Similarity chrominance
GAN . . . Generative Adversarial Network
GPU . . . . . Graphics Processing Unit
HDR . . . . . High Dynamic Range
HEIF . . . . . High Efficiency Image File Format
HEVC . . . . High Efficiency Video Coding
HVS . . . . . Human Visual System
JPEG . . . . . Joint Photographic Experts Group
JPEG AI . . . JPEG Artificial Intelligence
LPC . . . . . Linear Predictive Coding
ML . . . . . Machine Learning
MNIST . . . . . Modified National Institute of Standards and Technology
MS-ROI . . . . . Multi-Structure Region of Interest
MS-SSIM . . . . . Multiscale Structural Similarity
MSE . . . . . Mean Squared Error
NN . . . . . Neural Network
ONNX . . . . . Open Neural Network Exchange
PCA . . . . . Principle Component Analysis
PCM . . . . . Pulse Code Modulation
PSNR . . . . . Peak Signal-to-Noise Ratio
PSNR-HVS-M . . . . . Peak Signal-to-Noise Ratio Human Visual System Masking
QP . . . . . Quantization Parameter
R–D . . . . . Rate–Distortion
RAM . . . . . Random Access Memory
RNN . . . . . Recurrent Neural Network
ROI . . . . . Region of Interest
SAO . . . . . Sample Adaptive Offset
SAR . . . . . Synthetic Aperture Radar
SD . . . . . Standard Definition
SoC . . . . . System on a Chip
SSIM . . . . . Structural Similarity
SVM . . . . . Support Vector Machine
UHD . . . . . Ultra High Definition
VIF . . . . . Visual Information Fidelity
VIFP . . . . . Visual Information Fidelity Pixel-domain version
VMAF . . . . . Video Multimethod Assessment Fusion
WCG . . . . . Wide Color Gamut
WSL . . . . . Windows Subsystem for Linux

Kapitola 1
Úvod

Již od pravěku se lidstvo snaží zachytit svět kolem sebe na nějakou formu trvalého média. Prvním zařízením vedoucím k vývoji v oblasti zachycení obrazu je dírková komora (camera obscura), o které se dochovaly záznamy již z 5. století př. n. l. ze staré Číny a kterou popsal také Aristoteles kolem roku 350 př. n. l. Tímto zařízením je možné pozorovat jev zobrazení, přičemž je založeno na průchodu světla malým otvorem a následném zobrazení převráceného (a většinou zmenšeného) obrazu na stínítku. Proto je možné ho považovat za předchůdce fotoaparátu [5].

V dnešní době jsou digitální fotoaparáty a videokamery velmi rozšířené, k čemuž přispívá i skutečnost, že jsou součástí většiny mobilních telefonů. Z tohoto důvodu vzniká v každém okamžiku obrovské množství digitálních obrazových dat, která je potřeba uložit. Lidé dnes navíc tyto fotografie a videa dále sdílí prostřednictvím sociálních médií. Dalšími významnými oblastmi, kde dochází k přenosu či ukládání obrazové informace, jsou videokomunikace, streamovací platformy, ale i vědecké aplikace, astronomie nebo lékařství.

Podle článku [1] z prosince 2019 je

- každý den sdíleno 3,2 miliardy obrázků přes sociální média,
- každou minutu nahráno 300 hodin videí na YouTube,
- nebo každý den zhlédnuta 1 miliarda hodin videí na YouTube.

V dnešní době navíc tato čísla nadále rostou a tempo růstu se neustále zvětšuje.

Z tohoto důvodu je ve všech zmíněných oblastech nesmírně důležitá efektivní komprese obrazových dat a má tak smysl zdokonalovat současné kompresní algoritmy a zabývat se vývojem nových algoritmů a adaptací rozličných přístupů.

Jedním z poměrně nových a v současné době zkoumaných pohledů na problematiku komprese obrazové informace je využití neuronových sítí NN (Neural Network) a strojového učení ML (Machine Learning).

Druhá kapitola práce obsahuje velmi stručný úvod do neuronových sítí a strojového učení, především ale podává přehled aktuálního stavu využití neuronových sítí a strojového učení v kompresi obrazu. Obsahuje také informace o souvisejících aktivitách skupiny JPEG (Joint Photographic Experts Group).

Ve třetí kapitole jsou zmíněny a porovnány dostupné softwarové implementace a nástroje, které umožňují kompresi obrazu založenou na učení, s důrazem na možné experimentální využití.

Čtvrtá kapitola je věnována využití latentní reprezentace obrazových dat, která vychází z komprese založené na učení. Kapitola obsahuje zejména přehled aktuálního stavu této problematiky a je zakončena zmínkou o pravděpodobném budoucím vývoji této oblasti.

V páté kapitole je kompletně popsán provedený experiment, který demonstruje účinnost metod komprese obrazu založených na učení a jejich srovnání s konvenčními kodeky. Experiment je proveden na vybraných testovacích obrazových datech a výstupy jsou hodnoceny pomocí několika vhodných metrik pro objektivní hodnocení kvality obrazu a také pomocí Bjøntegaardovy metriky.

Na závěr jsou pak získané poznatky z oblasti komprese obrazu založené na učení shrnuty a dále jsou popsána možná rozšíření práce, především s ohledem na využití latentní reprezentace.

Kapitola 2

Strojové učení v kompresi obrazu

V posledních dvou letech dochází k detailnímu zkoumání možnosti využití strojového učení ke kompresi obrazu s cílem nalézt a vyvinout vhodné metody a implementace vedoucí ke standardizaci v této oblasti [24].

První úvahy o využití neuronových sítí v kompresi obrazu však sahají až do období přelomu 80. a 90. let 20. století [23]. Od té doby došlo k významnému pokroku v této oblasti, studiem dané problematiky se zabývalo mnoho vědeckých týmů po celém světě, které publikovaly mnoho článků, a v této kapitole je podán jejich poměrně rozsáhlý, nikoliv však kompletní, přehled.

2.1 Neuronové sítě a strojové učení

Nejprve je však potřeba se zaměřit na stručný popis neuronových sítí a strojového učení, přičemž neuronovými sítěmi jsou myšleny umělé neuronové sítě ANN (Artificial Neural Network).

2.1.1 Neuronové sítě

Neuronová síť je skupina vzájemně propojených neuronů určitým způsobem s cílem paralelního zpracování dat. ANN vznikly ze snahy modelovat funkci lidského mozku, který se skládá z neuronů propojených axony a dendrity. Na základě tohoto propojení vznikl matematický model neuronu. Z axonů jednotlivých neuronů vystupují signály $x_0, x_1, ...$, které jsou vedeny přes synaptické štěrbiny mezi axony a dendrity do těla dalšího neuronu. Signály $x_0, x_1, ...$ jsou násobeny synaptickými váhami $w_0, w_1, ...$ reprezentujícími sílu synapse, tedy důležitost daného vstupu, a následně v těle neuronu sečteny. Navíc je k této lineární kombinaci přidána odchylka b reprezentující práh aktivity neuronu. Celý výraz je následně postoupen aktivační funkci σ, která popisuje přenosovou funkci daného neuronu. Celý tento model lze tedy popsat vztahem

$$y = \sigma\left(\sum_i x_i w_i + b\right), \tag{2.1}$$

kde y je celkový výstup neuronu, který je dále šířen axonem [13].

V neuronových sítích jsou neurony poskládány do vrstev, přičemž podle typů a množství použitých vrstev, jejich vzájemného propojení a použité aktivační funkce lze rozlišit několik typů neuronových sítí [13, 43]:

- Dopředná plně propojená neuronová síť FF (Feed Forward) – skládá se z plně propojených vrstev, kde každý neuron jedné vrstvy má svůj výstup propojen se vstupy všech neuronů následující vrstvy.
- Rekurentní neuronová síť RNN (Recurrent Neural Network) – jde o dopřednou síť, která má navíc propojení mezi neurony i v rámci jedné vrstvy, díky čemuž dokáže sledovat časovou korelaci.
- Automatický kodér AE (Auto Encoder) – obsahuje vrstvu s nižší propustností, kolem které jsou symetricky další vrstvy – vstupní část tak realizuje kompresi a výstupní část dekompresi.
- Konvoluční neuronová síť CNN (Convolutional Neural Network) – architektura hluboké neuronové sítě DNN (Deep Neural Network, neuronová síť s více než jednou skrytou vrstvou) obsahující konvoluční a pooling vrstvy – v konvoluční vrstvě je každý neuron svým výstupem propojen se vstupy sousedních neuronů, pooling vrstva zjednodušeně řečeno realizuje podvzorkování, díky čemuž dokáže CNN sledovat prostorovou korelaci.

Zmíněný seznam neuronových sítí obsahuje jen základní typy. V současné době existuje velké množství architektur a struktur se specifickými funkcemi a nové jsou stále vyvíjeny [43].

2.1.2 Strojové učení

Strojovým učením se rozumí natrénování systému, typicky složeného z jedné nebo více neuronových sítí, které má za následek změnu vnitřních stavů systému s cílem přizpůsobení se okolním podmínkám. Vnitřní stavy systému jsou v rámci neuronových sítí reprezentovány dříve zmíněnými váhami $w_0, w_1, ...$ Samotné trénování je založeno na dostupnosti velkého množství dat, přičemž nejčastějším algoritmem trénování neuronových sítí je algoritmus zpětné propagace chyby *backpropagation* (zkratka spojení backward propagation of errors). Ten spočívá v tom, že síti je ukázáno velké množství vstupů a odpovídajících očekávaných výstupů a síť si podle toho iterativně sama upravuje vnitřní stavy tak, aby bylo dosaženo minimální odchylky mezi skutečným a požadovaným výstupem, většinou ve smyslu střední kvadratické chyby MSE (Mean Squared Error). Neuronové sítě tak mají schopnost samoučení [13].

2.2 Aktuální stav využití NN a ML v kompresi obrazu

V této části se nachází přehled a popis několika článků, které se vyskytují v oblasti využití strojového učení v kompresi obrazu. Články jsou seřazeny chronologicky dle svého data publikace, přehledně je pak shrnuje Tabulka 2.1.

2.2.1 Historický pohled

Jiang, autor článku [23], provedl průzkum v této oblasti již v roce 1999. Přestože od té doby došlo k výraznému pokroku v rámci vývoje neuronových sítí a strojového učení, především díky vysokému nárůstu dostupného výpočetního výkonu, podává tento článek stále aktuální pohled na rozdělení problematiky. Neuronové sítě lze v rámci komprese využít dvěma způsoby:

1. Plná náhrada celého konvenčního kodeku, tedy NN přímo ke kompresi (často označované jako end-to-end metody).
2. Implementace již existujících kodeků nebo jejich částí, případně nepřímé použití NN k vylepšení těchto kodeků.

Toto rozdělení potvrzuje i článek [34] z roku 2020. Co se týče konkrétních neuronových sítí, zmiňuje článek [23] několik základních architektur v obou daných způsobech použití:

- NN přímo ke kompresi (všechny založeny na algoritmu backpropagation)
  - Základní NN – obsahuje vstupní vrstvu, výstupní vrstvu a skrytou vrstvu o menší dimenzi než vstupní vrstva.
  - Hierarchická NN – přidává další skryté vrstvy s cílem snížit korelaci mezi jednotlivými pixely a bloky.
  - Adaptivní NN – k dispozici je několik NN s různým počtem skrytých vrstev, přičemž před kompresí je obraz rozložen na několik oblastí, je určena složitost těchto oblastí a podle ní pak použitá příslušná NN.
- NN k implementaci již existujících kodeků nebo jejich částí
  - Vlnková transformace – jednotlivé neurony představují jednotlivé členy lineární kombinace dceřinných vlnek odvozených posunutím a měřítkem od mateřské vlnky, přičemž optimalizace se svou účelovou funkcí zaměřuje právě na posunutí, měřítko a váhy těchto vlnek.
  - Prediktivní kódování – NN vyvinuté k realizaci lineárního prediktivního kódování LPC (Linear Predictive Coding), pulzně kódové modulace PCM (Pulse Code Modulation), případně rozdílové pulzně kódové modulace DPCM (Differential Pulse Code Modulation).

- Vektorová kvantizace – NN vyvinuté ke konstrukci kódové knihy.

Tento dnes již přes 20 let starý článek navíc předpovídá, že k významnému pokroku a především standardizaci v této oblasti bude potřeba celosvětové spolupráce. To dokazuje dnešní aktivita skupiny JPEG [24].

Článek předpovídá také vývoj v této oblasti v podobě konstrukce neuronových sítí s více skrytými vrstvami, které by umožnily sofistikovanější učení, což opět dokazuje současný stav využití DNN [13].

2.2.2 Současný stav

V poslední době, myšleno v předchozích 5–10 letech, se většina přístupů držela spíše možnosti implementace a zdokonalování různých částí konvenčních kodeků pomocí neuronových sítí. Použití strojového učení v oblasti zpracování obrazu se vedle toho nejvíce soustředilo na aplikace rozpoznávání objektů, detekci vzorů a obecně strojového vidění (Machine Vision). Využití NN k přímé kompresi se začalo ve větší míře objevovat až v posledních 2 letech, i když se najdou i výjimky.

Rao se svým týmem studovali různé struktury ANN již v roce 2010 [61]. Porovnávali jednotlivé struktury mezi sebou a s konvenčními metodami, aby dosáhli co nejlepších výsledků. Jako základ použili ve svých pokusech vrstvu s nízkou propustností realizující kompresi a navíc přidali několik skrytých vrstev redukujících korelaci mezi pixely a bloky pixelů. Důležitým krokem před postoupením obrázků neuronové síti bylo jejich předzpracování, především převedení na 1D data. Proces realizovali na programovatelném hradlovém poli FPGA (Field Programmable Gate Array). Závěrem jejich výzkumu byl fakt, že ANN jsou slibnou vizí do budoucna, jelikož produkovali shodné nebo lepší výsledky než konvenční metody.

V roce 2014 se použitím strojového učení v kompresi obrazu zabývali Quijas a Fuentes [60]. Za cíl si stanovili odstranění blokové struktury, která vzniká při kompresi JPEG především při nižším nastavení bitového toku, tedy vyšší úrovně komprese. Použili dopřednou neuronovou síť k predikci DCT (Discrete Cosine Transform) koeficientů a aproximaci intenzit pixelů na okrajích bloků tak, aby došlo k potlačení blokové struktury a redukci šumu v obrázku při zachování detailů. Na predikci každého DCT koeficientu natrénovali jednu neuronovou síť, která predikuje DCT koeficient ze znalosti kvantovaného DCT koeficientu a jeho pozice v bloku. Použili i další neuronové sítě k predikci intenzity pixelu ze znalosti jeho okolí, přičemž uvažovali různě velká okolí. Nakonec aplikovali vyhlazení Gaussovským filtrem. Výsledkem jejich algoritmu bylo jednak potlačení blokové struktury vzniklé kompresí a jednak zvýšení PSNR (Peak Signal-to-Noise Ratio).

V roce 2016 vydali Ballé se svým týmem článek, ve kterém popsali kompletní kompresi založenou na neuronových sítích [3]. Navrhli architekturu skládající se z analyzující CNN, uniformního kvantizéru a syntetizující CNN. Pomocí stochastického gradientního algoritmu optimalizovali celý model pomocí trénovací databáze dat z hlediska R–D (Rate–Distortion) křivek. Svou

architekturou dokázali překonat standardy JPEG a JPEG 2000 z hlediska kompresní účinnosti i výsledné kvality.

V roce 2017 začal zájem o oblast využití strojového učení v kompresi obrazu výrazně růst, jelikož bylo vydáno hned několik článků z tohoto oboru a tento trend pokračuje dodnes.

Prakash a jeho tým se zabývali využitím konvolučních neuronových sítí v rámci komprese obrazu [57]. Využili CNN pro detekci obsahově významných částí obrazu ROI (Region of Interest), které následně komprimovali s vyšší nastavenou kvalitou než méně důležité části, například pozadí. Algoritmus tedy založili na ROI, nicméně nehledali jen jednu významnou část, ale rovnou několik v jednom snímku. Výsledkem byla MS-ROI (Multi-Structure ROI) mapa, kterou využili výše uvedeným způsobem v JPEG kodéru. Rozšířili tak standardní JPEG kodér, nepřidali do komprimovaného obrázku žádná aditivní data, a tak mohli k dekódování použít jakýkoliv JPEG dekodér. Výpočetní náročnost tohoto postupu nebyla nijak vysoká, jelikož model natrénovali dopředu.

Jiang se svým týmem se také zabývali kompresí s využitím konvolučních neuronových sítí [22]. Navrhli kompletní metodu komprese, ve které využili dvě CNN, především s cílem dosažení vysoké kvality pro nízké bitové toky. První CNN natrénovali k vytvoření kompaktní reprezentace vstupního obrázku se zachováním strukturální informace. Tuto kompaktní reprezentaci pak postoupili běžnému kodeku, například JPEG. Druhou CNN natrénovali k rekonstrukci obrazových dat. Pro nejlepší spolupráci obou konvolučních neuronových sítí navrhli algoritmus učení, ve kterém se učí obě sítě současně. Díky zmíněnému postupu dosáhli kompatibility této metody s existujícími kodeky. Výsledky navíc ukázaly výrazné zlepšení kvality výsledných snímků oproti konvenčním kodekům, a to především pro nízké bitové toky.

Další článek z roku 2017 vydali Matsuda se svým týmem [47]. Zabývali se vylepšením potlačení kompresních artefaktů při kódování zejména videa realizovaného pomocí filtru s adaptivním offsetem SAO (Sample Adaptive Offset), které je použito ve standardu HEVC (High Efficiency Video Coding) jako protiblokový filtr uvnitř smyčky. Ve standardu HEVC je offset vzorků nastavován podle určitých kategorií, do kterých spadají. Klasifikační algoritmy jsou založené na bázi blok–blok, což vyžaduje mnoho přenášené informace. Matsuda a tým představili klasifikaci na bázi strojového učení s využitím metody podpůrných vektorů SVM (Support Vector Machine). Touto metodou dosáhli úspory v množství přenášených dat, nicméně jen okolo 1 %, navíc musí autoři zapracovat na výrazném výpočetním zjednodušení.

Toderici se svým týmem se v roce 2017 zabývali metodami přímé ztrátové komprese založené na rekurentních neuronových sítích [67]. Představili několik metod, přičemž všechny architektury se skládaly z kodéru a dekodéru založených na RNN, binarizéru, jehož cílem bylo reprezentovat každou vstupní celočíselnou hodnotu pomocí sekvence bitů, a entropického kodéru založeného na neuronové síti. Jejich metoda překonala vůbec poprvé standard JPEG pro širokou škálu bitových toků.

V roce 2018 vydali Liu a Yang článek zabývající se kompresí barevných

obrázků pomocí strojového učení [44]. Vycházeli z predikce chrominančních kanálů C_B a C_R z jasového kanálu Y pomocí modelů založených na ML. Přišli s novým algoritmem a celým kodekem, který odstranil problémy předchozích kodeků s časovou náročností a nevyužitím všech informací o barvě. Pro snížení výpočetní složitosti použili metodu analýzy hlavních komponent PCA (Principle Component Analysis) a jako účelovou funkci zvolili minimalizaci celkové chyby predikce. Svým návrhem dosáhli pro kódování s nízkým bitovým tokem lepších výsledků než JPEG, nicméně oproti JPEG 2000 z hlediska R–D křivek (potřebného počtu bitů pro určitou kvalitu) neobstáli.

Další článek z roku 2018 vydali Liu se svým týmem [45]. Za cíl si stanovili návrh nového kodeku, který by byl pro kompresi obrázků dále postupovaným hluboké neuronové síti k rozpoznávání objektů vhodnější než JPEG. Zaměřili se na modelování odlišnosti mezi lidským vizuálním systémem HVS (Human Visual System) a způsobem vnímání hlubokých neuronových sítí. Na základě toho vyvinuli kodek DeepN-JPEG, který vycházel z klasického JPEG, ale speciálně ho upravili pro potřeby rozpoznávání objektů pomocí DNN. Úprava spočívala především ve změně kvantizační matice, což vedlo k vyššímu kompresnímu poměru při zachování stejné přesnosti rozpoznávání.

V roce 2018 představili konkrétní využití strojového učení v kompresi obrazu také Kouanou se svým týmem [39]. Zaměřili se na kompresi barevných obrázků z biomedicíny, přičemž strojové učení využili ke konstrukci kódové knihy k vektorové kvantizaci po ortogonální transformaci.

V následujícím roce, tedy v roce 2019, se Punnappurath a Brown zabývali využitím surových RAW dat vedle komprimovaných JPEG dat v rámci strojového vidění [58]. Navrhli metodu trénování DNN, která brala v potaz jak chyby v obrazové věrnosti (perceptuální kvalitě), tak chyby v rekonstrukci RAW dat, a to pomocí zanesení této druhé části do účelové funkce neuronové sítě. Svůj přístup ukázali na síti trénované od začátku pro oba cíle, ale i na síti, která byla původně trénovaná jen pro chyby v obrazové věrnosti a následně vylepšena o tuto funkci. Svým algoritmem dosáhli lepší RAW rekonstrukce s minimálním dopadem na obrazovou věrnost, kterou měřili pomocí metriky MS-SSIM (Multiscale Structural Similarity). Experiment vyzkoušeli na datech z DSLR (Digital Single-Lens Reflex) kamery i mobilního telefonu.

Dalšími, kdo se v roce 2019 věnovali kompresi s využitím NN a ML, byli Lee se svým týmem [40]. Zaměřili se na vylepšení entropického kódování, přičemž ho implementovali jako adaptivní. Co se týče samotné implementace kodeku, vycházeli z práce Ballého s týmem z roku 2017 a 2018 a do struktury zakomponovali svůj entropický model. Ten optimalizovali, tedy natrénovali, vždy pro určitý cílový bitový tok (celkem 9 úrovní) a pro dvě různé metriky popisující zkreslení (MSE a MS-SSIM). Celkem tedy vzniklo 18 různých modelů. Navržený kodek dokázal svými výsledky překonat konvenční kodeky a také dříve navržené kodeky na bázi NN a ML.

V témže roce se Dumas ve své disertační práci věnoval vylepšení současných součástí kodeku HEVC pomocí strojového učení [13]. Zaměřil se především na část transformace a vnitrosnímkové predikce, u které kvůli množství proměnných představil množství architektur neuronových sítí.

Trénováním neuronových sítí se zabývali i Pistono se svým týmem v roce 2020 [55]. Studovali, zda je možné postoupit ML algoritmu k natrénování JPEG obrázky ještě před dekompresí. Cílem bylo vynechat JPEG dekompresi kvůli zlepšení výpočetní doby. Pro své experimenty použili data z různých fází JPEG dekomprese a zkoumali dopady na přesnost vytvoření modelu. Zahrnuli i dopad různého nastavení faktoru kvality komprese JPEG. Zjistili, že opravdu lze výpočetní složitost snížit použitím jen částečně dekomprimovaných JPEG dat bez ztráty přesnosti modelu, avšak při použití CNN je třeba provést určitou adaptaci modelu.

Tabulka 2.1 přehledně rozděluje všechny zmíněné články do 2 skupin podle toho, zda se zabývají přímou kompresí pomocí neuronových sítí, nebo zdokonalováním již existujících kodeků nebo jejich součástí.

