AI Coding Reality and Perspectives

www.huawei.com

DR. ELENA ALSHINA HUAWEI TECHNOLOGIES, GERMANY

11-12-2024

DR. ELENA ALSHINA

- *Mathematical modelling*
- *Video & Image compression & processing*
- *Neural network based algorithms*
- *HEVC/H.265, VVC/H.266, JPEG AI*

Education:

- *Master* of Physics Moscow Sate University (1995)
- *PhD* in Computer Science and Mathematical Modelling (1998)

Carrier:

- *Senior Researcher* Russian Academy Of Science (Institute for mathematical Modelling) 1998~2006
- *Associate Professor* National Research University of Electronic Technology 2000~2006
- *Principal Engineer* Samsung Electronics 2006 (Moscow), 2007~2018 (Suwon/Seoul)
- *Chief Video Scientist* Huawei Technologies (Munich) 2018-present

Positions:

- *Huawei*: Audiovisual Technology Lab Director; Media Coding Technology Lab Director
- *JVET*: Neural Network Video Compression AhG co-chair, exploration experiment coordinator
- *JPEG*: JPEG AI standardization project **chair and editor** (along with Prof. João Ascenso)

NNVC: Neural network-based video coding

AHG11 & AHG14 OF JVET (ISO SC 29 WG 5)

How NNVC work is organized?

Common training set: DIV2K, BVI-DVC, TVD.
to be extended … vimeo90K, call for training data responses (*so far only for information*)
Common test conditions:
https://vcgit.hhi.fraunhofer.de/jvet-ahg-nnvc/nnvc-ctc

Training cross-check:
after *independent re-training*
testing results within 0.1% BD-rate

No overfitting!

SADL :
Small Adhoc Deep-Learning Library
transparent implementation of NN operations
https://vcgit.hhi.fraunhofer.de/jvet-ahg-nnvc/sadl

Systematic study in Exploration Experiment:
results & reports are publicly available
SW available for MPEG & VCEG members

Complexity assessment:
kMAC/pxl, num of parameters, num of channels (input/output each layer) multiple 16, *far not perfect* ….