Tabulka 2.1: Rozdělení článků, které se zabývají využitím neuronových sítí a strojového učení v kompresi obrazu

Typ použití NN a ML	Články
Přímá komprese (end-to-end)	[61], [3], [22], [67], [24]
Zdokonalování konvenčních kodeků nebo jejich součástí	[60], [57], [47], [44], [45], [39], [58], [40], [13], [55]

V tabulce zmíněný článek [24] obsahuje první představení aktivity skupiny JPEG v oblasti komprese obrazu s využitím strojového učení, o které pojednává následující část.

2.2.3 JPEG AI

Článek [24] byl vydán v březnu 2019 na 83. setkání skupiny JPEG a bylo v něm oznámeno, že se skupina zabývá studiem metod komprese obrazu založených na strojovém učení s cílem vyvinout standard, který by tyto metody integroval. Za tímto účelem byla zahájena aktivita JPEG AI (JPEG Artificial Intelligence), jejímž cílem je vývoj kodeků založených na strojovém učení, které by dávaly lepší výsledky než konvenční metody komprese.

Na tento článek navazuje dokument [34] z dubna 2019, ve kterém je podán utříděný seznam relevantní literatury, dostupných softwarových implementací kompresních algoritmů na bázi strojového učení a dostupných databází obrazových dat vhodných k trénování a testování. Podobně jako ve zmíněném článku [23] z roku 1999 je zde rozlišeno, že některá řešení jsou kompletní kompresí založenou na strojovém učení, jiná se zaměřují na zlepšení již existujících kodeků a jejich částí, například nelineární transformace, kvantizace, nebo entropického kódování. V dokumentu jsou také shrnuty hlavní znaky, podle kterých je možné a také vhodné metody rozlišit a klasifikovat:

- typ neuronové sítě – například použité vrstvy, ale také, zda použít více NN najednou, každou pro jinou část kodeku,

- velikost kódové jednotky – NN jsou obvykle uzpůsobeny pro konkrétní velikost kódové jednotky – pokud snímek není násobkem této velikosti, je třeba ho tomu přizpůsobit,
- nástroje k vytvoření kompaktnější reprezentace vzhledem k prostorové korelaci,
- volba fixního nebo adaptivního bitového toku,
- možnost nastavení kvality a dosažení vysoké kvality pro nízký bitový tok,
- efektivita kódování a výpočetní náročnost.

V prosinci 2019 byl vydán dokument [31] navazující na 84. a prezentovaný na 85. setkání skupiny JPEG, ve kterém jsou shrnuty výsledky experimentálního vyhodnocení kompresní účinnosti 5 kodeků založených na ML proti 4 kodekům založeným na transformaci, konkrétně HEVC, WebP, JPEG 2000 a JPEG. Porovnání bylo provedeno na 8 snímcích v rozlišení od SD (Standard Definition), přes FHD (Full High Definition) až po UHD (Ultra High Definition), a to vždy pro 4 různé bitové toky. Podobně jako v článku [61] bylo zjištěno, že kodeky založené na ML jsou konkurenceschopné. Navíc byly objeveny i specifické artefakty spojené s kodeky založenými na ML. Kromě subjektivního hodnocení bylo provedeno i hodnocení pomocí objektivních metrik, aby se zjistilo, která z nich dobře koreluje se subjektivními testy s ohledem na specifické artefakty způsobené ML kompresí. Pro testování byly zvoleny metriky PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity), MS-SSIM (Multiscale SSIM), VIF (Visual Information Fidelity), VMAF (Video Multimethod Assessment Fusion) a další. Výsledky byly určeny hodnotami Pearsonova a Spearmanova korelačního koeficientu, podle kterých pro kodeky na bázi ML korelují objektivní a subjektivní testy mnohem méně než pro současné kodeky, nicméně nejvyšší vypovídající hodnotu má jako u současných kodeků metrika MS-SSIM.

K významnému pokroku došlo také v únoru 2020, kdy se konalo 86. setkání skupiny JPEG [25], na kterém byl vydán dokument [28] reprezentující první formální krok ke standardizaci kodeků na bázi strojového učení. Společně s tím byla od února do září 2020 pořádána JPEG AI MMSP Challenge [35]. Cílem [28] i [35] bylo nalezení konkrétních vhodných řešení komprese obrazu s využitím strojového učení, na kterých by se dal postavit další vývoj. Požadavkem byl podrobný popis kódovacího algoritmu, implementace dekodéru, která by umožnila testování na běžném počítači v rozumném čase, a komprimované datové toky a odpovídající dekódované obrázky. Pro trénování a vyhodnocení výkonnosti byla skupinou JPEG poskytnuta databáze testovacích snímků JPEG AI Datasets obsahující rozmanité snímky z hlediska různých vlastností, například rozlišení nebo barevnosti. Dále byly podrobně popsány metodologie vyhodnocování návrhů a použité metriky, přičemž k tomuto byl v srpnu 2020 po 88. setkání skupiny JPEG vydán speciální dokument [32]. Ten obsahuje popis metodologie subjektivního hodnocení předložených řešení (řešení předložilo 6 týmů – 4 týmy jako odpověď na [28] a 2 týmy na [35]), přičemž

kvůli pandemii Covid-19 bylo hodnocení provedeno způsobem crowdsourcing. Prezentace a diskuze nad návrhy proběhla na 89. setkání skupiny JPEG [26]. Bylo dokázáno, že není pochyb o tom, že kodeky na bázi neuronových sítí dokáží překonat konvenční kodeky z hlediska kompresní účinnosti. Tato konference dala také mimo jiné vzniknout dvěma dokumentům – [33] a [29].

Dokument [33] obsahuje výsledky objektivního a subjektivního hodnocení kvality všech 6 předložených řešení, jejich vzájemné porovnání, porovnání s konvenčními kodeky a také vyhodnocení výpočetní náročnosti. Subjektivní hodnocení bylo provedeno přesně podle výše zmíněného a dopředu známého postupu. Dokument [29] navíc poprvé popisuje, jak by měly být experimenty s kodeky založenými na neuronových sítích a strojovém učení v rámci JPEG AI prováděny a vyhodnocovány. Definuje:

- testovací databáze obrazových dat,
- konvenční kodeky, se kterými by mělo být prováděno porovnání,
- bitové toky, kterých by mělo být dosaženo a na kterých by měly být kodeky porovnávány,
- metriky objektivního hodnocení kvality,
- postup subjektivního hodnocení kvality.

Kapitola 3

Dostupné softwarové implementace a nástroje

Vývoj metod v oblasti strojového učení může být velmi komplexní proces spjatý s vyšší složitostí a finanční a časovou náročností. Zásadní je v této oblasti volba použitých nástrojů. Problémem může být i neexistence nástrojů, které by pokryly celý proces vývoje, a tak je často potřeba spojit a využít několik nástrojů dohromady, což může být časově náročné a může vést k chybám.

Tato kapitola podává stručný přehled o Cloud Computing řešení, dále pak porovnává dostupné softwarové implementace a nástroje, které umožňují kompresi obrazu založenou na učení, s důrazem na možné experimentální využití.

3.1 Cloud Computing

Díky velkým společnostem jako je Amazon, Google nebo Microsoft jsou služby strojového učení také nabízeny v rámci Cloud Computing, tedy výpočetního výkonu dostupného ve virtuálním cloudu. Uživatelé těchto systémů tak nemusí být přímo vybaveni vysokým výpočetním výkonem místní stanice, ale mohou využít výpočetní výkon poskytovaný servery těchto společností. V rámci Cloud Computing ovšem nejde pouze o poskytnutí výpočetního výkonu, ale i různých služeb, jejich částí a nástrojů připravených pro vývojáře. Zásadní výhodou tohoto řešení je přenechání veškeré správy na straně poskytovatele. Uživatel se nemusí starat o hardware, který výkon zprostředkuje, nemusí řešit aktualizace softwaru, jelikož vždy dostane nejnovější verzi, přístup ke službám má odkudkoliv, kde je k dispozici internetové připojení, a zabezpečení je na velmi vysoké úrovni [8].

Nabízených služeb a nástrojů je opravdu velké množství a jsou k dispozici pro různá odvětví včetně využití umělé inteligence. Při vztažení na problematiku komprese obrazu se nabízí uplatnění takových služeb především pro proces učení neuronových sítí, který je časově i výpočetně velmi náročný, případně pro proces testování, pokud je vyžadováno provedení velkého množství kompresí a dekompresí, které by na běžném hardwaru zabraly příliš mnoho času. Každý z poskytovatelů nabízí určitý model předplatného, kdy si

uživatel kupuje výpočetní prostředky na určitou dobu, ale často jsou dostupné i bezplatné možnosti zahrnující buď nějaké zkušební období, nebo například omezení dostupného výpočetního výkonu.

Tabulka 3.1 ukazuje přehled čtyř největších poskytovatelů služeb na trhu Cloud Computing a jejich podíl na tomto trhu v říjnu 2020 podle [52].

Tabulka 3.1: Přehled poskytovatelů Cloud Computing služeb

Poskytovatel	Název služby	Podíl na trhu
Amazon	Amazon Web Services	32 %
Microsoft	Microsoft Azure	19 %
Google	Google Cloud Platform	7 %
Alibaba Group	Alibaba Cloud	6 %

Do první desítky, která dohromady pokrývá 80 % Cloud Computing trhu, patří dále IBM, Salesforce, Tencent, Oracle, NTT a SAP. Zbylých 20 % zastávají menší poskytovatelé, mezi které patří například FloydHub nebo Paperspace se svou službou Gradient.

3.2 Nástroje hlubokého učení

V této části jsou shrnuty a srovnány dostupné nástroje hlubokého učení, výhody a nevýhody jednotlivých řešení a možnosti jejich použití společně s jinými nástroji tak, aby byl vytvořen celkově příznivý pracovní postup během celého procesu vývoje.

3.2.1 TensorFlow

TensorFlow je otevřená knihovna pro vývoj a trénování ML modelů vyvinutá společností Google, která obsahuje množství nástrojů. Je možné ji integrovat v rámci různých prostředí (v cloudu, prohlížeči, přímo na zařízení), různých programovacích jazyků a operačních systémů. Nabízí mnoho již připravených scénářů a návodů, které řeší základní ML úlohy, ale i pokročilejší problémy. Navíc za touto knihovnou stojí silná komunita v podobě vědců a vývojářů, kteří často dávají svá řešení určitých problémů volně k dispozici, což usnadňuje a zefektivňuje práci ostatním. TensorFlow se prezentuje jako ekosystém nabízející kompletní řešení od návrhu modelu, přes jeho trénování, testování a zdokonalování po nasazení k řešení reálného problému [64].

TensorFlow je nejčastěji používána v rámci jazyka Python a obecně patří tato kombinace k pravděpodobně nejpoužívanějším nástrojům v rámci hlubokého učení. V rámci Pythonu je vhodné využívat knihovnu `matplotlib`, která nabízí kvalitní vykreslování grafů podobné tomu v programu MATLAB, a knihovnu `numpy` zajišťující standardizovanou reprezentaci dat. Obecně je možné toto řešení kombinovat s dalšími knihovnami, které uživatel používá, a vytvořit tak velmi příjemné prostředí pro práci se strojovým učením. Mezi základní výhody tedy patří otevřenost celého řešení bez nutnosti placení poplatků

nebo licencí, univerzálnost použití na různých platformách a velká komunita. Toto řešení je v současné době jedno z nejvíce doporučovaných [70].

3.2.2 Caffe

Dalším poměrně používaným frameworkem hlubokého učení je Caffe, který původně vyvinul Yangqing Jia během svého doktorského studia na Kalifornské univerzitě v Berkeley. Mezi zásadní vlastnosti patří podpora různých architektur hlubokého učení, jednoduché přepínání mezi použití CPU (Central Processing Unit) nebo GPU (Graphics Processing Unit), vysoká rychlost a rozvíjející se komunita. Vývoj nakonec přešel pod společnost Facebook, která v roce 2017 představila Caffe2 s novými funkcemi mířícími především na AI v mobilních zařízeních. V roce 2018 byl ale framework Caffe2 integrován do frameworku PyTorch, a tak není samostatně dál podporován [6, 7].

3.2.3 PyTorch

Mezi další velké frameworky hlubokého učení vedle TensorFlow patří PyTorch vyvíjený společností Facebook. Jedná se o otevřenou knihovnu využívanou především pro aplikace strojového vidění, zpracování obrazu a řeči. Doporučeným programovacím jazykem je Python, ale PyTorch umožňuje také integraci při použití C++. Je také podporován všemi čtyřmi největšími poskytovateli Cloud Computing služeb zmíněnými výše, což může výrazně zjednodušit vývoj. V listopadu 2020 PyTorch přidal možnost spouštět modely přímo na mobilních zařízeních vybavených systémem na čipu SoC (System on a Chip) s operačním systémem Android a iOS. Podobně jako TensorFlow je možno PyTorch kombinovat s množstvím dalších knihoven, ale i tento samotný framework obsahuje moduly usnadňující mnohé úkony [59].

3.2.4 ONNX

ONNX (Open Neural Network Exchange) není dalším frameworkem hlubokého učení k trénování a vývoji ML modelů, ale jedná se o systém pro vzájemnou výměnu modelů mezi různými frameworky a zajištění jejich kompatibility. Systém je vyvíjen společnostmi Facebook a Microsoft a v současné době má podporu několika dalších společností. Jedná se tedy o formát reprezentace ML modelů, účelem je vzájemné předávání modelů mezi různými frameworky, přičemž se začínalo s podporou převodu mezi Caffe2 a PyTorch. Toto umožňuje použití různých frameworků v různých fázích vývoje tak, aby bylo dosaženo maximální efektivity, tedy například jeden framework může umožňovat velmi rychlé trénování, jiný zase integraci na mobilní platformy. Export do ONNX nebo import z ONNX v současnosti podporuje hned několik nástrojů včetně všech výše zmíněných [50, 51].

3.2.5 Deep Learning Toolbox

Velkou konkurencí pro Python a knihovnu TensorFlow nebo PyTorch je program MATLAB vyvíjený společností MathWorks se svým balíčkem Deep Learning Toolbox, který se dříve jmenoval Neural Network Toolbox. Jedná se také o celou knihovnu pro vývoj a implementaci hlubokých neuronových sítí a modelů strojového učení, přičemž navíc nabízí aplikaci umožňující vše provádět graficky. Balíček umožňuje využití mnoha předtrénovaných modelů nebo importování modelů z TensorFlow nebo ONNX. Pro urychlení trénování podporuje v základu výpočty pomocí grafické karty, při současném využití balíčku Parallel Computing Toolbox dokonce pomocí více grafických karet. Jistou nevýhodou ovšem je, že není zatím možné pro toto řešení využít Cloud Computing služby, a tak je nutné mít pro efektivní práci poměrně vysoký výkon místní stanice. Zásadním rozdílem také je, že se nejedná o otevřený balíček, ale je dostupný v rámci placené licence programu MATLAB [11].

Tento balíček je také prezentován jako kompletní řešení, které nabízí vše od přípravy, předzpracování, nebo simulace dat, přes návrh a trénování modelu, jeho integraci do složitějších systémů, simulaci a ověření funkčnosti celého systému až po jeho finální uvolnění v rámci specializovaných zařízení, podnikových systémů, běžných koncových zařízení nebo cloudových služeb. Obsahuje nástroje pro vývoj aplikací v několika specializovaných oblastech, například zpracování signálů, automatické řízení, zpracování obrazů, strojové vidění, nebo zpracování audia. Nástroje jsou pro tyto oblasti speciálně navržené, aby v daném scénáři fungovaly co nejlépe. Variabilitu z hlediska použitelného finálního zařízení, na kterém aplikace poběží, zajišťuje automatický generátor kódu. Ten dokáže generovat optimalizovaný kód pro různé platformy, jako jsou procesory Intel nebo ARM, programovatelná pole FPGA, systémy na čipu SoC, nebo grafické procesory NVIDIA. Stejně jako má TensorFlow, PyTorch a Python širokou komunitu, nabízí Deep Learning Toolbox kromě své vlastní komunity také podporu od inženýrů společnosti MathWorks. Svou sílu a spokojenost mezi uživateli ukazuje tento balíček i díky ocenění 2020 Gartner Peer Insights Customers' Choice for Data Science and Machine Learning Platforms, které získala společnost MathWorks v červenci 2020 a které znamená nejlepší volbu mezi zákazníky na poli platforem zabývajících se strojovým učením [16, 71].

3.2.6 Shrnutí

Z výše zmíněných řešení vyvstává především otázka, zda použít otevřené knihovny TensorFlow nebo PyTorch, nebo placené řešení v podobě Deep Learning Toolbox.

Obecně platí, že MATLAB nabízí výbornou práci s daty včetně jejich efektivního zpracování nebo zobrazení a prezentace, zatímco otevřená řešení toto v základu většinou nenabízí a je potřeba k tomu použít další knihovny nebo jiné nástroje, což může být uživatelsky méně přívětivé, na druhou stranu může každý používat to, na co je zvyklý. Druhou zásadní oblastí je podpora, kdy na straně otevřených řešení je většinou silná komunita a na straně

placených zase silná základna vývojářů, jejichž podporu si člověk kupuje se softwarem. Poslední důležitou otázkou jsou možnosti jednotlivých řešení. Otevřené platformy nabízí prostor pro experimentování a výzkum a jsou tak většinou vhodnější volbou pro testování zcela nových přístupů a řešení, nebo řešení specifických problémů a úloh. Komerční software většinou na tyto nové přístupy reaguje a jeho vývojáři je do něj implementují, avšak toto nutně vede ke zpoždění, a tak je toto řešení vhodné spíše pro používání již ověřených a zavedených postupů.

Tabulka 3.2 shrnuje základní použitelnost výše zmíněných nástrojů hlubokého učení – data byla převzata ze zdroje [9], kde je zároveň k dispozici poměrně rozsáhlé srovnání softwaru hlubokého učení. Všechny čtyři nástroje zmíněné v tabulce navíc podporují využití GPU a paralelizace, obsahují předtrénované modely a podporují mimo jiné RNN a CNN.

Tabulka 3.2: Přehled základních nástrojů hlubokého učení [9]

Software	Otevřený	Platforma	Rozhraní
Tensorflow	Ano	Linux, macOS, Windows, Android	Python, C, C++, Java, Go, JavaScript, R, Julia, Swift
Caffe	Ano	Linux, macOS, Windows	Python, Matlab, C++
PyTorch	Ano	Linux, macOS, Windows	Python, C++, Julia
Deep Learning Toolbox	Ne	Linux, macOS, Windows	Matlab

3.3 Implementace diskutované v rámci JPEG AI

Ve výše zmíněném dokumentu [34] byl v rámci aktivity JPEG AI sestaven seznam několika dostupných implementací týkajících se využití neuronových sítí v kompresi obrazu. Seznam obsahuje implementace realizující kompletní kompresi, ale i implementace zaměřující se na vylepšování současných kodeků nebo jejich částí. V kapitole 5 Porovnání metod komprese obrazu založených na učení a konvenčních kodeků je proveden experiment porovnávající některé z daných implementací mezi sebou, ale také s konvenčními kodeky, přičemž experiment je zaměřen pouze na implementace realizující kompletní (end-to-end) kompresi. Dvě z těchto metod byly vyzkoušeny a porovnány s konvenčními kodeky v článku [31].

Kapitola 4

Latentní reprezentace obrazových dat

V návaznosti na výzkum komprese založené na učení se několik výzkumných skupin začalo zabývat zkoumáním možností využití latentní reprezentace obrazových dat, tedy dat v transformační doméně, která jsou výstupem kodéru založeného na učení realizovaného pomocí NN. Mezi další možná označení patří termíny latentní kód nebo data v latentním prostoru nebo doméně. Obrázek 4.1 velmi zjednodušeně znázorňuje, o jaká data jde.

Obrázek 4.1: Grafické znázornění latentního kódu

Latentní reprezentace v sobě obsahuje všechna důležitá data potřebná k rekonstrukci více či méně zkreslené kopie původního snímku, ze kterého vznikla. Objevuje se tak otázka, zda se nejedná o doménu vhodnou k detekci příznaků, vzorů a struktur, které snímek obsahuje, a vzniká tak nový pohled například na problematiku klasifikace nebo segmentace snímků. Jedná se o transformační doménu, ve které by člověk hledal strukturní informaci jen ztěžka. Latentní prostor je lidem primárně skryt (slovo latentní přímo znamená skrytý nebo utajený), avšak pro NN může být vhodnějším kandidátem k detekci této informace než klasický prostor 2D snímku. Provádění takového zpracování v latentní doméně navíc výrazně snižuje množství potřebných přenášených dat oproti použití originálního snímku a zjednodušuje celý proces, především z hlediska časové náročnosti, díky vynechání dekódování komprimované informace na rekonstruovaný snímek. Podobná myšlenka byla zmíněna při popisu článku [55] v části 2.2.2 Současný stav, ve kterém bylo zkoumáno, zda je možné postoupit ML algoritmu k natrénování NN JPEG obrázky ještě před dekompresí kvůli zlepšení výpočetní doby. Ve zmíněném případě šlo o použití dat v komprimované doméně pro běžný kodek, v tomto případě jde o data v komprimované doméně pro kodek založený na učení.

4.1 Aktuální stav využití latentní reprezentace

V této části je zmíněno a popsáno několik článků, které se zabývají využitím latentní reprezentace obrazových dat. Články jsou podobně jako v části 2.2 Aktuální stav využití NN a ML v kompresi obrazu seřazeny chronologicky dle svého data publikace, přehledně je pak shrnuje Tabulka 4.1.

4.1.1 Historický pohled

Úvahy ubírající se směrem extrakce příznaků z komprimované domény především za účelem klasifikace se objevily již v 90. letech 20. století, tehdy šlo ale o konvenční kodeky. V roce 1993 Gray a Oehler popsali využití vektorové kvantizace zároveň pro kompresi a klasifikaci tak, aby komprimovaný datový tok obsahoval informace ke klasifikaci bez nutnosti dalšího zpracování [49]. V roce 1997 Vasconcelos a Lippman přišli s přístupem využití klasického kodeku JPEG a jeho rozšíření o analýzu obsahu založenou na odvození příznaků z transformačních koeficientů [68]. Dalšími, kdo se zabývali extrakcí příznaků z komprimované domény, byli Keaton a Goodman v roce 1999, kteří využívali statistiky prvního a druhého řádu aplikované na transformační koeficienty a porovnávané pomocí vzdálenosti s různými třídami v transformační doméně [36].

4.1.2 Současný stav

Články postihující využití latentní reprezentace se začaly objevovat až v posledních několika letech, většina z nich dokonce až v roce 2020. Mnoho článků z této oblasti se zabývá klasifikací snímků s využitím NN, často hledají cestu, jak tento proces automatizovat, jen některé ale využívají i latentní informaci. Několik článků se také zabývá problematikou vytvoření vhodné latentní reprezentace, ve které by šlo dobře rozpoznat a extrahovat nějaké příznaky. Objevují se také články zabývající se redukcí šumu pomocí NN, často ale bez použití latentní informace v klasické doméně 2D snímku. Vedle toho existuje i několik článků zabývajících se kompresí speciálních dat, typicky point cloud (mračno bodů), hyperspektrální nebo multi-view (vícepohledová) data.

Problematikou extrakce příznaků za účelem klasifikace nebo vyhledávání se zabývají také hashovací metody, jejichž cílem je převod vstupních dat do velmi malého objemu kompaktních binárních dat (hashe). V roce 2016 vydali Zhu se svým týmem článek představující strukturu založenou na DNN, která umožňuje současné nalezení kompaktní reprezentace vstupního obrázku a hashe [81]. Díky současnému hledání obou reprezentací, tedy využití znalostí o kompaktní reprezentaci k určení podoby hashe, dokázali autoři svou navrženou strukturou překonat konvenční hashovací metody.