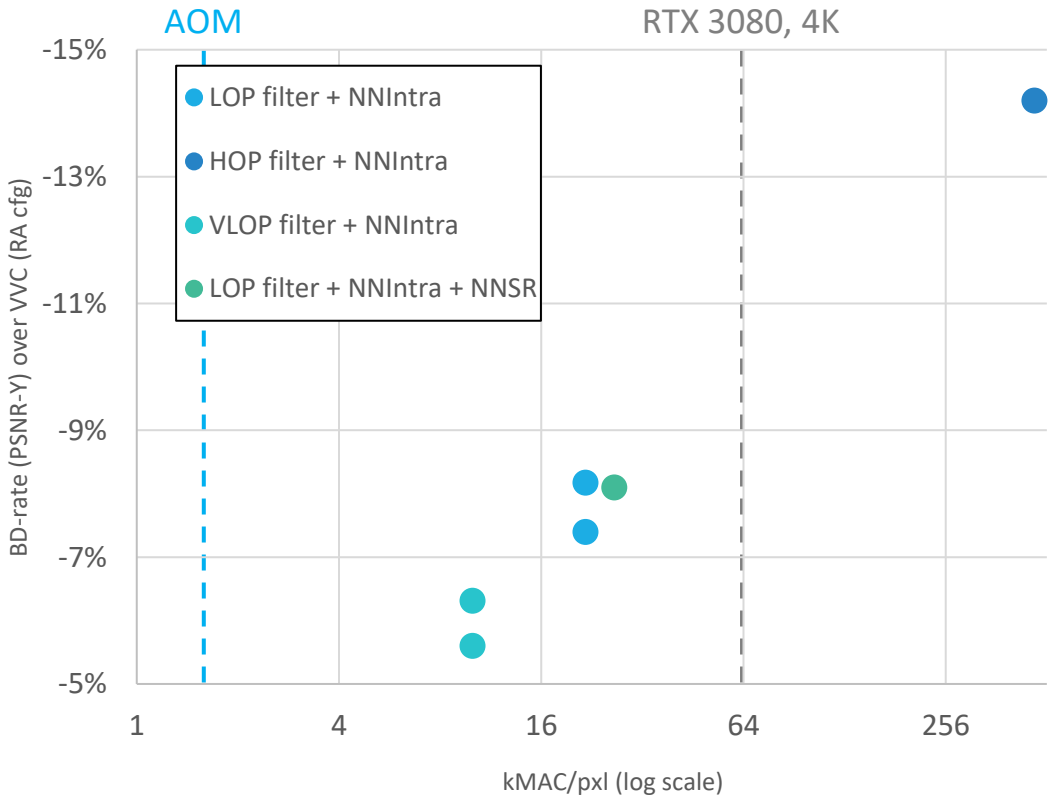
Rules: gain ↑, complexity ↓

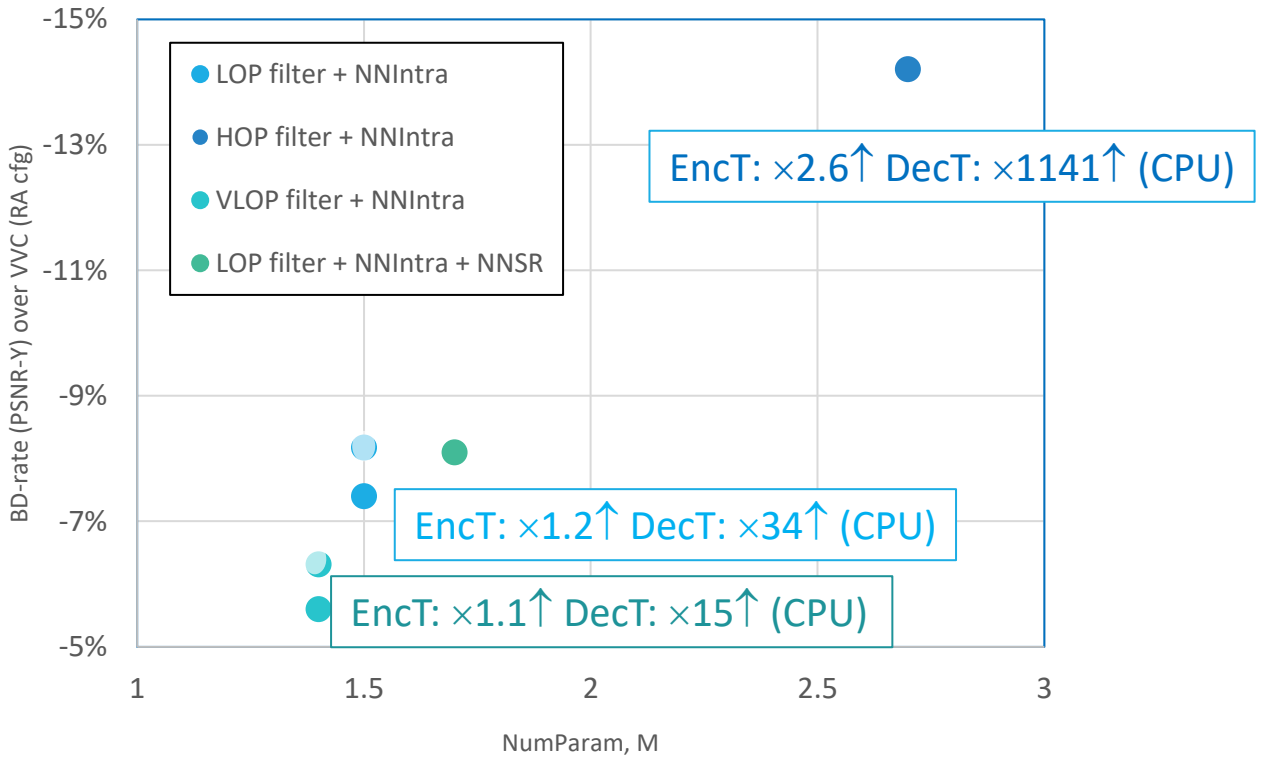
kMAC/pxl 'test' ≤ 'anchor' (***must***),
NumParam 'test ≤ 'anchor' (*desirable*),
Multiple 16 violations 'test' ≤ 'anchor' (***must***)