V roce 2017 publikovali Zhang se svým týmem dnes již velmi citovaný článek, ve kterém představili model DnCNN (Denoising CNN), který umožňuje provádět několik základních úloh zpracování obrazu, například redukci šumu, redukci blokových artefaktů vzniklých kompresí JPEG, nebo úlohu

super-resolution, s využitím architektury CNN [76]. CNN využili právě z důvodu, že je díky své hluboké architektuře vhodná k extrakci charakteristik obrazu, byly pro ni nalezeny metody urychlující trénovací fázi a je vhodná pro paralelizaci na moderních GPU. Na rozdíl od ostatních metod redukce šumu založených na NN, které přímo odhadují čistý snímek bez šumu, navrhli model DnCNN k odhadu rezidua mezi zašumělým vstupem a čistým snímkem, čistý snímek tedy od pozorování odečítali. Autoři také představili jednoduché modifikace umožňující použití modelu na všechny výše zmíněné úlohy a provedli poměrně rozsáhlé testy, ve kterých porovnali účinnost svého modelu s mnoha dalšími technikami pro různé úlohy, přičemž ve všech dosáhli výborných výsledků, navíc časová náročnost byla díky využití moderní GPU velmi přijatelná.

Ve stejném roce vydali Geng se svým týmem článek zabývající se klasifikací dat z radaru se syntetickou aperturou SAR (Synthetic Aperture Radar), který se používá k dálkovému průzkumu Země a mapování [17]. Představili metodu založenou na RNN, která dokáže automaticky ze snímků z radaru provést extrakci příznaků a na jejich základě se naučit prostorové korelace mezi částmi snímků a provést klasifikaci. Touto metodou dosáhli zvýšení přesnosti klasifikace snímků z SAR.

V roce 2018 vydali Pantraki a Kotropoulos článek, ve kterém se zabývali generováním starší a mladší podoby člověka založeném na snímcích obličeje [53]. Jde o problém přetvoření snímku z jedné věkové třídy na snímek z jiné věkové třídy, přičemž uplatnění najde například v úloze rozpoznání obličeje nezávisle na věku. Jako mezistupeň použili právě latentní doménu, do které snímek převedli, v ní poté provedli změny určitých příznaků podle cílové třídy a následně data opět převedli na snímek. Důležitým faktem je, že změnou prošly jen příznaky způsobující vizuální změnu věku, přičemž osobní rysy byly zachovány, díky čemuž byly výsledky velmi kvalitní.

Podobný článek vydali v roce 2019 Lee a Islam [41]. Zabývali se převodem snímku z jedné domény do jiné, při kterém dojde například k barevným korekcím, redukci šumu, doplnění neúplného snímku, nebo převodu do jiné třídy objektů a jiným změnám, a to s použitím DNN. Představili architekturu dvojitého automatického kodéru s oboustrannou regresní sítí, která propojila latentní prostory obou kodérů, které nejprve trénovali každý zvlášť pro svou latentní doménu, a následně trénovali regresní síť, která se tak naučila vztahy mezi oběma latentními prostory. Tímto způsobem dosáhli vysoké robustnosti vůči nekompletním nebo poškozeným snímkům při zachování vysoké přesnosti.

Odlišnou záležitostí se ve stejném roce zabývali Khan a Lau, kteří vydali článek zkoumající vlivy komunikačního kanálu na přenášenou latentní reprezentaci snímků [37]. Analyzovali rekonstruované snímky a schopnost jejich klasifikace pro různé úrovně šumu v kanálu, velikosti latentních vektorů a počtu kvantizačních bitů použitých pro latentní proměnné. Výsledkem jejich výzkumu byla vysoká citlivost latentní reprezentace na šum v přenosovém kanálu a kvantizační jevy, proto představili strukturu kombinující automatický kodér generující latentní reprezentaci a automatický kodér provádějící odšumování v přijímači, čímž výrazně vylepšili kvalitu rekonstrukce a klasifikace

snímků a kompenzovali tak zmíněnou citlivost latentní reprezentace.

Další článek zabývající se využitím latentní reprezentace vydali Ding se svým týmem taktéž v roce 2019 [12]. Představili metodu extrakce příznaků z latentní reprezentace, která byla naučena a ve které byla učením změřena důležitost různých příznaků. Ukázali, že extrakce příznaků z latentního prostoru je méně nachylná k chybám vzniklým šumem než extrakce z prostoru 2D snímku, nebo dokonce z vícerozměrného prostoru dat, navíc mnohem lépe postihuje spojitosti a podobnosti mezi daty. Svým přístupem dosáhli v několika testech výborných výsledků.

Ve stejném roce se Yao se svým týmem zabývali detekcí objektů ve snímcích z dálkového snímání ve vysokém rozlišení [72]. Tato detekce je velmi závislá na schopnosti extrahovat ze snímků vhodné příznaky, k čemuž použili architekturu CNN. Představili model, který se učí podobu objektů v latentní reprezentaci snímků, navíc do objektivní funkce přidali regularizační člen zajišťující učení se podoby objektu bez ohledu na jeho rotaci, respektive naučení se jeho podoby pro všechny jeho možné rotace. Díky tomu model dokáže detekovat objekty s vysokou přesností a efektivitou.

Stejně jako výše zmíněný článek [41] se i článek [46], který publikovali López se svým týmem v roce 2019, zabývá převodem snímku z jedné domény do jiné s využitím NN a použitím latentní informace. Autoři použili generativní kompetitivní sítě GAN (Generative Adversarial Network) skládající se ze dvou CNN, přičemž vytvořili architekturu dvojité cyklické GAN převyšující běžné přístupy z hlediska kvality, ale vyžadující složitější metody implementace a vyšší výpočetní kapacitu a čas konvergence.

Ke konci roku 2019 vydali Kharote se svým týmem článek zabývající se segmentací snímků prostaty z magnetické rezonance s cílem správné detekce rakoviny prostaty [38]. K segmentaci využili latentní informaci získanou ze snímků pomocí technik hlubokého učení s využitím automatických kodérů. Ukázali, že tento přístup se dokáže lépe vypořádat s odlišnostmi tvaru a velikosti napříč pacienty, díky čemuž dosáhli vysoké přesnosti segmentace a detekce nádorů.

Za zmínku rovněž stojí článek, který vydali na přelomu roku 2019 a 2020 Tian se svým týmem [66]. Podali v něm rozsáhlý přehled technik hlubokého učení pro redukci šumu – zaměřili se nejprve na využití CNN v různých aplikacích redukce šumu a následně i na další techniky. Kromě samotného přehledu provedli i rozsáhlé porovnání mnoha metod, v žádné části ale přímo nezmínili využití latentní reprezentace v úloze redukce šumu.

Odlišnou oblastí se zabývali Zhou se svým týmem, kteří v roce 2020 vydali článek zabývající se zarovnáním, nebo také sesouhlasením, či registrací snímků s využitím latentní informace [80]. Správné sesouhlasení je kritické například v oboru astronomické fotografie, nicméně autoři tohoto článku se zaměřili na snímky scén fotografovaných ze silnic, typicky si lze představit fotografie Google Street View. Představili metodu, která dokáže správně zarovnat snímky téhož místa focené za různých klimatických podmínek, tedy například letní snímek a zimní snímek se sněhem.

Další článek zmiňující latentní reprezentaci publikovali v roce 2020 Zhang

se svým týmem [77]. Představili metodu, která najde vhodný latentní prostor, ve kterém extrahuje příznaky a rozpozná a klasifikuje objekty. Nalezený latentní prostor je vhodný tím, že v sobě zahrnuje různé zdroje příznaků, a postihuje tak hlubší souvislosti. Toho autoři dosáhli použitím širších NN, namísto běžného přístupu zvyšování počtu vrstev, přičemž každá část NN se zaměřila na jiný zdroj příznaků a výsledný latentní supervektor pak byl spojením všech výstupů. Tento supervektor následně podstoupil transformaci do zobecněného latentního prostoru podle algoritmu, který autoři v článku představili. Autoři dosáhli tímto postupem výborných výsledků v rámci klasifikace.

Latentní reprezentaci použili také Gu se svým týmem, kteří v roce 2020 publikovali článek zabývající se odhadem 3D polohy ruky z 2D snímku [19]. Výzkum této oblasti má smysl z hlediska použití v rozšířené a virtuální realitě, nebo u systémů ovládání gesty. Představili strukturu skládající se ze dvou automatických kodérů, jeden pro prostor 2D snímku a jeden pro prostor 3D polohy, a bloků provádějících porovnání a převod mezi latentními prostory 2D snímku a 3D polohy.

Mishra se svým týmem se ve stejném roce zabývali problematikou clusterování, tedy rozdělování dat do určitých shluků, v latentním prostoru [48]. Zaměřili se také na vztah clusterů a daného latentního prostoru a definovali podmínky nutné k provedení clusterování v latentním prostoru. Také navrhli metodu konstrukce vhodného latentního prostoru.

V roce 2020 dále Suo se svým týmem publikovali článek zabývající se využitím latentní reprezentace point cloud dat [63]. Vhodná reprezentace je pro point cloud data velmi důležitá, jelikož bez úprav se jedná o velké datové objemy, navíc tato reprezentace může usnadnit některé náročné úlohy, například rozpoznávání nebo rekonstrukci nějakého místa. Autoři představili architekturu DNN, která dokázala v části kodéru z point cloud dat extrahovat globální i lokální příznaky a vztahy mezi nimi. Tímto způsobem dokázali vytvořit vhodnou latentní reprezentaci vstupních dat, ze které pak v části dekodéru rekonstruovali point cloud data. Ukázali tak, že mnoho výpočetně náročných úloh lze především s výhodou menší paměťové náročnosti provádět v latentním prostoru, a svým postupem dosáhli velmi dobrých výsledků v porovnání s konvenčními metodami užívanými v souvislosti s point cloud daty. Toto řešení má tak potenciál se dostat například do oblasti autonomních vozidel.

Další zajímavý článek v roce 2020 publikovali Zhang se svým týmem, kteří se zabývali využitím latentní reprezentace v úlohách rekonstrukce, nebo také obnovení, či restaurace snímků [78]. Na rozdíl od dřívějších přístupů používajících NN pro tyto úlohy bez uvažování latentní reprezentace zde autoři uvažovali latentní blok uvnitř NN, který dokázal lépe využít globální a lokální příznaky, které síť extrahovala. Jejich architektura obsahovala 2 větve – hlavní a latentní – hlavní extrahovala lokální příznaky a latentní zachovala globální informaci, díky čemuž dokázali tyto informace lépe použít. Svou architekturu a její efektivitu autoři demonstrovali na úlohách super-resolution, redukce šumu a redukce artefaktů vzniklých kompresí, přičemž ve všech úlohách

dosáhli lepších výsledků než konvenční metody.

Využitím latentní reprezentace v další zajímavé oblasti se zabývali Zhao se svým týmem, kteří zkoumali její použití v rámci klasifikace hyperspektrálních dat [79]. Navrhli metodu naučení se vhodného latentního prostoru, který obsahoval informace o několika typech příznaků, které hyperspektrální data poskytla a které maximálně využívaly spektrální a prostorovou informaci obsaženou v hyperspektrálních snímcích. Díky tomu navržená metoda dosáhla vysoké efektivity klasifikace, speciálně v případě malého množství trénovacích dat.

Cristovao se svým týmem se v roce 2020 zabývali využitím latentní reprezentace při interpolaci snímků ve videosekvenci, tedy dopočítávání nových snímků mezi již existujícími snímky [10]. Chtěli tak dosáhnout lepších výsledků než současné metody založené na posunu pixelů, které nemají informaci o struktuře celých dat, a proto nedokáží v určitých případech provést dobrou interpolaci. Navrhli architekturu s automatickým kodérem, který zakódoval prostorovou a časovou informaci do latentní reprezentace a nespoléhal na posuny pixelů, díky čemuž dokázal dobře vytvářet interpolované snímky například i s objekty, které byly částečně zakryté.

Další článek zabývající se point cloud daty v souvislosti s latentní reprezentací publikovali Zhang se svým týmem [75]. Zabývali se filtrací point cloud dat od šumu s cílem zachování ostrosti dat a bez nutnosti ladění mnoha parametrů. Oběma těmito problémy trpěly dřívější metody filtrace point cloud dat, a tak autoři navrhli metodu založenou na hlubokém učení, která vstupní data převedla do latentního prostoru, ze kterého pak dokázala vygenerovat čistá point cloud data bez šumu. Toho autoři docílili díky odhadu vektoru posunutí každého bodu a jeho navrácení do původní pozice. V porovnání s konvenčními metodami dosáhli lepších výsledků, avšak metoda má i některé nevýhody, mezi které patří problémy zachování ostrosti při výskytu vysoké hladiny šumu, nebo problémy s velkými dírami v datech – autoři by rádi do budoucna přidali informaci o celkovém tvaru objektu, čímž by problém mohli vyřešit.

Latentní prostor zmiňují i Aribido se svým týmem, kteří publikovali článek zabývající se automatickým anotováním seismických snímků s využitím DNN pomocí projekce latentního prostoru do naučených podprostorů [2]. Výsledkem jejich práce bylo automatické učení se nalezení geologických bodů zájmů ze získaných dat.

Fang a Zeng představili koncem roku 2020 novou metodu redukce šumu ve snímku s využitím CNN, kterou díky schopnosti extrakce příznaků ze vstupních dat použili k získání příznaků popisujících hrany v zašumělém snímku [14]. Samotnou úlohu redukce šumu pak neprovedli pomocí NN, jelikož podle autorů se současné metody na bázi hlubokého učení chovají jako černé skříňky, do jejichž vnitřní struktury nevidíme, a proto je obtížné jim důvěřovat pro další použití, na druhou stranu často dosahují lepších výsledků než konvenční metody. Proto použili CNN pouze k detekci hran, ale ne k samotné redukci šumu, ke které navrhli svou metodu. Díky tomu udrželi možnost transparence celého řešení a zároveň zlepšili schopnost redukce šumu pro metodu bez použití

hlubokého učení.

Další článek z konce roku 2020 publikovali You se svým týmem, kteří se zabývali clusterováním multi-view dat v doméně latentní reprezentace [74]. Latentní prostor dokáže postihnout lokální strukturální informaci z každého pohledu, ale také globální vlastnosti nasnímaných dat, díky čemuž navržená metoda výrazně zjednodušila clusterování velkého množství vícedimenzionálních dat a dosáhla vysoké efektivity.

V roce 2020 publikovali článek také Li se svým týmem, kteří zkoumali využití latentní reprezentace při redukci oparu nebo zamlžení snímků, což je velmi důležité v oblasti strojového vidění [42]. Navrhli metodu založenou na detekci prostorového rozložení mlhy s cílem získání čistého snímku bez jiných artefaktů. Architekturu postavili na dvou částech, první natrénovali k získání čistého snímku, druhou pak k odstranění vzniklých artefaktů. Navrhli také verzi pro použití v oblasti detekce objektů. Dále provedli poměrně podrobné srovnání s několika dalšími metodami odstranění mlhy a dosáhli výborných výsledků.

Poslední zde zmíněný článek z roku 2020 publikovali Yesilyurt a Kamisli, kteří se zabývali entropickým kódováním latentní reprezentace vzniklé při kompresi snímků kodeky založenými na učení [73]. Svou metodu založili na modelování hustoty pravděpodobnosti latentní reprezentace jako součinu podmíněných hustot pravděpodobností jednotlivých latentních proměnných. Dosáhli tak kompresní účinnosti podobné ostatním kodekům založeným na učení, ale uvedená metoda vyžadovala jednodušší NN a kratší dobu učení díky větší jednoduchosti a menšímu počtu parametrů.

V roce 2021 publikovali článek Shang se svým týmem, kteří se zabývali extrakcí příznaků s využitím latentní reprezentace [62]. Představili metodu, která brala v úvahu spojitosti prostoru vstupních dat (2D snímků) a latentního prostoru s cílem nejlepší extrakce příznaků. Samotnou extrakci příznaků pak provedli v latentním prostoru, čímž mimo jiné zvýšili odolnost vůči šumu vstupních dat. V porovnání s několika dalšími metodami dosáhli lepších výsledků, nevýhodou ale může být množství nastavitelných parametrů.

Tabulka 4.1 shrnuje všechny zmíněné články a přehledně je rozděluje do kategorií.

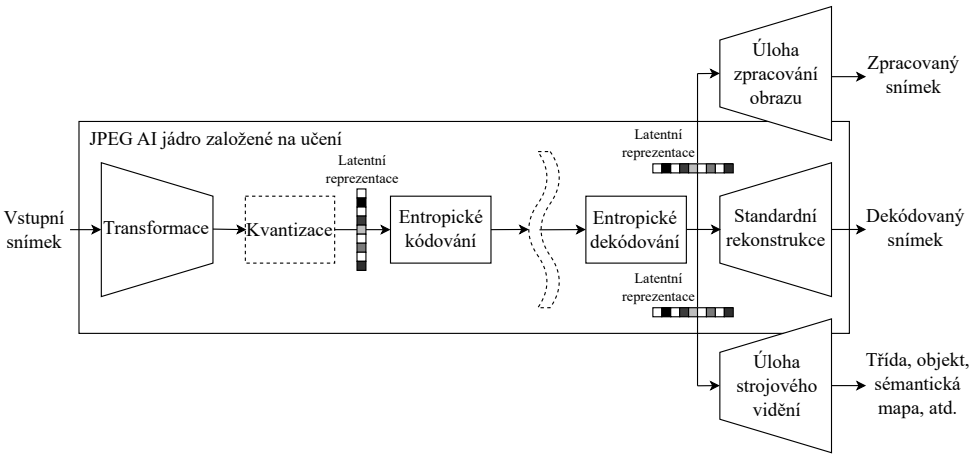
Tabulka 4.1: Rozdělení článků, které se zabývají využitím latentní reprezentace

Kategorie	Články
Extrakce příznaků, klasifikace	[81], [17], [37], [12], [77], [2], [62]
Segmentace, detekce, registrace	[72], [38], [80], [19], [48]
Redukce šumu, super-resolution, redukce kompresních artefaktů, interpolace	[76], [66], [78], [10], [14], [42]
Převod snímků mezi třídami, komprese	[53], [41], [46], [73]
Využití u speciálních dat (point cloud, hyperspektrální, multi-view)	[63], [79], [75], [74]

4.1.3 JPEG AI

V rámci skupiny JPEG byla oblast využití latentní reprezentace diskutována na jejím 90. setkání, které se konalo v lednu 2021 a ze kterého vzešel i dokument definující možné oblasti použití a požadavky na vstupní i komprimovaná data [27].

Obrázek 4.2 zachycuje celou myšlenku využití latentní reprezentace, kdy kromě standardní rekonstrukce s cílem získání rekonstruovaného snímku může být použita i pro úlohy zpracování obrazu, například redukce šumu, zvýšení kontrastu, nebo zvýšení rozlišení, a úlohy strojového vidění, které provádí extrakci vysokoúrovňové sémantické informace, například s cílem klasifikace do tříd, nebo segmentace do regionů [27].



Obrázek 4.2: Rámcová struktura kodéru JPEG AI [27] – přeloženo

Kódovaný bitový tok tak nabízí poměrně širokou variabilitu použití a může vést ke zjednodušení některých úloh zpracování obrazu a strojového vidění díky použití latentní reprezentace jako vstupu namísto originálního nebo dekódovaného snímku. V rámci standardizační aktivity JPEG AI bylo definováno několik základních úloh z obou skupin, kterých se toto využití může dotknout [27]:

- Úlohy zpracování obrazu
  - Super-resolution
  - Redukce šumu
  - Vylepšení obrazu s nízkou úrovní osvětlení
  - Barevná korekce
  - Kompenzace expozice
  - Retušování
- Úlohy strojového vidění
  - Klasifikace a vyhledávání obrázků

- Detekce, rozpoznání a identifikace objektů
- Sémantická segmentace
- Detekce události a rozpoznání akce
- Detekce a rozpoznání obličeje

Kromě těchto úloh byly definovány také možné oblasti použití [27]:

- Cloudová úložiště
- Vizuální dohled – video monitoring
- Autonomní vozidla a jiná zařízení
- Správa a organizace snímků
- Monitorování a kontrola streamovaných obrázků a videí
- Distribuce médií

Nakonec byly definovány požadavky, které by měl standard JPEG AI dodržet, aby umožnil své nasazení do výše zmíněných úloh a oblastí použití. V požadavcích na vstupní data se jedná především o rozlišení, bitovou hloubku, barevný prostor a typ obsahu. Nároky na komprimovaný datový tok zahrnují hlavně vysokou kompresní účinnost při zachování vysoké kvality, možnost použití pro úlohy zpracování obrazu a strojového vidění, vhodnou implementaci s ohledem na výpočetní složitost, paměťovou náročnost, nebo možnost paralelizace, nebo například možnost progresivního dekódování, které umožňuje alespoň nějaký náhled, přestože zatím nebyla přijata všechna data. Úplný seznam lze nalézt v dokumentu [27]. Významným cílem je také nalezení takového řešení, které nebude zatíženo licenčními poplatky [27].

4.2 Budoucí vývoj

Při zpětném pohledu na zmiňované články zabývající se využitím latentní reprezentace si lze všimnout, že žádný neposkytuje ucelený scénář využití latentní reprezentace současně pro kompresi a některou z úloh zpracování obrazu. V současné době zároveň neexistují žádné volně dostupné implementace, které by takový scénář využívaly, s výjimkou malého náznaku na databázi MNIST (Modified National Institute of Standards and Technology), která obsahuje velké množství ručně psaných číslic a je běžně používána pro trénování různých systémů zpracování obrazu [69].

Celé téma je tak velmi nové a ve fázi výzkumu a otevírá tak nové možnosti další práce. Jednou z nich je nalezení již existujících řešení provádějící dílčí úlohy komprese a zpracování obrazu a jejich vhodné zkombinování do jednoho funkčního celku. Jako metoda komprese obrazu založená na učení vhodná pro takový scénář se jeví v práci zmiňovaná metoda Ballé [3], která dosahuje výborné kompresní účinnosti, umožňuje použití různých modelů včetně trénování nových a její implementace je volně dostupná. Dala by se tak zkombinovat

s jinou metodou založenou na učení realizující některou z úloh zpracování obrazu, například potlačení šumu, a dosáhnout tím požadovaného výsledku. Takový systém by bylo možné trénovat buď jako celek, nebo ponechat část realizující kompresi fixní s daným již natrénovaným modelem a trénovat jen část, která provádí zpracování obrazu. Další možností, jak k řešení přistupovat, je kompletní návrh celého řešení od základu. Výsledné systémy je pak potřeba porovnat s již prověřenými a používanými metodami a vyhodnotit jejich úspěšnost.

Kapitola 5

Porovnání metod komprese obrazu založených na učení a konvenčních kodeků

Jak bylo zmíněno v části 3.3 Implementace diskutované v rámci JPEG AI, v této kapitole je proveden experiment porovnávající konvenční kodeky ke kompresi obrazu a několik metod založených na učení.

5.1 Metodika experimentu

Dokument [29] přesně popisuje metodiku testování kodeků založených na strojovém učení v rámci JPEG AI. Experiment v této práci byl tímto postupem inspirován, obsahuje ale i několik odlišností.