NNVC tools box

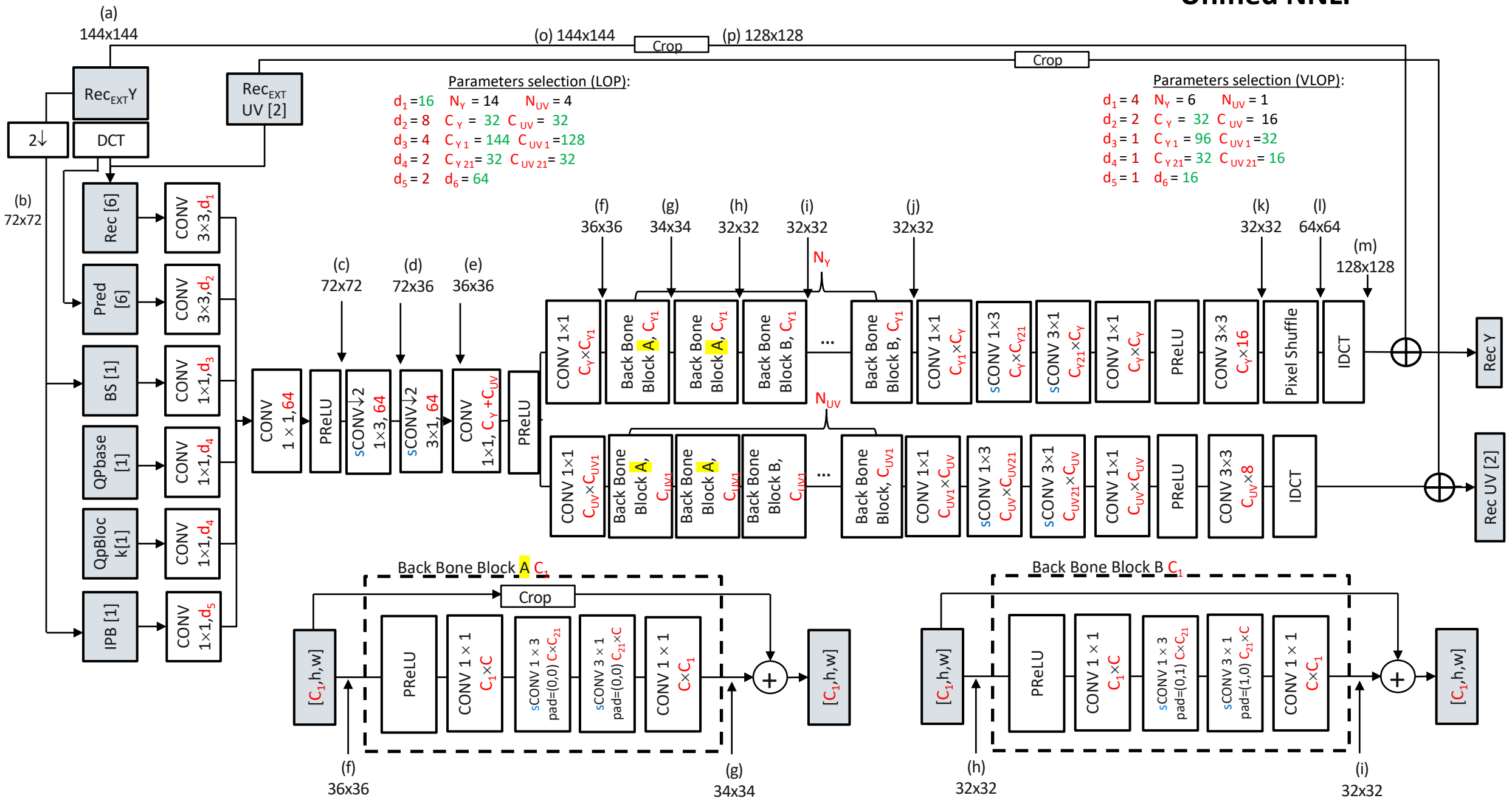
NNVC-11.0 SW tools:	kMAC/pxl	NumParam, M
NNLF – neural network-based in-loop filter		
LOP (Low Operation Point) CA-LOP content adaptive LOP	**17**	**0.2**
VLOP (Very Low Operation point) CA-VLOP content adaptive VLOP	5	0.1
HOP (High Operation Point)	466	1.4
NNIntra – neural network-based Intra	**5**	**1.3**
NNSR – neural network-based super resolution	5	0.1
NNPF – neural network-based post-filter	17	0.2

NNVC complexity performance trade-off

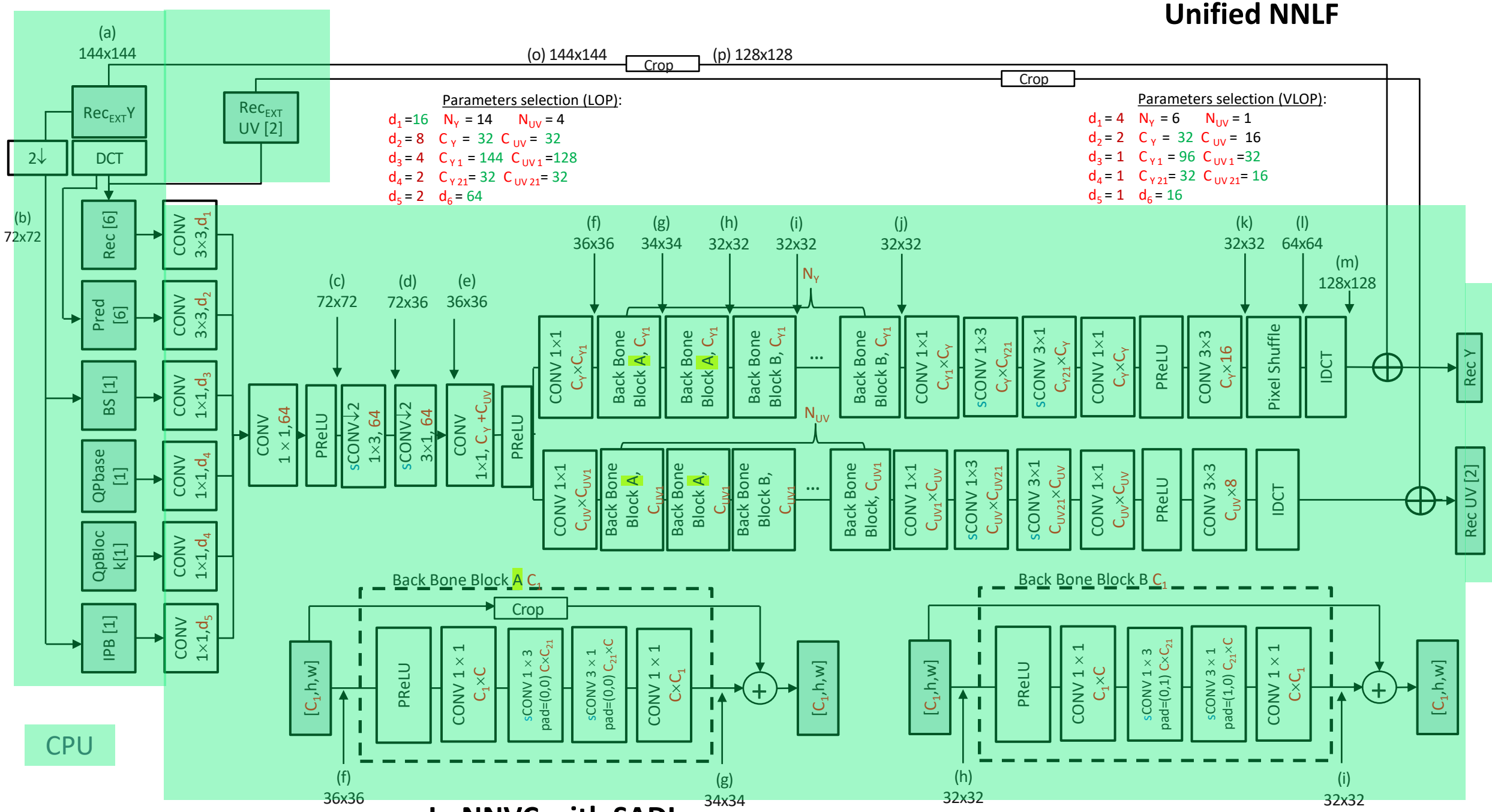




Unified NNLF

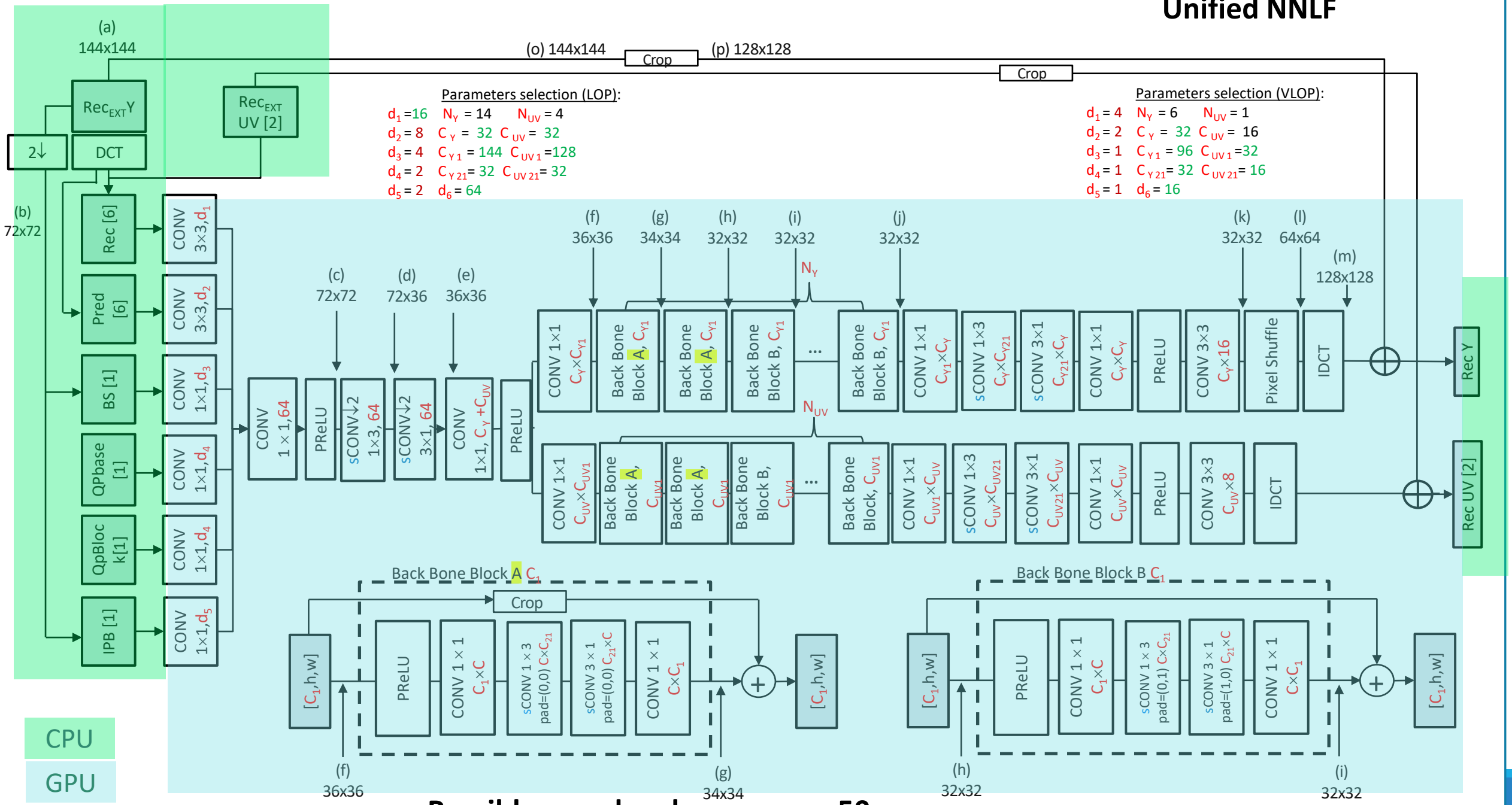


Unified NNLF



In NNVC with SADL

Unified NNLF



Possible speed up by approx. ×50

Can we really use GPU for video decoding?

Requirements:

- Bit-exact computations!!!!

How to achieve?

- Only integer operation (controlled in NNVC):
  - Training in float, inference int 16
- Overflow aware neural network quantizer (**not** controlled in NNVC)
  - Even all multipliers are int 16, accumulated register can be larger than 32 bits (overflow!!!)