Jako zástupci konvenčních kodeků byly pro experiment zvoleny kodeky JPEG, JPEG 2000, HEVC Intra a JPEG XL. JPEG je vůbec nejrozšířenějším standardem na světě, a tak není pochyb o jeho důležitosti. Vedle toho JPEG 2000 je kodekem využívaným spíše ve vědecké praxi nebo například lékařství, kde je důležité, aby výsledný snímek vykazoval minimální zkreslení. Co se týče kodeku HEVC Intra, jeho podstatou je využití vnitrosnímkové predikce z kodeku HEVC, který je určen ke kompresi videa. Tento kodek prokázal svou vysokou účinnost a v posledních letech se jeví jako jedno z kvalitních řešení, které by mohlo částečně nahradit původní JPEG – například společnost Apple již ve svých mobilních telefonech několik posledních generací nabízí kompresi fotografií právě tímto kodekem a ukládání do kontejneru HEIF (High Efficiency Image File Format) [21, 56].

Poslední jmenovaný kodek, tedy JPEG XL, je jedním z nejnovějších kodeků skupiny JPEG. Vzniká od roku 2017 a v současné době podstupuje poslední kroky vedoucí k jeho standardizaci, přičemž je již dostupná i oficiální referenční implementace. Hlavními cíli standardizace zde jsou vysoká kompresní účinnost, podpora mnoha základních i pokročilých aplikací (široký barevný gamut WCG (Wide Color Gamut), vysoký dynamický rozsah HDR (High Dynamic Range), podpora alpha kanálu a další) a osvobození od licenčních poplatků, díky čemuž má potenciál nahradit původní JPEG. Mezi zajímavé vlastnosti patří bezeztrátové transkódování JPEG–JPEG XL, kdy JPEG XL dokáže ještě více komprimovat JPEG data, dokáže ale také bezeztrátově obnovit původní DCT koeficienty JPEG [30].

K získání komprimovaného datového toku pomocí těchto kodeků byly dle doporučení z dokumentu [29] použity následující implementace:

- JPEG
  - Referenční software JPEG XT
  - Dostupný z: `https://jpeg.org/jpegxt/software.html`
- JPEG 2000
  - Kakadu software
  - Dostupný z: `https://kakadusoftware.com/`
- HEVC Intra
  - HEVC Test Model
  - Dostupný z: `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8/`
- JPEG XL
  - Referenční software JPEG XL
  - Dostupný z: `https://jpeg.org/jpegxl/software.html`

Kromě klasického porovnání kodeků pomocí několika metrik pro různé bitové toky v podobě R–D křivek je součástí experimentu také analýza výpočetní a časové náročnosti diskutovaných implementací a integrální srovnání všech kodeků pomocí Bjøntegaardovy metriky [4].

5.2 Implementace kodeků založených na učení

Z dostupných implementací kodeků na bázi strojového učení byly k testování a porovnání zvoleny následující:

- Toderici et al. – Full Resolution Image Compression with Recurrent Neural Networks [67]
  - `https://github.com/tensorflow/models/tree/archive/research/compression/image_encoder`
- Ballé et al. – End-to-end Optimized Image Compression [3]
  - `https://github.com/tensorflow/compression`
- Lee et al. – Context-adaptive Entropy Model for End-to-end Optimized Image Compression [40]
  - `https://github.com/JooyoungLeeETRI/CA_Entropy_Model`

Tabulka 5.1: Parametry zvolených implementací [34] – přeloženo a upraveno

Kodek	Typ NN	Velikost kódovací jednotky	Nastavení bitového toku
Toderici [67]	Rekurentní AE s konvolučními vrstvami	32 × 32	Nastavitelný bitový tok v rámci jednoho modelu
Ballé [3]	Konvoluční AE s nelineární transformací	256 × 256	Pro každý bitový tok jiný model
Lee [40]	Konvoluční AE	256 × 256	Pro každý bitový tok jiný model

Všechny tři zmíněné publikované články popisující výzkum, realizaci, použití a výsledky jednotlivých metod a softwarových implementací byly zmíněny a stručně popsány výše v části 2.2.2 Současný stav. Tabulka 5.1 ukazuje důležité parametry jednotlivých implementací.

Implementace metody Toderici navíc dokáže provést dekódování vzniklého bitového toku do všech dostupných nižších kvalit, než jaká byla nastavena při kódování. Metoda Ballé umožňuje navíc použití mnoha různých předpřipravených modelů a metoda Lee umožňuje výběr ze dvou modelů.

Metoda Lee se také zaměřuje především na adaptivní entropické kódování, nicméně dostupná implementace umožňuje kompletní kompresi a dekompresi, a tak byla zvolena také. Obecně vychází z práce Ballého s týmem, tedy článku [3], a tak jsou parametry implementace velmi podobné, ale díky optimalizovanému entropickému kódování by měla metoda Lee dosahovat v metrikách, pro které byla optimalizována, lepších výsledků.

Ostatní implementace zmíněné v dokumentu [34] neumožňují provést kompletní kompresi a dekompresi obrázku, tedy použití vhodné pro tento experiment, a to většinou z důvodu, že se zaměřují jen na část kodeku a bylo by nutné je použít jako součást jiného kódovacího schématu.

5.3 Zprovoznění a nastavení implementací jednotlivých kodeků

Tato část je věnována zprovoznění a nastavení všech výše zmíněných implementací kodeků, tedy jak těch konvenčních, tak těch založených na strojovém učení. Část týkající se zprovoznění byla zařazena z toho důvodu, že ne vždy byl dobře prezentován seznam systémových požadavků a programového vybavení k testování dané implementace. Dále jsou zde zmíněna možná nastavení parametrů jednotlivých kodeků a na závěr uvedeny příkazy, které byly k testování použity.

5.3.1 Zprovoznění implementací

Nejprve je potřeba zmínit, že testování bylo provedeno na platformě Windows. Některé z vybraných implementací jsou nicméně dostupné jen pro platformu Linux/UNIX, a tak bylo k dispozici několik možností, jak tento problém vyřešit. Jelikož společnost Microsoft v posledních letech zahrnula do svého operačního systému Windows 10 funkcionalitu WSL (Windows Subsystem for Linux), byla v experimentu použita. Zprovoznění WSL v rámci Windows 10 je poměrně jednoduché a přímočaré, vše je možné nastavit podle návodu na stránce `https://docs.microsoft.com/en-us/windows/wsl/install-win10`. Jelikož všechny tři diskutované kodeky založené na strojovém učení jsou napsané v programovacím jazyce Python a využívají jako framework hlubokého učení knihovnu TensorFlow, je potřeba se zabývat instalací i těchto součástí, nicméně zde již u některých kodeků záleží na platformě a verzi. Následujících několik odstavců se proto věnuje všem kodekům po jednom.

Co se týče implementace metody Toderici, je možné ji spustit přímo ve Windows bez nutnosti WSL. Potřebné programové vybavení zahrnuje jazyk Python 3.6.0 a knihovny TensorFlow 1.15.0, Protocol Buffers 3.6.0 a NumPy 1.19.3. Dále je potřeba stáhnout model `residual_gru.pb`. Celý postup přehledně shrnuje několik následujících bodů:

1. Stáhnout `python-3.6.0-amd64.exe` z `https://www.python.org/downloads/release/python-360/` a nainstalovat
2. Stáhnout `tensorflow-1.15.0-cp36-cp36m-win_amd64.whl` z `https://pypi.org/project/tensorflow/1.15.0/#files`
3. Otevřít Příkazový řádek
4. Nastavit cestu na adresář se staženými soubory, ve výchozím stavu
   - `cd c:\users\<username>\downloads`
5. Nainstalovat knihovny TensorFlow, Protocol Buffers a NumPy
   - `pip install tensorflow-1.15.0-cp36-cp36m-win_amd64.whl`
   - `pip install protobuf==3.6.0`

- `pip install numpy==1.19.3`

6. Z výše uvedeného odkazu na stránku dané implementace na serveru GitHub stáhnout skripty `encoder.py` a `decoder.py`

7. Stáhnout potřebný model z `http://download.tensorflow.org/models/compression_residual_gru-2016-08-23.tar.gz`

Zprovoznění implementace Ballého metody již vyžaduje aktivaci WSL. V tomto prostředí je pak potřeba opět nainstalovat Python a TensorFlow, byly zvoleny stejné verze jako pro metodu Toderici, tedy Python 3.6.0 a TensorFlow 1.15.0. V tomto případě je jediným rozdílem potřeba stažení správné verze z dříve odkazovaných stránek, tedy verze pro Linux. Po správné instalaci je ještě potřeba nainstalovat modely používané daným kodekem. Následující posloupnost kroků tento postup přehledně shrnuje:

1. Zapnout Ubuntu (WSL) a nainstalovat nutné základy na čisté instalaci

   - `sudo apt-get update`
   - `sudo apt-get install build-essential`
   - `sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev`

2. Stáhnout `Python-3.6.0.tar.xz` z `https://www.python.org/downloads/release/python-360/`

3. Extrahovat a nainstalovat Python

   - `cd /mnt/c/users/<username>/downloads`
   - `tar -xf Python-3.6.0.tar.xz`
   - `cd Python-3.6.0`
   - `./configure`
   - `sudo make install`
   - `sudo pip3 install -upgrade pip`

4. Nastavit Python 3.6 jako výchozí verzi

   - `sudo update-alternatives -config python`
   - `sudo update-alternatives -install /usr/bin/python python /usr/local/bin/python3.6 1`

5. Stáhnout `tensorflow-1.15.0-cp36-cp36m-manylinux2010_x86_64` z `https://pypi.org/project/tensorflow/1.15.0/#files`

6. Nainstalovat TensorFlow

   - `cd ..`

- `pip install tensorflow-1.15.0-cp36-cp36m-manylinux2010_x86_64.whl`

7. Z výše uvedeného odkazu na stránku dané implementace na serveru GitHub stáhnout z adresáře `models` skript `tfci.py`

8. Nainstalovat potřebné modely

   - `pip install tensorflow-compression==1.3`

Zprovoznění implementace metody Lee je po nastavení metody Toderici již velmi snadné, jelikož kodek opět běží přímo ve Windows a dříve nainstalované verze jazyka Python i knihovny TensorFlow je možné pro tuto implementaci použít. Je potřeba jen nainstalovat knihovny SciPy, Requests a Python Imaging Library pomocí příkazů `pip install scipy`, `pip install requests` a `pip install pillow`. Z výše uvedeného odkazu na stránku dané implementace na serveru GitHub je potřeba stáhnout celý obsah adresáře `CA_Entropy_Model`. Modely používané kodekem budou staženy automaticky po prvním spuštění skriptu ke kódování.

Použití dříve odkazovaných implementací kodeků JPEG a JPEG 2000 je poměrně snadné. Co se týče JPEG, je potřeba opět použít WSL, ve kterém je třeba zkompilovat kód referenčního softwaru pomocí příkazu `make final`. Pokud jde o JPEG 2000, byla použita verze Kakadu SW pro Linux běžící opět ve WSL. Po stažení a rozbalení je pouze potřeba přidat aktuální adresář do dvou proměnných podle návodu v souboru `README.txt`, který je součástí archivu.

Zprovoznění kodeku HEVC Intra je opět lehce komplikovanější, a tak celý postup shrnuje několik následujících bodů:

1. Zapnout Ubuntu (WSL)

2. Změnit cestu na pracovní adresář

3. Stáhnout dříve odkazovanou implementaci

   - `wget -r -np -nH -cut-dirs=3 -e robots=off -R index.html.tmp -R index.html https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8/`

4. Nainstalovat starší verzi překladačů `gcc` a `g++`

   - `sudo apt-get install gcc-7`
   - `sudo apt-get install g++-7`
   - `sudo update-alternatives -install /usr/bin/gcc gcc /usr/bin/gcc-7 800 -slave /usr/bin/g++ g++ /usr/bin/g++-7`

5. Změnit cestu na adresář pro kompilaci na platformě Linux a zkompilovat

   - `cd HM-16.20+SCM-8.8/build/linux`

- `make`

Posledním kodekem je JPEG XL, jehož zprovoznění je přehledně popsáno v souboru `README.md`, který je dostupný na odkazované stránce referenčního softwaru JPEG XL. Pro úplnost je postup shrnut i zde:

1. Zapnout Ubuntu (WSL)
2. Změnit cestu na pracovní adresář
3. Stáhnout implementaci a důležité součásti
   - `git clone https://gitlab.com/wg1/jpeg-xl.git -recursive`
   - `sudo apt install cmake pkg-config libbrotli-dev`
4. Nainstalovat a nastavit překladač Clang
   - `sudo apt install clang-7`
   - `export CC=clang-7 CXX=clang++-7`
5. Zkompilovat referenční software JPEG XL
   - `cd jpeg-xl`
   - `mkdir build`
   - `cd build`
   - `cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_TESTING=OFF ..`
   - `cmake -build . - -j$(nproc)`

5.3.2 Nastavení implementací

Každá z implementací nabízí několik nastavitelných parametrů. V této části jsou přehledně představeny použité příkazy ke kompresi a dekompresi a význam jednotlivých použitých parametrů pro všechny implementace. Některé z nich nabízí velké množství dalších parametrů, které ale nebyly použity, a tak zde nejsou zmíněny.

Tabulka 5.2 ukazuje příkazy použité ke kompresi vstupního snímku a dekompresi komprimovaného bitového toku pomocí jednotlivých kodeků.

Co se týče všech implementací, `<input>`, resp. `<output>` znamená název vstupního, resp. výstupního snímku. Pro upřesnění je v tabulce uvedena vždy i přípona názvu souboru, udávající jeho typ. Lze si všimnout, že v rámci JPEG, resp. JPEG 2000, resp. HEVC Intra není vstupem soubor PNG, ale PNM, resp. BMP, resp. YUV. Tato změna je dána zvolenými implementacemi, které soubor PNG na vstupu nepodporují. Potřebný převod je pro kodeky JPEG a JPEG 2000 proveden v programu MATLAB, pro kodek HEVC Intra pomocí programu `ffmpeg`, jehož stažení a instalace jsou popsány později v části týkající se zprovoznění metriky VMAF. V rámci JPEG 2000, resp. HEVC Intra je potřeba provést převod i výstupního snímku po dekompresi z formátu BMP, resp. YUV na PNG.

Tabulka 5.2: Přehled použitých příkazů pro kompresi a dekompresi

Kodek	Komprese/ dekomprese	Použitý příkaz
Toderici [67]	Komprese	`python encoder.py -input_image=<input>.png -output_codes=<output>.npz -iteration=<qp> -model=residual_gru.pb`
	Dekomprese	`python decoder.py -input_codes=<input>.npz -output_directory=<output_dir> -model=residual_gru.pb`
Ballé [3]	Komprese	`python tfci.py compress <model>-<qp> <input>.png`
	Dekomprese	`python tfci.py decompress <input>.tfci`
Lee [40]	Komprese	`python encode.py -model_type <model> -input_path <input>.png -compressed_file_path <output>.cmp -quality_level <qp>`
	Dekomprese	`python decode.py -compressed_file_path <input>.cmp -recon_path <output>.png`
JPEG	Komprese	`./jpeg -q <qp> -h -qt 3 -s 1x1,2x2,2x2 <input>.pnm <output>.jpeg`
	Dekomprese	MATLAB: `im=imread('<input>.jpeg'); imwrite(im,'<output>.png');`
JPEG 2000	Komprese	`./kdu_compress -i <input>.bmp -o <output>.jp2 -rate <qp> Qstep=0.001 -tolerance 0 -full -precise`
	Dekomprese	`./kdu_expand -i <input>.jp2 -o <output>.bmp`
HEVC Intra	Komprese	`./TAppEncoderStatic -c ../cfg/encoder_intra_main_scc.cfg -f 1 -fr 1 -q <qp> -wdt <width> -hgt <height> -i <input>.yuv -b <output>.bin -o /dev/null`
	Dekomprese	`./TAppDecoderStatic -b <input>.bin -o <output>.yuv`
JPEG XL	Komprese	`./cjxl -q <qp> <input>.png <output>.jxl`
	Dekomprese	`./djxl <input>.jxl <output>.png`

V rámci metody Toderici dochází při dekompresi k dekódování do stejné a všech nižších kvalit, než do jaké bylo provedeno kódování, a tak je pro výstup dekódování uveden vždy adresář `<output_dir>`.

Parametr `<model>` v rámci Ballého metody nabízí výběr jednoho z mnoha modelů dostupných pro danou implementaci a vytvořených v rámci několika různých prací. V rámci práce [3] byl vytvořen model `bmshj2018`, a to ve 4 variantách:

- `bmshj2018-factorized-mse`,
- `bmshj2018-factorized-mssim`,
- `bmshj2018-hyperprior-mse`,
- `bmshj2018-hyperprior-mssim`.

Z těchto byl pro experiment použit model `bmshj2018-factorized-msssim`, který podle [31] dává nejlepší průměrné výsledky v rámci R–D křivek napříč různými metrikami.

V rámci metody Lee umožňuje parametr `<model>` výběr ze dvou hodnot:

- 0 – optimalizovaný pro MSE,
- 1 – optimalizovaný pro MS-SSIM.

Jelikož byl i v Ballého metodě zvolen model optimalizovaný pro MS-SSIM, byl vybrán i v případě metody Lee.

U komprese JPEG je použitých parametrů hned několik, jejich význam je následující:

- `-q <qp>` – nastavení kvality komprese,
- `-h` – použití optimálního Huffmanova kódování,
- `-qt 3` – použití vizuálně vylepšených kvantizačních tabulek,
- `-s 1x1,2x2,2x2` – použití barevného podvzorkování 4:2:0.

U komprese JPEG 2000 je situace podobná, význam použitých parametrů je následující:

- `-rate <qp>` – nastavení kvality komprese,
- `Qstep=0.001` – škálovací kvantizační parametr (neovlivňuje výsledný bitový tok),
- `-tolerance 0 -full` – nejvyšší efektivita využití dostupného bitového toku,
- `-precise` – vynucené použití přesnější numeriky.

Několik parametrů vyžaduje také kodek HEVC Intra:

- `-c ../cfg/encoder_intra_main_scc.cfg` – definování konfiguračního souboru (použit soubor doporučený v dokumentech [31] a [29]),
- `-f 1` – počet snímků, které mají být kódovány,
- `-fr 1` – počet snímků za sekundu vstupních dat (pro statický snímek 1),
- `-q <qp>` – nastavení kvality komprese,
- `-wdt <width>` – šířka vstupního snímku,
- `-hgt <height>` – výška vstupního snímku,
- `-o /dev/null` – žádný rekonstruovaný snímek.

Kodek JPEG XL umožňuje nastavení velkého množství parametrů týkající se například barevného zpracování, rychlosti kodeku, nebo kvality. Kvalitu umožňuje dokonce nastavit pomocí tří různých způsobů – pomocí Butteraugli metriky ([20, 18]) vyjadřující psychovizuální podobnost původního a komprimovaného snímku v rozsahu od 0 (nejvyšší kvalita) do 15 (nejnižší kvalita), pomocí nastavení kvalitativního parametru `-q` v rozsahu od $-\infty$ (nejnižší kvalita) do 100 (nejvyšší kvalita), které v rozsahu od 0 do 100 hrubě odpovídá nastavení kvality JPEG, a pomocí přímého nastavení cílového bitového toku. Ze všech možných parametrů byla vždy nastavována pouze kvalita. Jelikož nastavení kvality pomocí Butteraugli metriky nedosahuje ani na nejnižší kvalitu požadovaných nízkých bitových toků a přímé nastavení cílového bitového toku nefunguje spolehlivě, bylo zvoleno použití kvalitativního parametru, tedy:

- `-q <qp>` – nastavení kvality komprese.

Pro účely komprese je nejdůležitější parametr nastavující bitový tok komprimovaného souboru a tedy výslednou kvalitu rekonstruovaného obrazu. Ve všech kodecích je označen jako kvantizační parametr `<qp>` (Quantization Parameter). Možnosti nastavení pro jednotlivé kodeky shrnuje Tabulka 5.3.

Tabulka 5.3: Možnosti nastavení jednotlivých kvantizačních parametrů

Kodek	Možnost nastavení kvantizačního parametru
Toderici [67]	0 až 15
Ballé [3]	1 až 8
Lee [40]	1 až 9
JPEG	0 až 100
JPEG 2000	přesný bitový tok v bpp
HEVC Intra	0 až 51
JPEG XL	$-\infty$ až 100

5.4 Výběr a příprava testovacích snímků

Dokument [29] popisuje také databázi obrazového obsahu JPEG AI, která byla vytvořena za účelem trénování, ověření a testování kodeků založených na učení. Obecně jde o databázi snímků ve formátu PNG s rozlišením od 256 × 256 do 8K s bitovou hloubkou 8 bitů na barevný kanál s plným barevným rozlišením, tedy 24 bitů na pixel. Část určená k natrénování modelu obsahuje několik tisíc snímků, část určená k ověření konvergence navrženého řešení pak několik stovek. Část určená k testování je pro všechny skrytá do doby, než poskytnou nějaké řešení kodéru a dekodéru, aby nemohla být použita k trénování.

V experimentu jsou použity v jednotlivých implementacích již natrénované modely, které jsou poskytnuty autory daných řešení, kteří k trénování použili vlastní databáze snímků. Takovou situaci dokument [29] také připouští. Pro experiment byla vytvořena testovací sada snímků, do které bylo zvoleno několik snímků z databází Kodak, McMaster, CBSD68 a CSet8 společně s několika snímky ze stránky Signature Edits [15] poskytující RAW obrazová data různých scén ve vysokém rozlišení volně ke stažení a jakémukoliv použití. Informace k jednotlivým databázím jsou následující:

- Kodak
  - Počet snímků: 24
  - Formát: PNG
  - Rozlišení: 768 × 512 (nebo 512 × 768)
  - Dostupné z: `http://r0k.us/graphics/kodak/`

- McMaster
  - Počet snímků: 18
  - Formát: TIFF
  - Rozlišení: 500 × 500
  - Dostupné z: `https://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm`
- CBSD68
  - Počet snímků: 68
  - Formát: PNG
  - Rozlišení: 481 × 321 (nebo 321 × 481)
  - Dostupné z: `https://github.com/clausmichele/CBSD68-dataset/tree/master/CBSD68/original_png`
- CSet8
  - Počet snímků: 8
  - Formát: PNG
  - Rozlišení: 256 × 256
  - Dostupné z: `https://github.com/ysix7/Dataset/blob/master/CSet8.zip`

Pro účely experimentu byly z důvodu sjednocení dat a omezení metody Toderici, která vyžaduje na vstupu snímek s rozměry v násobcích 32, snímky z databáze McMaster převedeny do formátu PNG a oříznuty na velikost 480 × 480 a snímky z databáze CBSD68 oříznuty na velikost 480 × 320. U databáze McMaster bylo oříznutí provedeno odebráním 10 obrazových bodů z každé strany snímku, u databáze CBSD68 odebráním posledního řádku a posledního sloupce. RAW obrazová data ze stránky Signature Edits byla také převedena do formátu PNG a oříznuta pro potřeby některých implementací. Snímky byly vždy oříznuty na nejvyšší možné požadované rozlišení a následně naškálovány na jedno nebo dvě nižší rozlišení, aby byly k dispozici k testování kodeků. Všechny konverze, oříznutí a změny velikosti byly provedeny v programu MATLAB, přičemž při změně velikosti byla použita funkce `imresize` s výchozím nastavením, které využívá bikubickou interpolaci. Všechny výsledné snímky jsou tak připraveny ve formátu PNG.