- Some implementation efforts:
  - C++ → SADL (*most of us stop here*)
  - C++ → libtorch (*most of us stop here*) → CUDA (*need to go to this level to control GPU*)
  - Python → pytorch (*most of us stop here*) → CUDA (*need to go to this level to control GPU*)
- Careful selection of operations:
  - Some operations do not (yet) have integer implementation on GPU/NPU

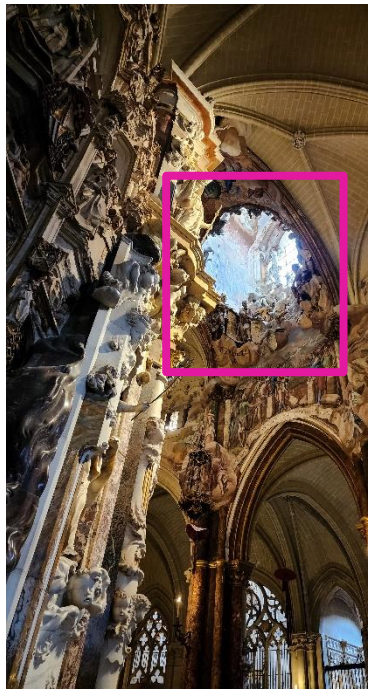
Isn't it too much? Similar to SIMD instructions (ECM, VTM SW are full of those)