Výsledné snímky pokrývají širokou škálu rozlišení od nejnižšího 256 × 256 až po nejvyšší 7680 × 5120 pixelů. Většina snímků s nižším rozlišením pochází ze zmíněných databází, které bývají ve zpracování obrazu často používány. Naopak snímky s vysokým rozlišením jsou získány z RAW obrazových dat. Co se týče dalších parametrů, všechny výsledné snímky mají bitovou hloubku 8 bitů na kanál s plným barevným rozlišením, tedy 24 bitů na pixel.

Jak bylo již zmíněno, k testování bylo z každé databáze zvoleno jen několik snímků, a to s cílem získat testovací sadu rozmanitých snímků z hlediska

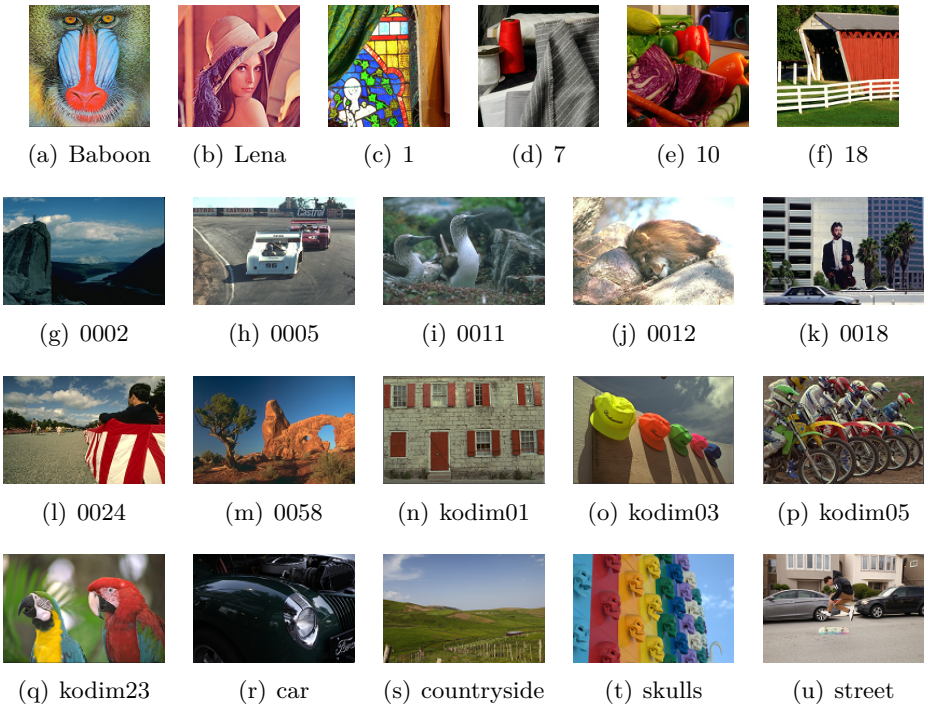
obsažené prostorové frekvence, barevnosti snímků nebo úrovně rozostření pozadí. Tabulka 5.4 ukazuje všechna rozlišení, která byla vybrána k testování, a k nim odpovídající použité testovací snímky. V tabulce je také zaznamenán celkový počet pixelů pro dané rozlišení pro rychlou představu o často udávaném rozlišení digitálních fotoaparátů v megapixelech (Mpx). Na Obrázku 5.1 je pak vidět náhled všech snímků zvolených k testování.

Tabulka 5.4: Použitá rozlišení a odpovídající snímky

Rozlišení			Snímky	
Horizontální × vertikální počet pixelů	Celkový počet pixelů [Mpx]	Poměr stran	Databáze/ zdroj	Názvy snímků [*.png]
256 × 256	0,066	1:1	CSet8	Baboon, Lena
480 × 320	0,154	3:2	CBSD68	0002, 0005, 0011, 0012, 0018, 0024, 0058
480 × 480	0,230	1:1	McMaster	1, 7, 10, 18
768 × 512	0,393	3:2	Kodak	kodim01, kodim03, kodim05, kodim23
1920 × 1280	2,458	3:2	Signature Edits	car, countryside, skulls, street
3840 × 2560	9,830	3:2		
7680 × 5120	39,322	3:2		car

Rozlišení snímků ze stránky Signature Edits jsou zvolená tímto způsobem na základě dvou kritérií:

1. Počet obrazových bodů na šířku obrazu musí odpovídat známým hodnotám pro rozlišení FHD, UHD-1 a UHD-2.
2. Poměr stran musí odpovídat nejpoužívanějšímu poměru stran v digitálních fotoaparátech, tedy 3:2.



(a) Baboon (b) Lena (c) 1 (d) 7 (e) 10 (f) 18

(g) 0002 (h) 0005 (i) 0011 (j) 0012 (k) 0018

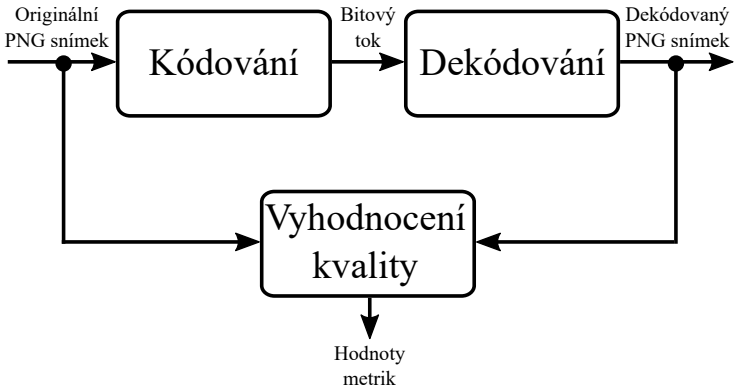
(l) 0024 (m) 0058 (n) kodim01 (o) kodim03 (p) kodim05

(q) kodim23 (r) car (s) countryside (t) skulls (u) street

Obrázek 5.1: Náhled všech použitých testovacích snímků

5.5 Komprese a dekomprese

Obrázek 5.2 ukazuje vlastní postup při hodnocení účinnosti komprese. V rámci experimentu je potřeba všechny výše uvedené vybrané snímky kódovat a dekódovat pomocí všech vybraných kodeků. Následně je potřeba porovnat originální a dekódovaný PNG snímek a vyhodnotit kvalitu/míru zkreslení pomocí vhodných metrik. Výstupem jsou hodnoty použitých metrik pro různé snímky a různé použité kodeky.



Obrázek 5.2: Standardní procedura vyhodnocení míry zkreslení vzniklé kompresí

Dokument [29] přesně definuje cílové bitové toky, které mají být pomocí komprese dosaženy – 0,03; 0,06; 0,12; 0,25; 0,50; 0,75; 1,00; 1,50 a 2,00 bpp. Jednotka bpp představuje zkratku pro počet bitů na pixel a spočítá se tedy jako

$$bpp = \frac{b}{p}, \tag{5.1}$$

kde b je celkový počet bitů komprimovaného snímku (vzniklého bitového toku) a p je celkový počet pixelů daného snímku. V reálu není vždy možné dosáhnout přesně požadovaného bitového toku, proto dokument definuje i maximální odchylku od požadovaného bitového toku, a to 15 %.

Jak bylo již zmíněno, každý ze zvolených kodeků umožňuje nastavení kvality a tedy bitového toku pomocí nějakého kvalitativního parametru, který ale většinou nabývá jen několika hodnot. Z tohoto důvodu v experimentu nebyly dosaženy výše zmíněné bitové toky, ale byla provedena komprese pro všechny dostupné hodnoty kvality u každého kodeku a bitový tok vždy nějaký vyšel. Výsledné R–D křivky tak neobsahují přesně změřené hodnoty kvality pro všechny kodeky na stejných bitových tocích, nicméně po spojení výsledných bodů dávají alespoň určitou představu. Co se týče zvolených konvenčních kodeků, JPEG nabízí nastavení kvality v rámci kvantizačního parametru v rozsahu 0 až 100, použitá implementace kodeku JPEG 2000 umožňuje přímé nastavení bitového toku, HEVC Intra nabízí nastavení QP v rozsahu 0 až 51 a JPEG XL v rozsahu $-\infty$ až 100. V rámci komprese JPEG by tak podle doporučeného postupu měl být dosažen požadovaný bitový tok

v rámci externě řízeného cyklu, nicméně z důvodu, že nebyl přesně nastavován bitový tok ani u kodeků založených na strojovém učení, bylo pro JPEG pevně zvoleno 11 různých hodnot nastavení kvality od 0 do 100 s krokem 10. V rámci JPEG 2000 byly nastaveny hodnoty bitového toku definované výše. Obdobně jako u JPEG bylo u kodeku HEVC Intra pevně zvoleno 11 různých hodnot nastavení kvality, a to 10 hodnot od 0 do 45 s krokem 5 a 11. hodnota 51. Pro JPEG XL byly experimentálně zvoleny hodnoty kvality odpovídající přibližně požadovaným cílovým bitovým tokům, konkrétně se jedná o hodnoty -300, -100, -30, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

Po úspěšném nastavení a zprovoznění všech implementací se otevřela možnost automatizace celého procesu komprese a dekomprese. Za tímto účelem byl vytvořen skript v programu MATLAB s názvem `ML_comparison.m`. Funkce skriptu je poměrně přímočará:

1. Načítá originální snímky z dříve vybrané testovací sady ve formátu PNG.
2. Provádí jejich kompresi pomocí všech vybraných kodeků pro všechna dostupná nastavení kvality.
3. Vypočítává a ukládá hodnotu dosaženého bitového toku pro danou kompresi a nastavení kvality.
4. Provádí dekompresi všech vzniklých bitových toků na snímky ve formátu PNG.
5. Ukládá všechny dekomprimované snímky.

Pro správnou funkci implementace metody Lee v rámci tohoto skriptu je navíc potřeba v souboru `utils.py`, který je součástí dané implementace, změnit znak plného bloku, který je použit pro výpis průběhu komprese a dekomprese, na jiný znak, jinak dochází k chybě v kódování vypisovaného textu v rámci programu MATLAB.

5.6 Hodnocení kvality

Experiment byl zaměřen čistě na objektivní hodnocení kvality komprimovaného obrazu, nezahrnoval tedy subjektivní testy, jejich statistické vyhodnocení a výpočet korelačních koeficientů pro určení vypovídající hodnoty objektivních metrik. K objektivnímu hodnocení kvality může být použita řada metrik, které více či méně odpovídají subjektivnímu vjemu kvality. Dokument [29] obsahuje seznam doporučených metrik k objektivnímu hodnocení kvality včetně odkazů na zdrojové kódy jejich implementací.

Pro testování byly nejprve zvoleny pouze metriky MS-SSIM a klasické PSNR. Jak bylo zmíněno v kapitole 2.2.3 JPEG AI, testy korelovanosti objektivních metrik se subjektivním hodnocením popsané v [31] ukázaly, že pro kodeky založené na strojovém učení je korelovanost menší než pro konvenční, nicméně stále je nejvyšší pro metriku MS-SSIM. Nicméně vedle této studie byla v průběhu finalizace této práce v rámci aktivity JPEG AI provedena nová

studie [65], která zahrnuje více objektivních metrik, navíc subjektivní výsledky byly získány metodou crowdsourcing. Ze studie vyplynuly jako nejvhodnější k hodnocení kodeků založených na učení, ale i konvenčních kodeků, metriky: VIFP (Visual Information Fidelity Pixel-domain version), PSNR-HVS-M (Peak Signal-to-Noise Ratio Human Visual System Masking), FSIMc (Feature Similarity chrominance) a VMAF (Video Multimethod Assessment Fusion). Zásadním rozdílem oproti původní studii je tak posun metriky MS-SSIM až na 5./6. místo, na druhou stranu metriky VIFP a VMAF se ukázaly jako vhodné v obou studiích.

Na základě těchto závěrů byly všechny výše zmíněné metriky zařazeny vedle původně zvolených metrik PSNR a MS-SSIM. Metriky PSNR, MS-SSIM, VIFP a PSNR-HVS-M jsou založeny jen na jasové informaci, zatímco FSIMc a VMAF uvažují také informaci z barevných kanálů. Co se týče implementací, v rámci zjednodušení byly základní metriky PSNR a MS-SSIM počítány pomocí balíčku MeTriX MuX pro program MATLAB (dostupný z `https://github.com/sattarab/image-quality-tools/tree/master/metrix_mux`). Ačkoliv je možné některé další zvolené metriky počítat také pomocí tohoto balíčku, byly pro všechny ostatní metriky zvoleny implementace odkazované přímo ve studii [65]:

- VIFP: `https://pypi.org/project/sewar/`
- PSNR-HVS-M: `http://www.ponomarenko.info/psnrhvsm.htm`
- FSIMc: `https://www4.comp.polyu.edu.hk/~cslzhang/IQA/FSIM/FSIM.htm`
- VMAF: `https://github.com/Netflix/vmaf`

Co se týče zprovoznění jednotlivých metrik, nejsložitější je to s metrikou VMAF, ale následující odstavec obsahuje pro úplnost informace ke všem zvoleným metrikám. Pro metriky PSNR a MS-SSIM počítané pomocí balíčku MeTriX MuX stačí po jeho stažení a uložení do libovolného adresáře na disku před prvním použitím spustit pomocí programu MATLAB skript `configure_metrix_mux.m`, který nastaví potřebné cesty a předpřipraví některé části pro rychlejší výpočty. Pro metriku VIFP je podle návodu na odkazované stránce pouze potřeba provést příkaz `pip install sewar`, který nainstaluje příslušný Python balíček a po kterém je možné metriku (a několik dalších, které balíček nabízí) ihned používat. Metriky PSNR-HVS-M a FSIMc jsou na uvedených odkazech dostupné přímo jako funkce pro program MATLAB, a tak stačí po jejich stažení a uložení do libovolného adresáře na disku přidat cestu k nim do pracovního adresáře. Pro metriku VMAF je postup zprovoznění ve Windows a s využitím WSL shrnut v následujících několika bodech:

1. Stáhnout celý adresář `vmaf-master` z `https://github.com/Netflix/vmaf`
2. Stáhnout zdrojový kód programu FFmpeg z `http://ffmpeg.org/`

3. Zapnout Ubuntu (WSL)
4. Nainstalovat knihovnu `libvmaf` podle `https://github.com/Netflix/vmaf/blob/master/libvmaf/README.md`
   - `python3 -m pip install virtualenv`
   - `python3 -m virtualenv .venv`
   - `source .venv/bin/activate`
   - `pip install meson`
   - `sudo apt-get install nasm ninja-build doxygen`
   - Změnit cestu na složku `libvmaf`, která se nachází uvnitř staženého adresáře `vmaf-master`, při stažení do výchozí složky je cesta `cd /mnt/c/users/<username>/downloads/vmaf-master/libvmaf`
   - `meson build -buildtype release`
   - `ninja -vC build`
   - `ninja -vC build test`
   - `sudo ninja -vC build install`
5. Změnit cestu na složku se staženým programem FFmpeg, při stažení do výchozí složky je cesta `cd /mnt/c/users/<username>/downloads`
6. Extrahovat daný soubor pomocí `tar -xf ffmpeg-snapshot.tar.bz2`
7. Nainstalovat program FFmpeg podle `https://github.com/Netflix/vmaf/blob/master/resource/doc/ffmpeg.md`
   - `cd ffmpeg`
   - `./configure --enable-libvmaf`
   - `make -j4`
   - `sudo make install`
8. Uložit do libovolného adresáře na disku model `vmaf_v0.6.1.json` ze složky `vmaf-master/model`
9. Smazat vše stažené a extrahované

Pro samotné vyhodnocení byl podobně jako pro kompresi a dekompresi vytvořen skript v programu MATLAB s názvem `ML_metrics.m`. Metriky PSNR, MS-SSIM, PSNR-HVS-M a FSIMc jsou počítány přímo pomocí funkcí napsaných pro program MATLAB, a tak je jejich zakomponování do skriptu přímočaré. Metrika VIFP je vedle toho napsaná pro Python, a tak je nutné ji ze skriptu volat pomocí funkce `system` a příslušného příkazu. Samotnou hodnotu metriky je pak možno získat z druhé proměnné, kterou funkce `system` vrací, tedy výstup na příkazový řádek. Poslední metrika VMAF je napsaná jako knihovna pro jazyk C, přičemž, jak bylo výše zmíněno, byla použita společně s programem FFmpeg, navíc ne v prostředí Windows, ale v prostředí Linux pomocí WSL. Tuto metriku je tak potřeba také volat pomocí

funkce `system`, příkazu `wsl` a příslušného příkazu `ffmpeg`. V tomto případě je pak možné hodnotu metriky získat ze souboru s výsledky, který je vždy po výpočtu vytvořen. Tabulka 5.5 shrnuje použité příkazy k výpočtu metrik.

Funkce vytvořeného skriptu `ML_metrics.m` je opět poměrně přímočará a odpovídá Obrázku 5.2:

1. Načítá originální snímky z dříve vybrané testovací sady ve formátu PNG.
2. Načítá zkreslené snímky po kompresi a dekompresi různými kodeky pro různé bitové toky.
3. Vypočítává a ukládá hodnoty jednotlivých metrik pro danou kompresi a nastavení kvality.

Tabulka 5.5: Přehled použitých příkazů k výpočtu metrik

Metrika	Příkazy
PSNR	`psnr=metrix_mux(im_orig,im_dist,'PSNR');`
MS-SSIM	`msssim=metrix_mux(im_orig,im_dist,'MSSIM');`
VIFP	`[~,cmdout]=system('sewar vifp <im_orig_path> <im_dist_path>');` `vifp=str2double(cmdout(8:end));`
PSNR-HVS-M	`[psnr_hvs_m,~]=psnrhvsm(im_orig,im_dist);`
FSIMc	`[~,fsimc]=FeatureSIM(im_orig,im_dist);`
VMAF	`[~,~]=system(['wsl LD_LIBRARY_PATH=$LD_LIB RARY_PATH:/usr/local/lib/x86_64-linux-gnu; export LD_LIBRARY_PATH; ffmpeg -i <im_dist_path> -i <im_orig_path> -lavfi libvmaf=log_fmt=json:log_path=<log_file>: model_path=<path_to_model>/vmaf_v0.6.1.json -f null -']); f=fopen('<log_file>'); A=textscan(f,'%s'); fclose(f); delete <log_file>; A=A{1,1}; vmaf=str2double(char(A(34)));`

5.7 Hodnocení výpočetní náročnosti kodeků

Dokument [29] také doporučuje v rámci vyhodnocení uvést seznam parametrů charakterizujících složitost jednotlivých kodeků, implementační řešení a platformu, na které byly testy provedeny. Jde tedy o popis vlastního kódování a dekódování, složitost trénovací fáze zde není uvažována. Jde o následující parametry:

- počet parametrů modelu a jejich přesnost,
- časová náročnost při použití pouze CPU,

- časová náročnost při použití i GPU,
- použitý framework hlubokého učení,
- specifikace CPU a GPU.

Tabulka 5.6 ukazuje zmíněné parametry pro jednotlivé použité kodeky. Jelikož nebylo možné z důvodu nekompatibility grafické karty testovacího systému s použitým frameworkem zapojit k řešení i GPU (konkrétně se jedná o požadavek frameworku TensorFlow na CUDA architekturu ve verzi 3.0 a vyšší, nicméně použitá GPU NVIDIA GeForce GT635M dosahuje pouze verze 2.1), parametry týkající se GPU v tabulce chybí. Co se týče časové náročnosti, je pro každý jednotlivý kodek uvedena doba komprese jednoho snímku pro různé hodnoty nastavení kvality a pro různá rozlišení, jelikož tyto parametry měly na dobu komprese nezanedbatelný vliv. Hodnota nastavení kvality komprese je v tabulce uvedena relativně vzhledem k celému rozsahu kvalit nabízených daným kodekem, 0 % tedy odpovídá nejnižšímu možnému nastavení kvality pro daný kodek a naopak 100 % odpovídá nejvyššímu možnému nastavení kvality pro daný kodek. Kromě časové náročnosti bylo do tabulky přidáno i procento využití procesoru, jelikož ne vždy byl využit na 100 %, a množství využité paměti RAM (Random Access Memory) během procesu kódování, které se také zásadně lišilo. Procento využití procesoru je udáno jako jeho celkové využití během procesu kódování, zatímco pro využitou paměť RAM je udána jen hodnota využívaná přímo programem realizujícím kompresi. U procesoru je totiž hlavním výstupem fakt, zda komprese využívá celý dostupný výpočetní výkon a mohla by tak teoreticky proběhnout rychleji na lepším procesoru, nebo je omezení jinde. Použitý procesor je dvoujádrový IntelCore i5-3230M vyrobený 22 nm technologií se základní frekvencí 2,6 GHz a možností zvýšení pomocí Intel® Turbo Boost Technology až na 3,2 GHz. U paměti RAM je naopak důležitá absolutní hodnota jejího využití, která je udána jako střední hodnota nárůstu využití RAM oproti klidovému stavu za dobu komprese zaokrouhlená na celé stovky MB. Testovací systém má k dispozici 8 GB paměti RAM typu DDR3 s frekvencí 1600 MHz. Během testů časové náročnosti na daném systému neběžel žádný jiný program, pouze nutné úlohy operačního systému na pozadí, které způsobily vytížení procesoru maximálně 5 %. Čas byl změřen pomocí funkcí `tic` a `toc` v programu MATLAB.

Implementace metody Lee zatěžovala procesor na 100 % a využívala více RAM vždy jen na začátku komprese, po většinu doby byly hodnoty na hodnotách z tabulky. U komprese JPEG, JPEG 2000, HEVC Intra a JPEG XL odpovídá zmíněné využití RAM hodnotě přidělené WSL, kde komprese probíhala.

Ačkoliv bylo pro experiment vybráno a uvažováno celkem 26 snímků v rozlišení od 256 × 256 do 7680 × 5120 pixelů, v průběhu komprimování se objevil problém. Kvůli pouze 8 GB dostupné paměti RAM se nepodařilo komprimovat snímky v rozlišení 3840 × 2560 a 7680 × 5120, maximální možné rozlišení tak bylo 1920 × 1280. Jak je vidět v Tabulce 5.6, už pro rozlišení 1920 × 1280 některé kodeky vyžadovaly vysoké množství paměti RAM, pro vyšší rozlišení

Tabulka 5.6: Parametry jednotlivých kodeků

Kodek	Použitý jazyk	Použitý framework hlubokého učení	Počet parametrů modelu	Kvalita [%]	Rozlišení	Časová náročnost [s]	Využití CPU [%]	Využití RAM [MB]
Toderici [67]	Python	Tensorflow	1 (iterace)	0	768 × 512	182,262	100	1200
					1920 × 1280	1124,680	100	3500
				50	768 × 512	181,129	100	1200
					1920 × 1280	1050,972	100	3500
				100	768 × 512	185,576	100	1300
					1920 × 1280	1042,773	100	3500
Ballé [3]			1 (kvalita)	0	768 × 512	4,084	100	900
					1920 × 1280	8,253	100	1700
				50	768 × 512	3,912	100	1100
					1920 × 1280	8,430	100	1700
				100	768 × 512	5,091	100	1100
					1920 × 1280	14,570	100	2200
Lee [40]			1 (kvalita)	0	768 × 512	26,375	45	300
					1920 × 1280	128,033	45	300
				50	768 × 512	27,493	45	300
					1920 × 1280	131,045	45	300
				100	768 × 512	81,503	50	1000
					1920 × 1280	410,186	38	1000
JPEG	C++	–	–	0	768 × 512	0,294	27	300
					1920 × 1280	0,410	32	300
				50	768 × 512	0,325	29	300
					1920 × 1280	0,419	33	300
				100	768 × 512	0,386	34	300
					1920 × 1280	0,618	37	300
JPEG 2000	C++	–	–	0	768 × 512	0,418	32	300
					1920 × 1280	0,838	48	300
				50	768 × 512	0,421	34	300
					1920 × 1280	0,857	52	300
				100	768 × 512	0,428	35	300
					1920 × 1280	0,857	54	300
HEVC Intra	C++	–	–	0	768 × 512	6,890	32	300
					1920 × 1280	22,557	32	400
				50	768 × 512	16,876	32	300
					1920 × 1280	43,042	32	500
				100	768 × 512	15,540	32	300
					1920 × 1280	73,242	32	500
JPEG XL	C++	–	–	0	768 × 512	0,745	43	300
					1920 × 1280	2,158	45	500
				50	768 × 512	0,619	48	300
					1920 × 1280	1,153	51	500
				100	768 × 512	1,747	44	300
					1920 × 1280	6,790	51	500

tak dostupná paměť testovacího systému nebyla dostatečná. Pokud by nebyla problémem paměť RAM, určitě by byl určitou překážkou i výpočetní výkon testovacího systému. I v rozlišení 1920 × 1280 zabrala komprese jednoho snímku do všech požadovaných kvalit všemi kodeky přibližně 7 hodin a 15 minut. Je třeba dodat, že při použití GPU se dá předpokládat kratší doba výpočtů. Z Tabulky 5.6 by šlo z hlediska výpočetní náročnosti některých kodeků usoudit, že doba komprese je přímo úměrná počtu pixelů, a tak by se pro rozlišení 3840 × 2560 a 7680 × 5120 dostala doba výpočtů na příliš vysokou hodnotu. Z tohoto důvodu byla nejprve komprese a dekomprese na daném systému provedena jen pro 21 snímků v rozlišení 256 × 256 až 1920 × 1280. Z hlediska jejich skladby se však nic nezměnilo, jelikož zbylých 5 snímků byly pouze větší varianty snímků v rozlišení 1920 × 1280. Pro snímky ve vyšším rozlišení byla následně použita jiná konfiguraci hardwaru. Zásadní změnou je použití 16 GB DDR4 RAM na frekvenci 2133 MHz místo 8 GB DDR3 RAM na frekvenci 1600 MHz, menší změnou pak použití novějšího, i když výkonově jen nepatrně lepšího, procesoru IntelCore i5-7200U, který je stejně jako původní IntelCore i5-3230M dvoujádrový, ale je vyrobený 14 nm technologií a frekvence se pohybuje mezi základní 2,5 GHz a turbo 3,1 GHz.

Tato konfigurace umožnila provést kompresi snímků v rozlišení 3840 × 2560,

a to pro všechny kodeky do všech kvalit. Bohužel se ani s touto konfigurací nepodařilo dekomprimovat všechny snímky zakódované pomocí metody Toderici, úspěch nastal pouze pro kvality 0 až 7 z možných 0 až 15. Pro ostatní kodeky proběhla dekomprese v pořádku pro všechny kvality. Toto svědčí o vysoké paměťové náročnosti metody Toderici, což se ukázalo také již u snímků v nižším rozlišení. Vzhledem k tomu, že pro danou implementaci se pro kvalitu 7 bitový tok pohybuje od 0,9 bpp do 1,1 bpp a že experiment ukázal, že zásadní rozdíly mezi kodeky se objevují především v oblasti nízkých bitových toků do 1 bpp, bylo rozhodnuto o dostatečnosti získaných dat, tedy dat do kvality 7 pro metodu Toderici. Poslední jeden snímek v rozlišení 7680 × 5120 byl zároveň z důvodu výpočetní a paměťové náročnosti vynechán.

Nová konfigurace se podepsala také na časové náročnosti. Tabulka 5.7 ukazuje jednak porovnání rychlosti komprese a dekomprese mezi oběma konfiguracemi pro jednotlivé kodeky testované na snímcích s rozlišením 1920 × 1280 a jednak porovnání rychlosti komprese a dekomprese pro novou konfiguraci mezi snímky s rozlišením 1920 × 1280 a 3840 × 2560. Všechny hodnoty jsou udány intervalem, ve kterém se konkrétní hodnota pohybuje v závislosti na nastavené kvalitě – pro vyšší kvalitu je doba zpravidla vyšší.

Tabulka 5.7: Časová náročnost mezi konfiguracemi a pro snímky 3840 × 2560

Kodek	1920 × 1280				3840 × 2560	
	Původní konfigurace (i5-3230M, 8 GB DDR3 RAM)		Nová konfigurace (i5-7200U, 16 GB DDR4 RAM)			
	Komprese [s]	Dekomprese [s]	Komprese [s]	Dekomprese [s]	Komprese [s]	Dekomprese [s]
Toderici [67]	⟨1000;1200⟩	⟨40;840⟩	⟨340;360⟩	⟨10;290⟩	⟨3000;3600⟩	⟨40;1400*⟩
Ballé [3]	⟨8;14⟩	⟨19;35⟩	⟨6;10⟩	⟨8;13⟩	⟨16;33⟩	⟨30;150⟩
Lee [40]	⟨130;410⟩	⟨450;1460⟩	⟨70;230⟩	⟨260;810⟩	⟨300;900⟩	⟨960;4600⟩
JPEG	⟨0,4;0,6⟩	⟨0,3;1,2⟩	⟨0,3;0,6⟩	⟨0,3;1,0⟩	⟨0,6;0,9⟩	⟨1,2;3,8⟩
JPEG 2000	⟨0,8;0,9⟩	⟨0,7;0,9⟩	⟨0,7;0,9⟩	⟨0,7;1,0⟩	⟨1,3;1,4⟩	⟨1,3;2,1⟩
HEVC Intra	⟨22;75⟩	⟨0,6;1,1⟩	⟨20;68⟩	⟨0,6;1,2⟩	⟨80;260⟩	⟨1,7;4,0⟩
JPEG XL	⟨1,1;6,8⟩	⟨0,7;2,0⟩	⟨0,8;6,2⟩	⟨0,6;2,3⟩	⟨3,5;24,5⟩	⟨2,2;8,0⟩

* maximální kvalita 7

5.8 Vyhodnocení výsledků

Výsledkem úspěšného dokončení skriptů `ML_comparison.m` a `ML_metrics.m` jsou pro všechny snímky, všechny kodeky a všechna nastavení kvality hodnoty jednotlivých metrik a odpovídající bitové toky. Na základě těchto výsledků je možné vynést R–D křivky a provést porovnání hned z několika úhlů pohledu.

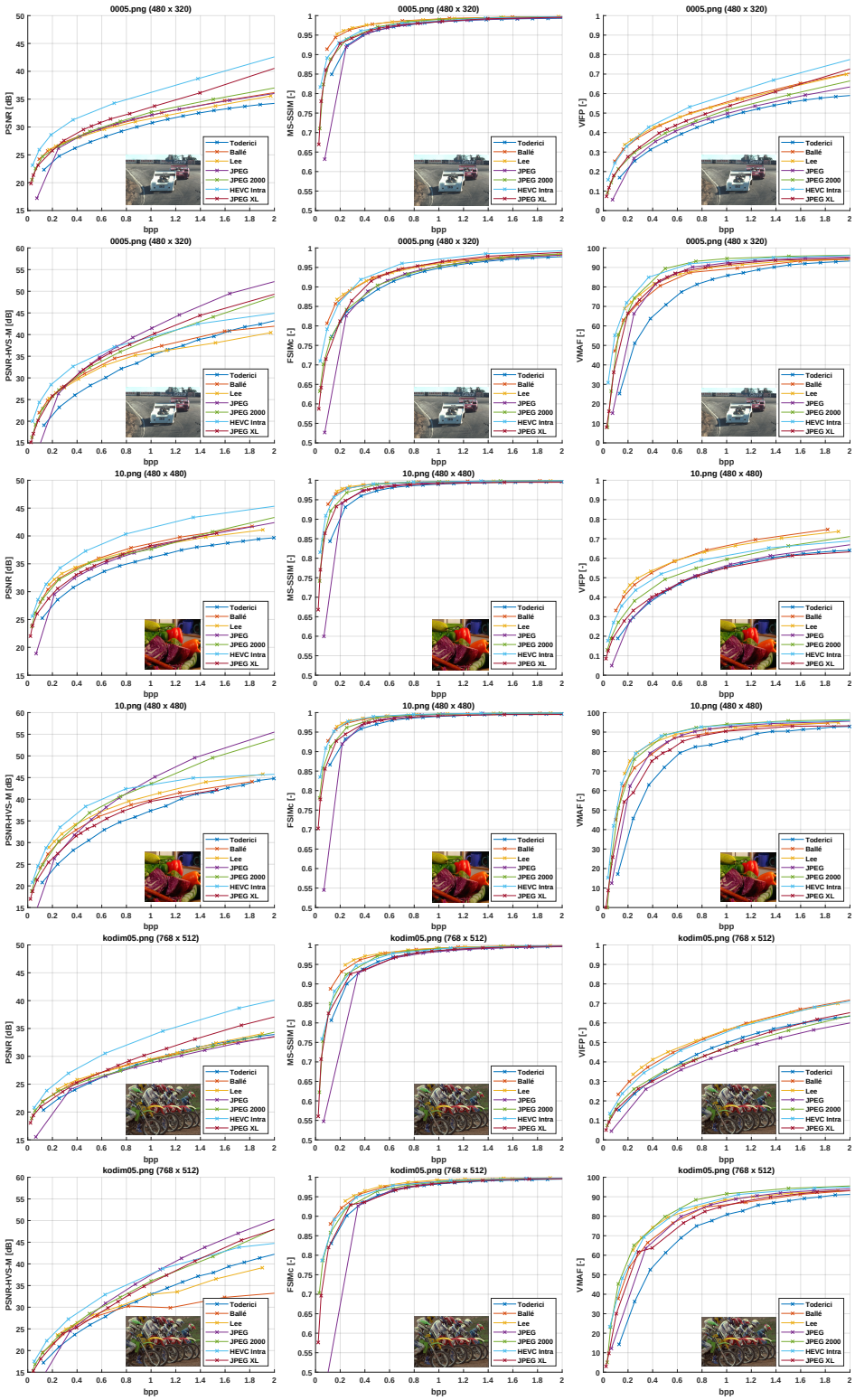
5.8.1 Kompresní účinnost

Obrázek 5.3 ukazuje výsledné hodnoty všech použitých metrik pro vybrané snímky napříč různými použitými rozlišeními v závislosti na bitovém toku, tedy dříve zmiňované R–D křivky. Výsledky pro všechny snímky lze nalézt na přiloženém CD – viz příloha A.

Z výsledků je na první pohled patrné, že kodeky založené na NN a ML jsou z hlediska kompresní účinnosti srovnatelné s konvenčními kodeky, někdy dosahují i lepších výsledků, a to z hlediska hodnocení různými metrikami. Toto platí zejména pro metody Ballé a Lee, zatímco metoda Toderici dosahuje často nejhorších výsledků ze všech použitých kodeků, a to při hodnocení většinou metrik. Výjimkou jsou pro tuto metodu úplně nejnižší cílové bitové toky (přibližně 0,1 bpp), kdy většinou překonává alespoň kodek JPEG. Podle očekávání se potvrdilo, že metoda Lee mírně překonává metodu Ballé, ze které vychází a kterou rozšiřuje o entropické kódování. Přestože byly obě metody trénovány pro optimalizaci MS-SSIM, dosahují výborných výsledků i z hlediska ostatních metrik. V další části hodnocení jsou proto při zmiňování kodeků založených na učení uvažovány metody Ballé a Lee, jejichž efektivita je výrazně vyšší než pro metodu Toderici.

Největších rozdílů si lze všimnout především při velmi nízkých cílových bitových tocích, typicky pod 0,5 bpp, kdy kodeky založené na NN a ML dosahují lepší obrazové věrnosti, tedy lepších hodnot metrik, než většina konvenčních kodeků. Největší rozdíly se objevují při použití metriky VIFP, která je hodnocená jako metrika nejlépe korelující s výsledky subjektivních testů, a tak lze říct, že viditelný poměrně velký rozdíl mezi R–D křivkami kodeků založených na učení a většiny konvenčních kodeků odpovídá subjektivně lepší vnímané kvalitě. Tuto lepší obrazovou věrnost dosáhly kodeky založené na učení podle metriky VIFP v celém rozsahu použitých bitových toků, což je výborný výsledek, který ale na druhou stranu neodpovídá závěrům, které poskytují ostatní metriky, především PSNR-HVS-M. Podle této metriky dosahují kodeky založené na učení lepších výsledků než většina konvenčních kodeků pouze pro nízké bitové toky, na vyšších jsou ale výrazně překonány konvenčními kodeky včetně základního JPEG. Výsledky hodnocení pomocí metriky FSIMc se velmi podobají výsledkům při použití metriky MS-SSIM, přestože je v nich navíc zahrnuta barevná informace. Zajímavé závěry poskytuje metrika VMAF, podle které jsou kodeky založené na učení překonány kromě kodeku HEVC Intra také kodekem JPEG 2000, a to většinou i pro nízké cílové bitové toky, navíc obecně dokáže JPEG 2000 dosáhnout nižších bitových toků. Zmíněný kodek HEVC Intra vykazuje velmi efektivní kompresi, kdy převyšuje svou efektivitou všechny ostatní kodeky především při uvažování metrik PSNR a PSNR-HVS-M, ale řadí se k nejlepším kodekům i z hlediska všech ostatních metrik – při uvažování metriky VIFP je jeho kvalita srovnatelná s kodeky založenými na učení, podle metriky VMAF se podobá kodeku JPEG 2000, vždy se tedy může rovnat s těmi nejlepšími.

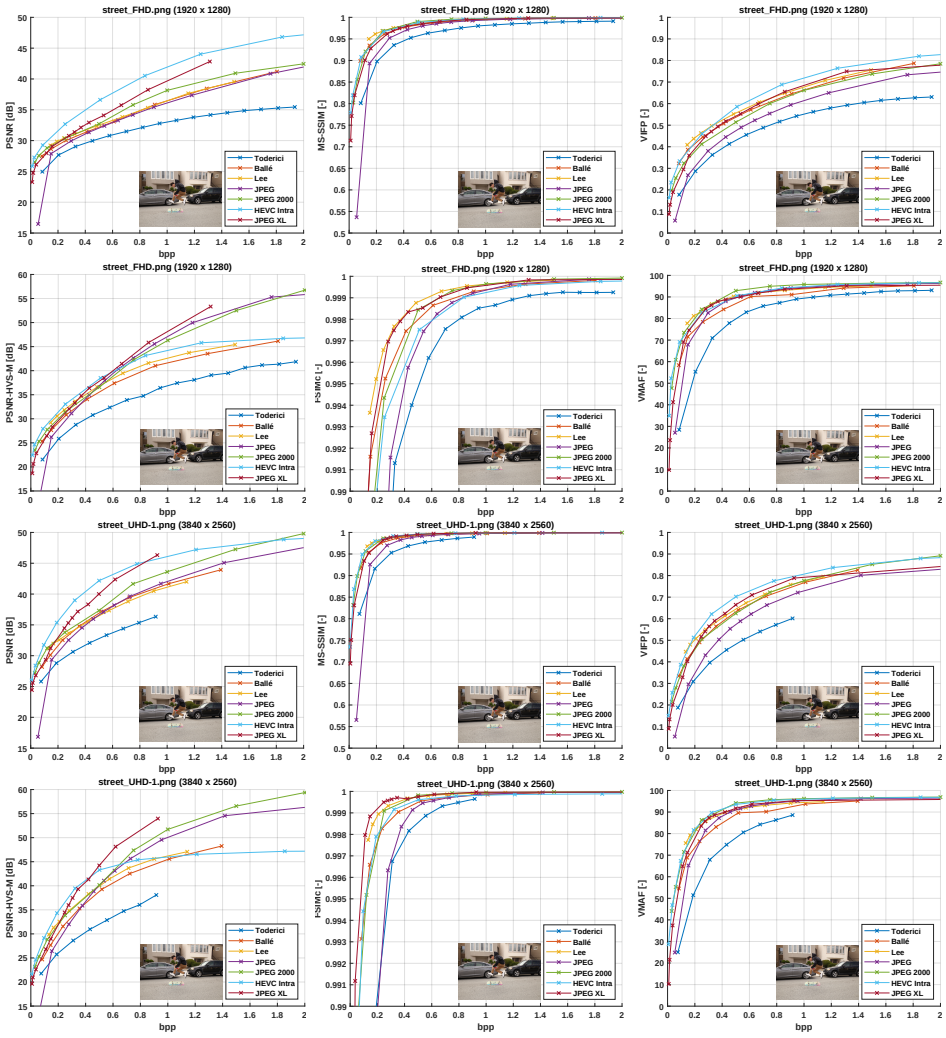
Výsledky jsou také obsahově závislé, čehož si lze všimnout při porovnání snímků s velkými jednolitými plochami a snímků s mnoha detaily. Metoda Toderici dává lepší výsledky pro snímky s mnoha detaily, například `kodim05.png`, naopak u ostatních snímků, kde jsou především jednolité plochy, podává horší výsledky i než JPEG. Naopak metody Ballé a Lee společně s konvenčním HEVC Intra si drží stabilně nejlepší výsledky pro různé typy snímků. Kodek JPEG XL vykazuje také stabilní výsledky napříč různými snímky, jeho kompresní účinnost je ale v daném nastavení dle většiny metrik průměrná.



Obrázek 5.3: Výsledné hodnoty všech metrik pro vybrané snímky

5.8.2 Dopad rozlišení na kompresní účinnost

Co se týče dopadu rozlišení testovacích snímků na kompresní účinnost, Obrázek 5.4 znázorňuje porovnání výsledků pro tentýž snímek ve dvou rozlišeních označených jako FHD (1920 obrazových bodů na řádek při poměru stran 3:2, tedy 1920 × 1280) a UHD-1 (3840 obrazových bodů na řádek při poměru stran 3:2, tedy 3840 × 2560). Z výše uvedených důvodů jsou R–D křivky pro metodu Toderici a snímky s rozlišením UHD-1 vyneseny jen do cílového bitového toku 0,9 až 1,0 bpp. Především z hlediska metrik MS-SSIM a FSIMc si lze všimnout, že pro vyšší rozlišení jsou rozdíly mezi konvenčními kodeky a kodeky založenými na učení menší. Pro metriku FSIMc bylo dokonce vhodné upravit osu grafu s hodnotami metriky pouze na rozsah 0,99 až 1, aby byly rozdíly viditelné. Z hlediska ostatních metrik se při změně rozlišení blízkost R–D křivek jednotlivých kodeků ani jejich pořadí nemění a lze tak říct, že metriky podávají stabilní výsledky napříč rozlišeními. Při porovnání R–D křivek mezi FHD a UHD-1 snímkem vždy pro stejnou metriku si pak lze všimnout, že UHD-1 snímky dosahují lepší kvality při nižší hodnotě bpp než FHD snímky. Důležitým jevem, který se zde objevuje, je, že pro takto vysoká rozlišení, především tedy UHD-1, kodeky založené na učení začínají ztrácet i v oblasti velmi nízkých bitových toků, ve které pro nižší rozlišení vynikaly. Z hlediska hodnocení většinou metrik tak vychází konvenční kodeky HEVC Intra, JPEG XL a JPEG 2000 často lépe než kodeky založené na učení, a to často v tomto zmíněném pořadí. Konvenční kodeky tak ukazují svou sílu pro snímky ve vysokém rozlišení, zatímco kodeky založené na učení jsou vhodné spíše pro nižší rozlišení. Možnou příčinou tohoto jevu může být trénování kodeků založených na učení na obrazových datech s nižším rozlišením.

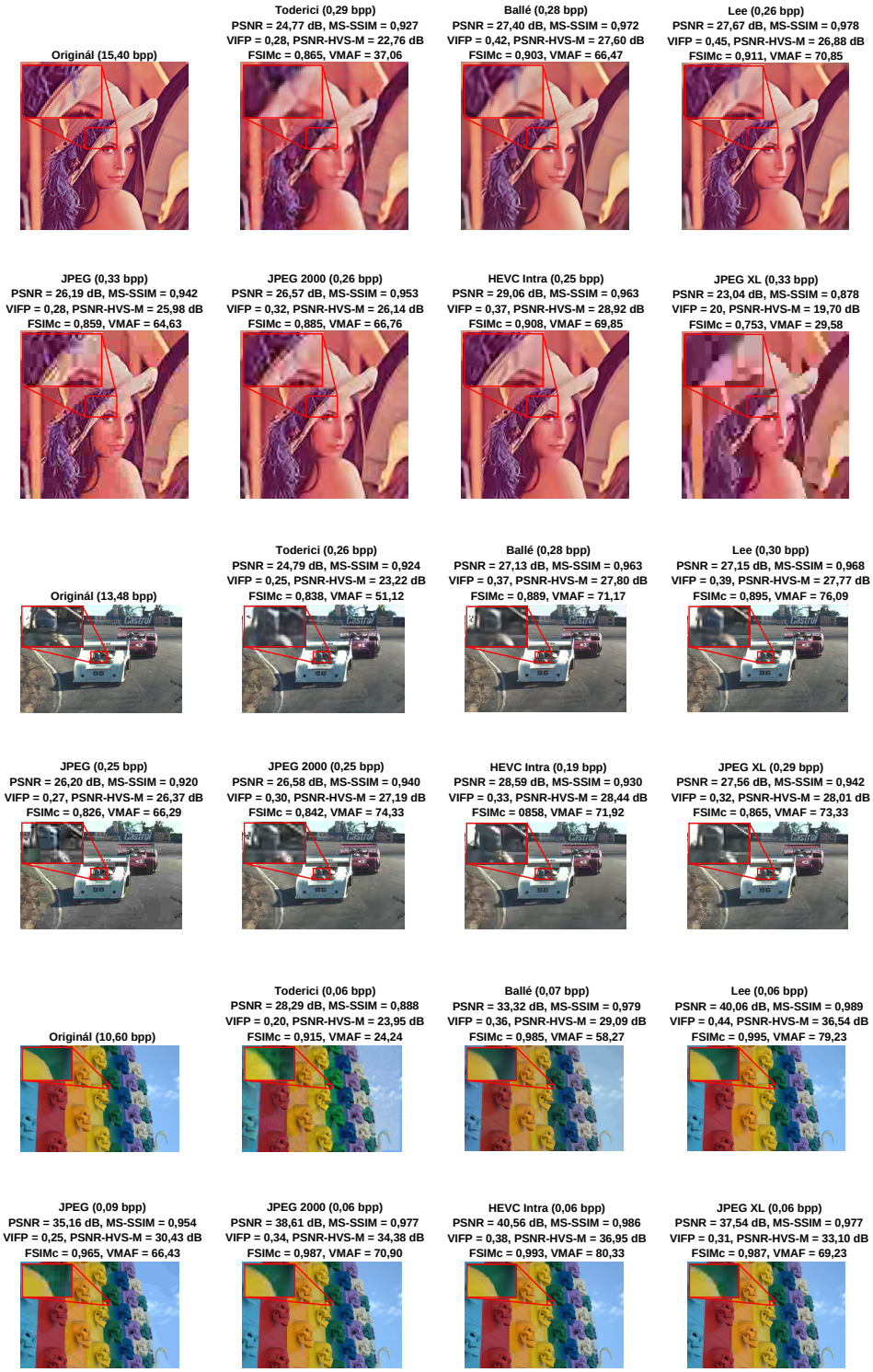


Obrázek 5.4: Porovnání všech metrik pro stejný snímek v rozlišení FHD a UHD-1

5.8.3 Kompresní artefakty

Obrázek 5.5 ukazuje pro tři různé snímky originál a komprimované varianty pro všechny použité kodeky včetně zvýraznění detailu, ze kterého si lze všimnout typických artefaktů pro jednotlivé kodeky. Do obrázku byly doplněny i konkrétní hodnoty bitového toku, kterého bylo tou kterou metodou dosaženo, a připsány i spočítané hodnoty metrik. Aby byly rozdíly mezi jednotlivými kodeky patrné, byl pro porovnání zvolen velmi nízký bitový tok, který byl u některých kodeků dosažen nastavením s nejnižší možnou kvalitou.

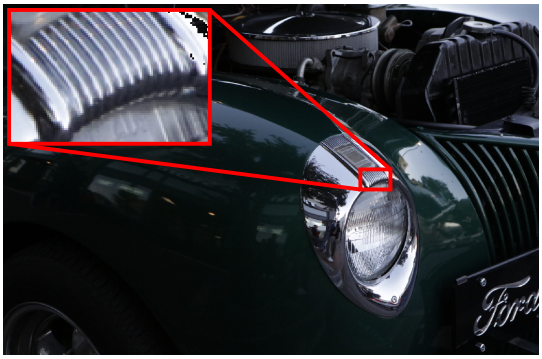
Co se týče kodeku JPEG, na všech snímcích je znatelná bloková struktura, která se projevuje především na plochách s pozvolným přechodem, je ale viditelná i v oblastech s mnoha detaily. Při zkoumání výsledků kodeku JPEG 2000 si lze všimnout typických artefaktů v podobě celkového rozmazání snímku vedoucí ke ztrátě detailů a překmitů na hranách způsobující artefakty podél hran. Z principu tohoto kodeku však nedochází ke vzniku blokových artefaktů. U kodeku HEVC Intra je v určitých oblastech znatelné zpracování v určitých blocích, nicméně celkově je kvalita výsledného snímku na velmi vysoké úrovni, snímek je ostrý a plný detailů, kodek tak patří k nejlepším. Z detailu JPEG XL jsou viditelné bloky o adaptivní velikosti, z celkového snímku pak mírné vyhlazení jednolitých oblastí. V použitém nastavení nedokáže JPEG XL dobře konkurovat nejlepším kodekům, kvalita je však rozhodně lepší než u základního JPEG. Zároveň je ale potřeba zopakovat, že kodek JPEG XL nabízí široké možnosti nastavení, a tak by se mohla jeho kompresní účinnost výrazně zlepšit po optimalizaci jeho parametrů. U kodeků založených na učení si lze všimnout některých podobných, ale i některých nových artefaktů. Podstatně jiné artefakty způsobuje metoda Toderici, v jejíchž výsledcích pro velmi nízké bitové toky se objevuje jistý vzor nebo struktura, která je nejlépe patrná na detailu třetího snímku. Tato struktura způsobuje výrazný pokles obrazové věrnosti pro takto malé bitové toky, což je vidět i na výše uvedených R–D křivkách. Dalším výrazným artefaktem je změna barevného podání u metod Toderici a Ballé, které je nejlépe viditelné na třetím snímku. V rámci metody Ballé je změna viditelná především na obloze v pozadí snímku, pro metodu Toderici jsou změny výraznější i v popředí a zdá se, že částečně souvisí se zmíněným vzorem ve snímku, který způsobuje v některých místech zvýšení saturace barev. Toto se ve snímku projevuje nestálými barvami a rušivými skvrnami s vyšší saturací na jednolitých plochách. Metoda Toderici trpí i na výrazné rozmazání celého snímku a ztrátu velkého množství detailů. Z tohoto důvodu lze subjektivně souhlasit s výsledky objektivních metrik, v rámci kterých tato metoda příliš neobstála. Co se týče metod Ballé a Lee, tak kromě zmíněného barevného zkreslení u metody Ballé jsou výsledky velmi kvalitní. Snímky působí mnohem více vyhlazené a neobsahují agresivní rychlé přechody nebo překmity. Vyhlazení zároveň není přehnané, takže nezpůsobuje přílišné rozmazání, a tak je zachováno i poměrně velké množství detailů, čehož si lze všimnout například v detailu druhého snímku. Opět lze tedy subjektivně souhlasit s výbornými výsledky těchto metod v rámci objektivních testů a je obdivuhodné, jak kvalitních výsledků dokáží tyto kodeky dosáhnout, když jde v daných příkladech o kompresní poměr 1:40 až 1:150.



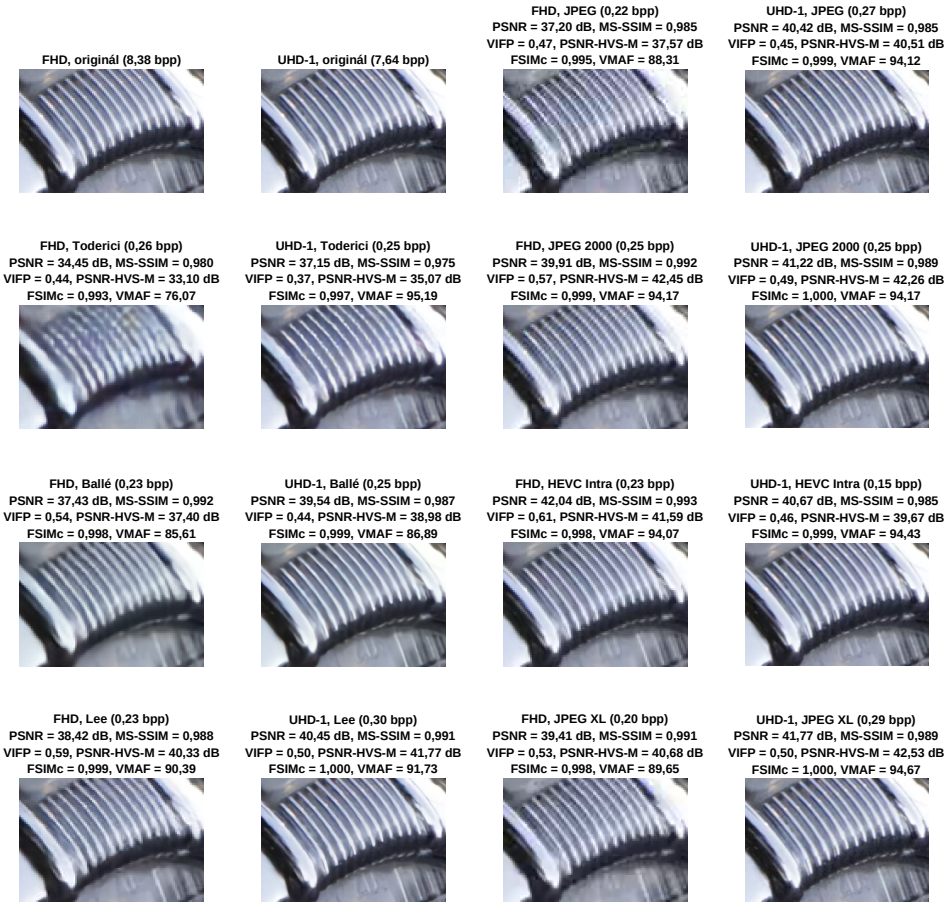
Obrázek 5.5: Porovnání originálu a komprimovaných variant

5.8.4 Dopad rozlišení na kompresní artefakty

Co se týče ukázky detailu v závislosti na rozlišení, následující obrázky porovnávají detail stejného snímku v různém rozlišení. Obrázek 5.6 ukazuje zvolený snímek se zvýrazněným detailem, v tomto případě jde o nekomprimovaný snímek v rozlišení 1920 × 1280. Obrázek 5.7 pak znázorňuje porovnání zvoleného detailu originálního a komprimovaných a dekomprimovaných snímků pomocí všech kodeků pro snímky v rozlišení FHD a UHD-1. Detaily jsou uspořádány tak, aby bylo vždy možné vedle sebe porovnat FHD a UHD-1 varianty kódované jedním kodekem. Ukázky jsou opět doplněny o informaci o dosaženém bitovém toku a hodnotách všech použitých metrik. Z daných hodnot si lze podobně jako z R–D křivek všimnout, že UHD-1 snímky dosahují lepší kvality při nižší hodnotě bpp než FHD snímky. Snímky byly vybrány s ohledem na dosažení co nejpodobnější hodnoty cílového bitového toku jak mezi jednotlivými kodeky v rámci stejného rozlišení, tak i mezi rozlišeními. Díky zvolenému nízkému bitovému toku je opět možné vidět typické artefakty pro jednotlivé kodeky. Jak bylo ale zmíněno výše, pro tato vysoká rozlišení, především UHD-1, jsou již artefakty méně znatelné a většina kodeků podává srovnatelně kvalitní výsledky. Pro všechny kodeky s výjimkou metody Toderici a Ballé jsou UHD-1 komprimované verze nerozeznatelné od originálních. Co se týče ostrosti a množství detailů, podává metoda Ballé také výborné výsledky, stále je ale přítomné barevné zkreslení. Vedle toho u metody Toderici se i v nejvyšším rozlišení projevuje pravidelná rušivá struktura, která výsledný snímek poměrně výrazně degraduje. Na FHD verzích jsou pak artefakty viditelné lépe, přičemž se shodují s výše popsanými. V tomto konkrétním případě princip kodeku JPEG, především rozdělení do bloků a fakt, že stejnosměrné složce přikládá vysokou váhu, způsobuje i mírné barevné zkreslení a ostřejší barevné přechody mezi jednotlivými bloky, než odpovídá originálnímu pozvolnému přechodu mezi zelenou a fialovou barvou ve středu detailu. Výraznější barevné zkreslení se objevuje také u metody Toderici, které je v detailu vidět objevením nažloutlé oblasti v horní části. Barevné zkreslení metody Ballé je vlivem nižšího rozlišení a větší viditelnosti artefaktů v podobě rozmazání celého snímku více rušivé než u UHD-1 verze, opět se týká celého snímku a je v porovnání se všemi ostatními snímky viditelné na první pohled. Kodek JPEG XL působí subjektivně svou kvalitou jako mezistupeň mezi JPEG a JPEG 2000, kterému je velmi podobný. Toto odpovídá i výše zmíněnému srovnání pomocí R–D křivek. Subjektivně nejlépe působí kodek HEVC Intra těsně následovaný metodou Lee, čemuž odpovídá i nejvyšší dosažená hodnota metriky VIFP. Snímek si po kódování oběma kodeky zachovává téměř všechny detaily, nepůsobí rozmazaným dojmem, netrpí barevným zkreslením nebo ztrátou kresby ve velmi světlých nebo velmi tmavých oblastech. Jedná se o velmi kvalitní výsledek, přičemž kompresní poměr je v obou případech 1:36.



Obrázek 5.6: Znázornění detailu, na kterém je provedeno porovnání



Obrázek 5.7: Porovnání detailu originálu a komprimovaných variant snímku v rozlišení FHD a UHD-1

5.8.5 Srovnání pomocí Bjøntegaardovy metriky

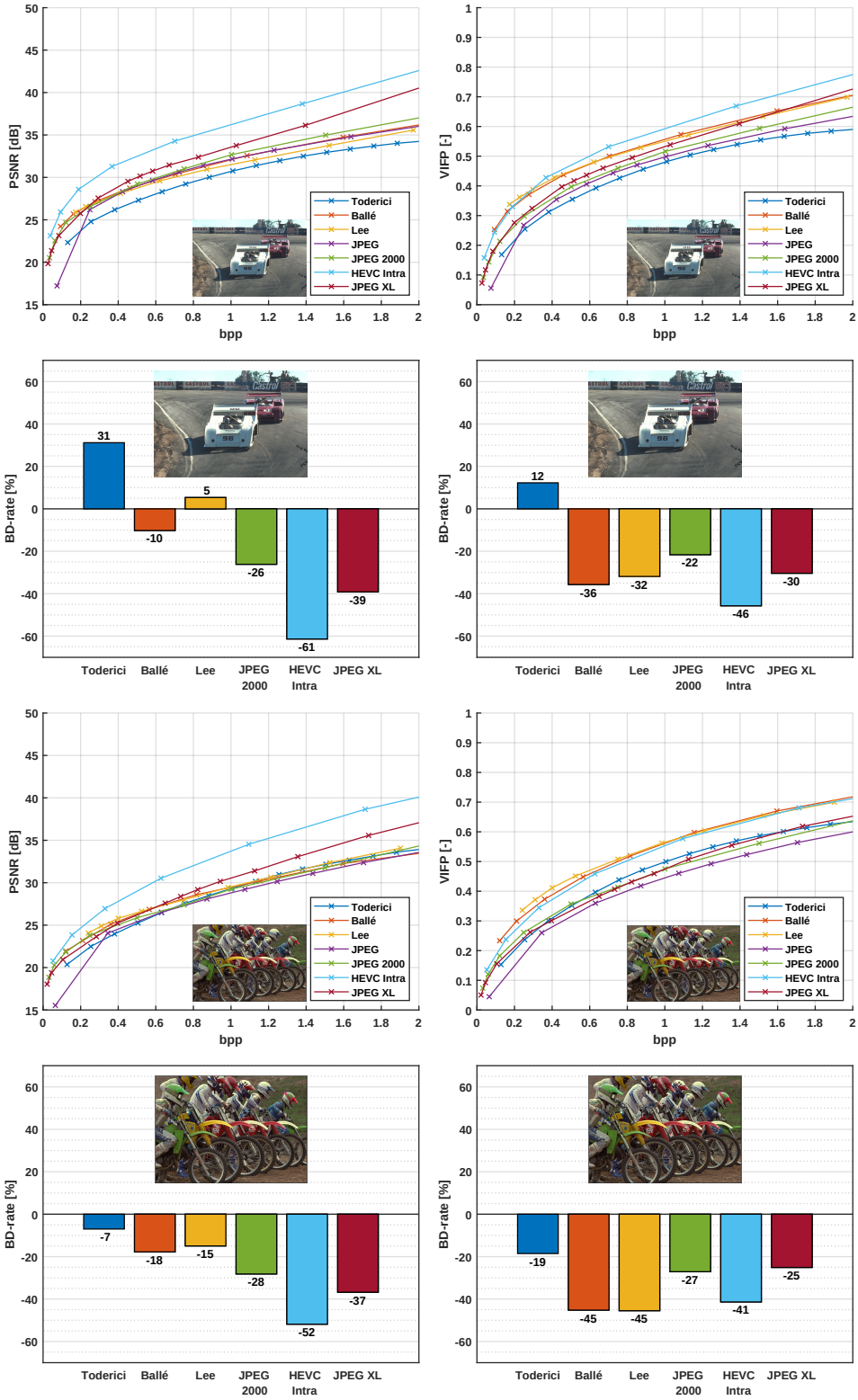
Na závěr prezentace výsledků je zde zařazeno integrální srovnání všech kodeků provedené pomocí Bjøntegaardovy metriky. Informace a implementace pro program MATLAB jsou dostupné z [4]. Tato metrika dává jako výstup srovnání dvou R–D křivek jedno číslo, a proto je pro srovnávání ideální. Alternativou může být například srovnání pomocí plochy pod R–D křivkou. Výstupem metriky je parametr BD (Bjøntegaard Delta), který má dvě možné varianty:

- BD-PSNR – průměrný PSNR zisk v dB,
- BD-rate – průměrnou úsporu bitového toku v %.

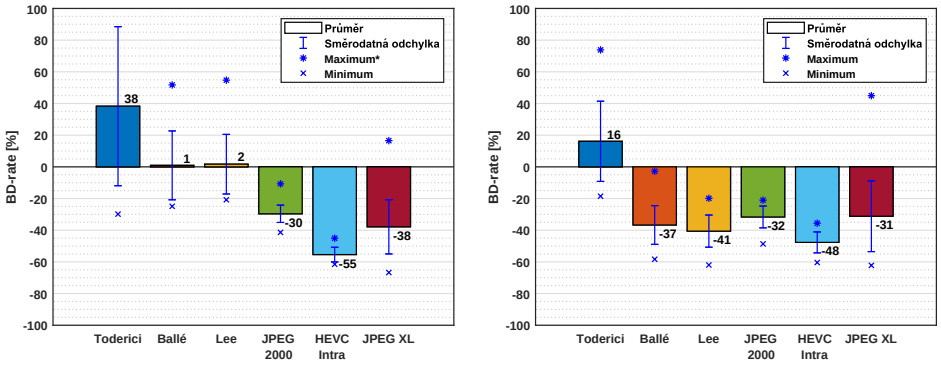
Pro experiment byla použita varianta poskytující průměrnou úsporu bitového toku vyjádřená pro každý kodek vzhledem k v současné době nejpoužívanějšímu kodeku JPEG. Takto bylo provedeno srovnání kodeků pomocí R–D křivek založených na metrice PSNR, pro kterou je Bjøntegaardova metrika původně určena, a VIFP, která je, jak bylo výše několikrát zmíněno, metrikou nejlépe korelující se subjektivními testy hodnocení kvality, a to pro všechny použité snímky.

Obrázek 5.8 ukazuje takové srovnání na příkladu dvou snímků. Výsledky pro všechny snímky a založené na všech metrikách lze nalézt na přiloženém CD – viz příloha A. Kromě samotného parametru BD-rate, jehož záporná hodnota znamená úsporu a kladná hodnota nárůst bitového toku vzhledem ke kodeku JPEG, je pro okamžité srovnání součástí také náhled R–D křivek, ze kterých závěry vychází. Tyto dva snímky byly zařazeny záměrně, kdy pro první z nich jsou kodeky založené na učení méně účinné, a pro druhý více. Výsledky pro první snímek založené na metrice PSNR ukazují velmi nízkou efektivitu metody Toderici, ale také poměrně špatné výsledky ostatních kodeků založených na učení. Při uvažování výsledků založených na metrice VIFP pak metoda Toderici stále zůstává nevhodná, metody Ballé a Lee ale překonaly JPEG 2000 a JPEG XL. Podle obou metrik vítězí kodek HEVC Intra. Odlišná situace nastala u druhého snímku, kdy všechny kodeky dosahují úspory bitového toku oproti JPEG, a to při uvažování výsledků založených na obou metrikách. Podle PSNR opět vítězí kodek HEVC Intra, který je ale překonán metodami Ballé a Lee při uvažování výsledků podle metriky VIFP.

Obrázek 5.9 pak zobrazuje průměrný BD-rate pro všechny kodeky přes všechny testovací snímky, směrodatnou odchylku, maximum a minimum. Z výsledků založených na PSNR se dá usoudit, že kodeky založené na učení jsou výrazně horší než konvenční kodeky včetně základního JPEG, nicméně tato metrika nekoreluje se subjektivními testy. Naproti tomu metrika VIFP se subjektivními testy koreluje, a proto lze výsledkům z části (b) přiložit větší váhu a považovat je za validní. Metody Ballé a Lee se tak ukazují jako velmi schopné kodeky konkurující těm konvenčním, které výrazně překonávají kodek JPEG, mírně překonávají kodeky JPEG 2000 a JPEG XL, nicméně nedosahují kvalit kodeku HEVC Intra, který je sice nejlepší ze všech konvenčních kodeků, na druhou stranu ale také dosahuje nejvyšší výpočetní náročnosti.



Obrázek 5.8: Srovnání kodeků pomocí BD-rate vzhledem k JPEG na příkladu dvou snímků



(a) Založeno na PSNR

(b) Založeno na VIFP

Obrázek 5.9: Srovnání kodeků pomocí BD-rate vzhledem k JPEG průměrně přes všechny snímky (* maximum pro metodu Toderici založené na PSNR je 184, v zobrazeném rozsahu tedy není viditelné)

5.9 Další možné rozšíření

Tento experiment by bylo možné dále rozšířit o použití různých předtrénovaných modelů v rámci kodeků, které to umožňují. Tato skutečnost se opírá o podstatu kodeků založených na učení, kdy je možné modely optimalizovat pro konkrétní metriku. Proto při nalezení metriky, která nejlépe postihuje zkreslení způsobená kodeky založenými na učení a která pak nejlépe odpovídá výsledkům subjektivních testů, je možné model optimalizovat právě pro ni a dosáhnout tak nejlepších možných výsledků. V současné době se tak jeví jako nejlepší optimalizovat modely pro metriku VIFP.

Kapitola 6

Závěr

Úvodní kapitola a první část druhé kapitoly této práce byly věnovány motivaci a teoretickým základům, které přibližují studované téma komprese obrazu založené na učení. Součástí druhé kapitoly byl i přehled aktuálního stavu využití NN a ML v kompresi obrazu. Tento přehled pokryl informace o původních snahách v této oblasti, přes nemalé množství článků až po současné aktivity v rámci skupiny JPEG.

Ve třetí kapitole práce byl podán teoretický popis Cloud Computing služeb, které mohou v oblasti strojového učení přinést odlišný přístup k řešení, a přehled dostupných nástrojů hlubokého učení, které je možné v současné době používat. V obou těchto oblastech byly popsány výhody a nevýhody dostupných řešení, přičemž nástroje hlubokého učení byly mezi sebou navíc porovnány. Na závěr této kapitoly byla zařazena zmínka o implementacích popsaných JPEG AI, které vedly k provedení experimentu.

Čtvrtá kapitola byla věnována latentní reprezentaci obrazových dat. Nejprve byl podán přehled jejího využití, opět od prvních náznaků, přes několik článků z posledních let až po aktuální aktivity skupiny JPEG. Zásadní oblastí, která by mohla mít budoucnost, je využití latentní reprezentace současně pro kompresi a některou z úloh zpracování obrazu nebo strojového vidění.

V páté kapitole, která je zároveň stěžejní částí celé práce, jelikož naplňuje její cíl, byl proveden experiment porovnávající několik konvenčních kodeků (JPEG, JPEG 2000, HEVC Intra a JPEG XL) s metodami založenými na učení. Cílem tohoto experimentu bylo zhodnocení účinnosti komprese pomocí aktuálně dostupných metod komprese obrazu založených na učení, určení, zda se dokáží svou efektivitou rovnat konvenčním kodekům, a objevení specifických artefaktů pro tyto metody, které mohou být výrazně jiné než pro konvenční kodeky. V kapitole byl popsán výběr, zprovoznění a nastavení jednotlivých kodeků, dále pak výběr a příprava vhodných testovacích dat a samotná komprese a dekomprese. Následně byla provedena analýza výpočetní a paměťové náročnosti jednotlivých kodeků. Po kompresi a dekompresi bylo vybráno několik metrik vhodných k analýze výsledných snímků, opět bylo popsáno jejich uvedení do provozu a konfigurace a nakonec byly dané metriky pro všechny kodeky a všechny kvality vypočteny a uloženy. Získané výsledky byly prezentovány v podobě R–D křivek, součástí prezentace výsledků byl také popis typických artefaktů jednotlivých kodeků, závislosti

kvality komprese na obsahu snímků nebo jejich rozlišení a nakonec bylo provedeno integrální srovnání všech kodeků vzhledem k v současné době nejpoužívanějšímu kodeku JPEG pomocí Bjøntegaardovy metriky. Jako velmi zkrácený a stručný závěr z tohoto experimentu lze říci, že kodeky založené na učení jsou konkurenceschopné, mnohdy překonávají konvenční kodeky a blíží se i kvalitám HEVC Intra, nicméně jejich slabou stránkou je vysoká paměťová náročnost a časová náročnost komprese i dekomprese. Konvenční kodeky jsou většinou navrhovány jako nesymetrické tak, aby doba dekomprese byla řádově nižší než doba komprese, u kodeků založených na učení byla ale mnohdy u testovaných metod situace opačná, kdy dekomprese byla někdy i výrazně náročnější. Je ale potřeba zmínit, že experiment nijak nehodnotil fázi učení, kvůli které by bylo kódování řádově náročnější. Také je potřeba zdůraznit, že prezentované výsledky a z nich vyvozené závěry se váží ke zvoleným implementacím kodeků, přičemž některé z nich nejsou finální, dále pak ke zvolenému nastavení parametrů kódování, zvoleným testovacím datům a metrikám.

V návaznosti na tuto práci by bylo možné dále rozšířit provedený experiment porovnání kodeků o použití různých předtrénovaných modelů optimalizovaných pro vhodnější metriky, v současnosti se nejlépe jeví metrika VIFP. Tímto by bylo možné dosáhnout kodeky založenými na učení ještě lepších výsledků. Dalším možným navázáním se jeví možnosti kolem využití latentní reprezentace současně pro kompresi a úlohy zpracování obrazu nebo strojového vidění.

Literatura

[1] 126 Amazing Social Media Statistics and Facts. Brandwatch [online]. [vid. 2020-10-07]. Dostupné z: https://www.brandwatch.com/blog/amazing-social-media-statistics-and-facts/

[2] ARIBIDO, O. J., G. ALREGIB a M. DERICHE. Self-Supervised Annotation of Seismic Images Using Latent Space Factorization. In: *2020 IEEE International Conference on Image Processing (ICIP)* [online]. 2020, s. 2421–2425. ISSN 2381-8549. Dostupné z: doi:10.1109/ICIP40778.2020.9190698

[3] BALLÉ, Johannes, Valero LAPARRA a Eero SIMONCELLI. End-to-end Optimized Image Compression. In: *International Conference on Learning Representations (ICLR)* [online]. 2016. Dostupné z: https://www.researchgate.net/publication/309737992_End-to-end_Optimized_Image_Compression

[4] *Bjontegaard metric* [online]. [vid. 2021-04-11]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/27798-bjontegaard-metric

[5] *Camera obscura* [online]. 2020 [vid. 2020-10-07]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Camera_obscura&oldid=981747738

[6] *Caffe (software)* [online]. 2020 [vid. 2021-01-10]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Caffe_(software)&oldid=983661597

[7] Caffe2. *Facebook Research* [online]. [vid. 2021-01-10]. Dostupné z: https://research.fb.com/downloads/caffe2/

[8] *Cloud computing* [online]. 2020 [vid. 2021-01-01]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=996789345

[9] *Comparison of deep-learning software* [online]. 2020 [vid. 2021-01-10]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Comparison_of_deep-learning_software&oldid=994354971

[10] CRISTOVAO, P., H. NAKADA, Y. TANIMURA a H. ASOH. Generating In-Between Images Through Learned Latent Space Representation Using Variational Autoencoders. *IEEE Access* [online]. 2020, 8, 149456–149467. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.3016313

[11] *Deep Learning Toolbox Documentation* [online]. [vid. 2021-01-07]. Dostupné z: https://www.mathworks.com/help/deeplearning/index.html

[12] DING, Deqiong, Xiaogao YANG, Fei XIA, Tiefeng MA, Haiyun LIU a Chang TANG. Unsupervised feature selection via adaptive hypergraph regularized latent representation learning. *Neurocomputing* [online]. 2020, 378, 79–97. ISSN 0925-2312. Dostupné z: doi:10.1016/j.neucom.2019.10.018

[13] DUMAS, Thierry. *Deep learning for image compression* [online]. B.m., 2019 [vid. 2020-09-29]. These de doctorat. Rennes 1. Dostupné z: https://www.theses.fr/2019REN1S029

[14] FANG, Yingying a Tieyong ZENG. Learning deep edge prior for image denoising. *Computer Vision and Image Understanding* [online]. 2020, 200, 103044. ISSN 1077-3142. Dostupné z: doi:10.1016/j.cviu.2020.103044

[15] Free Raw Photos. *Signature Edits - Improve Your Photography* [online]. [vid. 2020-11-25]. Dostupné z: https://www.signatureedits.com/free-raw-photos/

[16] Gartner Peer Insights 2020. *Deep Learning* [online]. [vid. 2021-01-07]. Dostupné z: https://blogs.mathworks.com/deep-learning/2020/07/28/gartner-peer-insights-2020/

[17] GENG, J., H. WANG, J. FAN a X. MA. SAR Image Classification via Deep Recurrent Encoding Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing* [online]. 2018, 56(4), 2255–2269. ISSN 1558-0644. Dostupné z: doi:10.1109/TGRS.2017.2777868

[18] *google/butteraugli* [online]. C++. B.m.: Google, 2021 [vid. 2021-04-26]. Dostupné z: https://github.com/google/butteraugli

[19] GU, J., Z. WANG, W. OUYANG, W. ZHANG, J. LI a L. ZHUO. 3D Hand Pose Estimation with Disentangled Cross-Modal Latent Space. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* [online]. 2020, s. 380–389. ISSN 2642-9381. Dostupné z: doi:10.1109/WACV45572.2020.9093316

[20] *Guetzli* [online]. 2021 [vid. 2021-04-26]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Guetzli&oldid=1005850898

[21] *High Efficiency Image File Format* [online]. 2021 [vid. 2021-04-10]. Dostupné z: https://en.wikipedia.org/w/index.php?title=High_Efficiency_Image_File_Format&oldid=1015260281

[22] JIANG, Feng, Wen TAO, Shaohui LIU, Jie REN, Xun GUO a Debin ZHAO. An End-to-End Compression Framework Based on Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology* [online]. 2018, 28(10), 3007–3018. ISSN 1558-2205. Dostupné z: doi:10.1109/TCSVT.2017.2734838

[23] JIANG, J. Image compression with neural networks – A survey. *Signal Processing: Image Communication* [online]. 1999, 14(9), 737–760. ISSN 0923-5965. Dostupné z: doi:10.1016/S0923-5965(98)00041-1

[24] *JPEG - 83rd Meeting – Geneva, Switzerland - JPEG explores Artificial Intelligence for Image Coding* [online]. [vid. 2020-09-29]. Dostupné z: https://jpeg.org/items/20190327_press.html

[25] *JPEG - 86th Meeting – Sydney, Australia - JPEG Committee releases a call for evidence for image compression based on AI* [online]. [vid. 2020-09-29]. Dostupné z: https://jpeg.org/items/20200217_press.html

[26] *JPEG - 89th Meeting – Online - JPEG initiates standardisation of image compression based on AI* [online]. [vid. 2020-11-14]. Dostupné z: https://jpeg.org/items/20201014_press.html

[27] *JPEG - 90th Meeting – Online - JPEG AI becomes a new work item of ISO/IEC* [online]. [vid. 2021-02-27]. Dostupné z: https://jpeg.org/items/20210205_press.html

[28] *JPEG - Call for Evidence on Learning-based Image Coding Technologies (JPEG AI)* [online]. [vid. 2020-09-30]. Dostupné z: https://jpeg.org/items/20200224_cfe_peg_ai.html

[29] *JPEG - JPEG AI - Learning-based Image Coding Common Test Conditions* [online]. [vid. 2020-11-14]. Dostupné z: https://jpeg.org/items/20201028_jpeg_ai_common_test_conditions.html

[30] *JPEG - JPEG XL* [online]. [vid. 2021-04-10]. Dostupné z: https://jpeg.org/jpegxl/documentation.html

[31] *JPEG - Performance Evaluation of Learning based Image Coding Solutions and Quality Metrics* [online]. [vid. 2020-09-30]. Dostupné z: https://jpeg.org/items/20191203_jpeg_ai_performance_evaluation.html

[32] *JPEG - Report on decisions made with respect to the subjective evaluation of the JPEG AI Call for Evidence* [online]. [vid. 2020-09-30]. Dostupné z: https://jpeg.org/items/20200812_cfe_jpeg_ai.html

[33] *JPEG - Report on the JPEG AI Call for Evidence Results* [online]. [vid. 2020-11-14]. Dostupné z: https://jpeg.org/items/20201022_jpeg_ai_cfe_report.html

[34] *JPEG - Report on the State-of-the-art of Learning based Image Coding* [online]. [vid. 2020-09-30]. Dostupné z: https://jpeg.org/items/20190402_learning_based_image_coding_report.html

[35] *JPEG-AI MMSP Challenge · Learning-based image compression* [online]. [vid. 2020-09-29]. Dostupné z: https://jpegai.github.io/

[36] KEATON, T. a R. GOODMAN. A compression framework for content analysis. In: *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL'99)* [online]. 1999, s. 69–73. Dostupné z: doi:10.1109/IVL.1999.781126

[37] KHAN, F. N. a A. P. T. LAU. Robust and efficient data transmission over noisy communication channels using stacked and denoising autoencoders. *China Communications* [online]. 2019, 16(8), 72–82. ISSN 1673-5447. Dostupné z: doi:10.23919/JCC.2019.08.007

[38] KHAROTE, P. R., M. S. SANKHE a D. PATKAR. Prostate Segmentation and Tumor Detection from MR Images Using Latent Features. *2019 IEEE 16th India Council International Conference (INDICON)* [online]. 2019, 1-4. Dostupné z: doi:10.1109/INDICON47234.2019.9030316

[39] KOUANOU, Aurelle Tchagna, Daniel TCHIOTSOP, René TCHINDA, Christian Tchito TCHAPGA, Adelaide Nicole Kengnou TELEM a Romanic KENGNE. *International Journal of Image, Graphics and Signal Processing (IJIGSP)*. 2018, 10(11), 38.

[40] LEE, Jooyoung, Seunghyun CHO a Seung-Kwon BEACK. Context-adaptive Entropy Model for End-to-end Optimized Image Compression. In *International Conference on Learning Representations (ICLR)* [online]. 2019. Dostupné z: http://arxiv.org/abs/1809.10452

[41] LEE, S. a N. U. ISLAM. Robust Image Translation and Completion Based on Dual Auto-Encoder With Bidirectional Latent Space Regression. *IEEE Access* [online]. 2019, 7, 58695–58703. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2019.2914273

[42] LI, Y., Y. LIU, Q. YAN a K. ZHANG. Deep Dehazing Network With Latent Ensembling Architecture and Adversarial Learning. *IEEE Transactions on Image Processing* [online]. 2021, 30, 1354–1368. ISSN 1941-0042. Dostupné z: doi:10.1109/TIP.2020.3044208

[43] LEIJNEN, Stefan a Fjodor VEEN. The Neural Network Zoo. *Proceedings* [online]. 2020, 47, 9. Dostupné z: doi:10.3390/proceedings47010009

[44] LIU, Xuedong a Jie YANG. Fast and High Efficient Color Image Compression Using Machine Learning. In: *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)* [online]. 2018, s. 470–473. Dostupné z: doi:10.1109/IMCEC.2018.8469518

[45] LIU, Zihao, Tao LIU, Wujie WEN, Lei JIANG, Jie XU, Yanzhi WANG a Gang QUAN. DeepN-JPEG: a deep neural network favorable JPEG-based image compression framework. In: *Proceedings of the 55th Annual Design Automation Conference* [online]. New York, NY, USA: Association for Computing Machinery, 2018, s. 1–6 [vid. 2020-09-29]. DAC '18. ISBN 978-1-4503-5700-5. Dostupné z: doi:10.1145/3195970.3196022

[46] LÓPEZ, J., A. MAURICIO a G. CÁMARA. Exploring Double Cross Cyclic Interpolation in Unpaired Image-to-Image Translation. In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* [online]. 2019, s. 124–130. ISSN 2377-5416. Dostupné z: doi:10.1109/SIBGRAPI.2019.00025

[47] MATSUDA, Ichiro, Tomokazu ISHIKAWA, Yusuke KAMEDA a Susumu ITOH. A machine learning approach to reducing image coding artifacts. In: *2017 25th European Signal Processing Conference (EUSIPCO)* [online]. 2017, s. 1485–1489. ISSN 2076-1465. Dostupné z: doi:10.23919/EUSIPCO.2017.8081456

[48] MISHRA, D., A. JAYENDRAN a P. A. P. Effect of the Latent Structure on Clustering With GANs. *IEEE Signal Processing Letters* [online]. 2020, 27, 900–904. ISSN 1558-2361. Dostupné z: doi:10.1109/LSP.2020.2996935

[49] OEHLER, K. L. a R. M. GRAY. Combining image classification and image compression using vector quantization. In: *[Proceedings] DCC '93: Data Compression Conference* [online]. 1993, s. 2–11. Dostupné z: doi:10.1109/DCC.1993.253150

[50] *ONNX | Home* [online]. [vid. 2021-01-10]. Dostupné z: https://onnx.ai/

[51] *Open Neural Network Exchange* [online]. 2020 [vid. 2021-01-10]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Open_Neural_Network_Exchange&oldid=994701693

[52] PANETTIERI, Joe, 2020. Cloud Market Share 2020: Amazon AWS, Microsoft Azure, Google, IBM. *ChannelE2E* [online]. 3. listopad 2020 [vid. 2021-01-01]. Dostupné z: https://www.channele2e.com/channel-partners/csps/cloud-market-share-2020-amazon-aws-microsoft-azure-google-ibm/

[53] PANTRAKI, E. a C. KOTROPOULOS. Face Aging as Image-to-Image Translation using Shared-Latent Space Generative Adversial Networks. In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* [online]. 2018, s. 306–310. Dostupné z: doi:10.1109/GlobalSIP.2018.8646447

[54] PATEL, Manish I., Sirali SUTHAR a Jil THAKAR. Survey on Image Compression using Machine Learning and Deep Learning. In: *2019 International Conference on Intelligent Computing and Control Systems (ICCS)* [online]. 2019, s. 1103–1105. Dostupné z: doi:10.1109/ICCS45141.2019.9065473

[55] PISTONO, Maxime, Gouenou COATRIEUX, Jean-Claude NUNES a Michel COZIC. Training Machine Learning on JPEG Compressed Images. In: *2020 Data Compression Conference (DCC)* [online]. 2020, s. 388–388. ISSN 2375-0359. Dostupné z: doi:10.1109/DCC47342.2020.00070

[56] Použití formátů HEIF a HEVC na zařízeních Apple. *Apple Support* [online]. [vid. 2021-04-10]. Dostupné z: https://support.apple.com/cs-cz/HT207022

[57] PRAKASH, Aaditya, Nick MORAN, Solomon GARBER, Antonella DILILLO a James STORER. Semantic Perceptual Image Compression Using Deep Convolution Networks. In: *2017 Data Compression Conference (DCC)*) [online]. 2017, s. 250–259. ISSN 2375-0359. Dostupné z: doi:10.1109/DCC.2017.56

[58] PUNNAPPURATH, Abhijith a Michael S. BROWN. Learning Raw Image Reconstruction-Aware Deep Image Compressors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2020, 42(4), 1013–1019. ISSN 1939-3539. Dostupné z: doi:10.1109/TPAMI.2019.2903062

[59] *PyTorch* [online]. [vid. 2021-01-10]. Dostupné z: https://www.pytorch.org

[60] QUIJAS, Jonathan a Olac FUENTES. Removing JPEG blocking artifacts using machine learning. In: *2014 Southwest Symposium on Image Analysis and Interpretation* [online]. 2014, s. 77–80. Dostupné z: doi:10.1109/SSIAI.2014.6806033

[61] RAO, Pachara V., Suhas MADHUSUDANA, Nachiketh S.S. a Kusuma KEERTHI. Image Compression using Artificial Neural Networks. In: *2010 Second International Conference on Machine Learning and Computing* [online]. 2010, s. 121–124. Dostupné z: doi:10.1109/ICMLC.2010.33

[62] SHANG, Ronghua, Lujuan WANG, Fanhua SHANG, Licheng JIAO a Yangyang LI. Dual space latent representation learning for unsupervised feature selection. *Pattern Recognition* [online]. 2021, 114, 107873. ISSN 0031-3203. Dostupné z: doi:10.1016/j.patcog.2021.107873

[63] SUO, C., Z. LIU, L. MO a Y. LIU. LPD-AE: Latent Space Representation of Large-Scale 3D Point Cloud. *IEEE Access* [online]. 2020, 8, 108402–108417. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2999727

[64] *TensorFlow* [online]. [vid. 2021-01-07]. Dostupné z: https://www.tensorflow.org/

[65] TESTOLINA, Michela, Evgeniy UPENIK a Touradj EBRAHIMI. Subjective-Objective Correlation Study on the JPEG AI Call for Evidence Results. Interní dokument JPEG ISO/IEC JTC 1/SC29/WG1M90015

[66] TIAN, Chunwei, Lunke FEI, Wenxian ZHENG, Yong XU, Wangmeng ZUO a Chia-Wen LIN. Deep Learning on Image Denoising: An overview. *arXiv:1912.13171 [cs, eess]* [online]. 2020 [vid. 2021-03-06]. Dostupné z: http://arxiv.org/abs/1912.13171

[67] TODERICI, George, Damien VINCENT, Nick JOHNSTON, Sung Jin HWANG, David MINNEN, Joel SHOR a Michele COVELL. Full Resolution Image Compression with Recurrent Neural Networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. 2017, s. 5435–5443. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2017.577

[68] VASCONCELOS, N. a A. LIPPMAN. Library-based coding: a representation for efficient video compression and retrieval. In: *Proceedings DCC '97. Data Compression Conference* [online]. 1997, s. 121–130. ISSN 1068-0314. Dostupné z: doi:10.1109/DCC.1997.581989

[69] WANI, Qasim. *QasimWani/deep-compression* [online]. Jupyter Notebook. 2020 [vid. 2021-04-15]. Dostupné z: https://github.com/QasimWani/deep-compression

[70] Which environment is better between Matlab and Python to design deep learning models? *ResearchGate* [online]. [vid. 2021-01-07]. Dostupné z: https://www.researchgate.net/post/Which-environment-is-better-between-Matlab-and-Python-to-design-deep-learning-models

[71] *Why should I choose matlab deep learning toolbox over other opensource frameworks like caffe, onnx, pytorch, torch etc? - MATLAB Answers - MATLAB Central* [online]. [vid. 2021-01-07]. Dostupné z: https://www.mathworks.com/matlabcentral/answers/421259-why-should-i-choose-matlab-deep-learning-toolbox-over-other-opensource-frameworks-like-caffe-onnx

[72] YAO, X., X. FENG, G. CHENG, J. HAN a L. GUO. Rotation-Invariant Latent Semantic Representation Learning for Object Detection in VHR Optical Remote Sensing Images. In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium* [online]. 2019, s. 1382–1385. ISSN 2153-7003. Dostupné z: doi:10.1109/IGARSS.2019.8899285

[73] YESILYURT, A. B. a F. KAMISLI. End-to-end Learned Image Compression with Conditional Latent Space Modeling for Entropy Coding. In: *2020 28th European Signal Processing Conference (EUSIPCO)* [online]. 2021, s. 501–505. ISSN 2076-1465. Dostupné z: doi:10.23919/Eusipco47968.2020.9287779

[74] YOU, C.-Z., H.-H. FAN a Z.-Q. SHU. Non-negative Sparse Laplacian regularized Latent Multi-view Subspace Clustering. In: *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)* [online]. 2020, s. 210–213. ISSN 2473-3636. Dostupné z: doi:10.1109/DCABES50732.2020.00062

[75] ZHANG, D., X. LU, H. QIN a Y. HE. Pointfilter: Point Cloud Filtering via Encoder-Decoder Modeling. *IEEE Transactions on Visualization and Computer Graphics* [online]. 2021, 27(3), 2015–2027. ISSN 1941-0506. Dostupné z: doi:10.1109/TVCG.2020.3027069

[76] ZHANG, K., W. ZUO, Y. CHEN, D. MENG a L. ZHANG. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing* [online]. 2017, 26(7), 3142–3155. ISSN 1941-0042. Dostupné z: doi:10.1109/TIP.2017.2662206

[77] ZHANG, W., Q. M. J. WU, Y. YANG, T. AKILAN a H. ZHANG. A Width-Growth Model With Subnetwork Nodes and Refinement Structure for Representation Learning and Image Classification. *IEEE Transactions on Industrial Informatics* [online]. 2021, 17(3), 1562–1572. ISSN 1941-0050. Dostupné z: doi:10.1109/TII.2020.2983749

[78] ZHANG, X., P. GAO, K. ZHAO, S. LIU, G. LI a L. YIN. Image Restoration via Deep Memory-Based Latent Attention Network. *IEEE Access* [online]. 2020, 8, 104728–104739. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2999965

[79] ZHAO, Y., J. PENG, Y. WEI, Q. PENG a Y. MOU. Multiple-Feature Latent Space Learning-Based Hyperspectral Image Classification. *IEEE Geoscience and Remote Sensing Letters* [online]. 2020, 1–5. ISSN 1558-0571. Dostupné z: doi:10.1109/LGRS.2020.3008847

[80] ZHOU, H., J. MA, C. C. TAN, Y. ZHANG a H. LING. Cross-Weather Image Alignment via Latent Generative Model With Intensity Consistency. *IEEE Transactions on Image Processing* [online]. 2020, 29, 5216–5228. ISSN 1941-0042. Dostupné z: doi:10.1109/TIP.2020.2980210

[81] ZHU, Siying, Bong-Nam KANG, a Daijin KIM. A deep neural network based hashing for efficient image retrieval. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* [online]. 2016, s. 002483–002488. Dostupné z: doi:10.1109/SMC.2016.7844612

Příloha A

CD

Na přiloženém CD se v adresáři `experiment` nachází oba vytvořené skripty pro program MATLAB s názvy `ML_comparison.m` a `ML_metrics.m` a jejich výstupy, tedy výsledky ve formě souborů `bpp.mat` a `results.mat`. Kromě těchto souborů je zde připravená také adresářová struktura vyžadovaná danými skripty pro možné zopakování experimentu a soubor `README.txt`, který obsahuje všechny důležité informace k obsahu adresářů a zprovoznění celého experimentu.

Adresář `plot_results` obsahuje další skripty pro program MATLAB s názvy `results_metrics.m` a `results_bjontegaard.m`, které umožňují prohlédnout si získané výsledky. První jmenovaný zobrazuje R–D křivky pro všechny snímky, všechny kodeky a všechny metriky, druhý pak BD-rate vzhledem k JPEG opět pro všechny snímky, všechny kodeky a všechny metriky. Kromě těchto dvou skriptů jsou v tomto adresáři ještě následující soubory, které skripty využívají:

- `results.mat` – soubor se všemi výsledky experimentu,
- `test_dataset` – adresář obsahující miniatury testovacích snímků použité při vykreslování R–D křivek,
- `bjontegaard.m` – funkce pro výpočet Bjøntegaardovy metriky (dostupná z [4]).

Zároveň je v kořenovém adresáři uložena tato diplomová práce ve formátu `.pdf` pod názvem `DP_daniel_safar.pdf`.