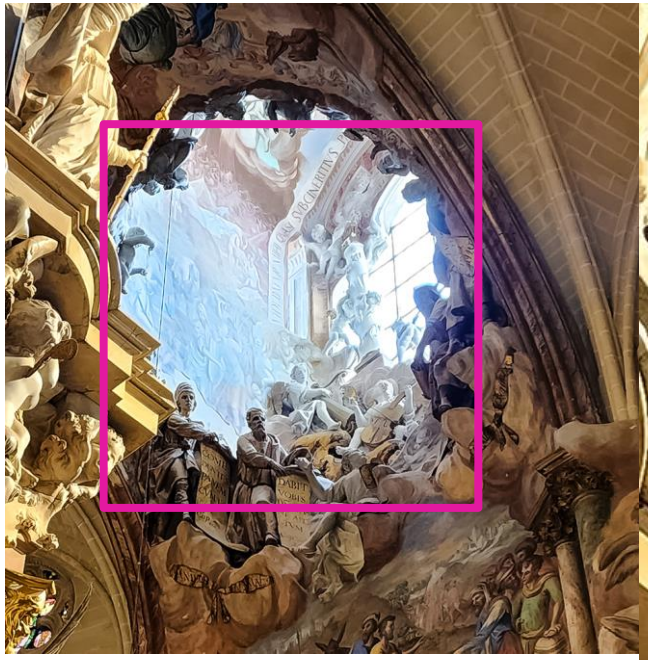
(*my estimate*) using LOP of NNVC one can get ~8% BD-rate gain, less than ×1.1EncT ↑ , less than ×1.5 DecT ↑ vs VVC

Images and video coding

SIMILAR, BUT DIFFERENT

How do we use **images** on a Smartphone?





zooming, checking for details

How do we use **video** on a Smartphone?

No zooming
focus on
actions & moving

Use on a smartphone: image ≠ video

Images

8K and higher

weak conformance (±ε)

View for 5+ seconds
power consumption :
decoder << screen

The only scenario:
Capture & encode
(zero latency)

Video

2K ≈ 4K

strong conformance (bit exact)

Decode 60+ frames per second
power consumption :
decoder ~ screen

A) Off-line encoding
(You Tube, Netflix, TikTok…)

B) Low-latency encoding

Image sharing ecosystem

sender

iPhone 13 Pro (3 bpp)

- 3024 × 4032

Samsung S23 (3 bpp)

- 6120 × 8160

Google Pixel (6 bpp)

- 3024 × 4032

optics & sensor & ISP CODEC for storage

transmission

receiver

WeChat* (0.4 bpp)

- 1280 × 1707

* *If image transmitted w/o down-sampling then down-loading takes minutes!*

KakaoTalk (0.3 bpp)

- 1530 × 2040

WhatsApp (0.3 bpp)

- 3840 × 5120

Telegram (0.5 bpp)

- 960 × 1280

standard CODEC for transmission (lower rate)

Your friend never recieves what you have sent!!!!!!

Opps, compression still needed!!! But encoding must be ultra-fast…

Why E2E AI chosen for JPEG AI?

Image size 8160×6120
Target rate 0.75 bpp

Codec (*)	CPU Encoding, s
JPEG	5
HEVC / H.265	2689
VVC / H.266	18725

(*) reference SW

5+ hours !!!!

Encoder search for modern codec comes closer to "on-line" NN training

Let's try "off-line" NN training !

Freedom VVC Intra encoder has

- *Partitioning:*
  - *W×H, W,H* ∈{4,8,16,32}
  - *Intra Sub-partition (ISP) Mode*
- *Prediction mode:*
  - *Angular Intra Prediction With 65 Angles +* Planar + DC
  - *Wide-Angle Intra Prediction (WAIP)*
  - *Matrix-Based Intra Prediction (MIP)*
  - *Position Dependent Prediction Combination (PDPC)*
  - *Multiple Reference Line (MRL) Prediction*
  - *Cross Component Linear Model (CCLM)*
- *Transform:*
  - *DCT-II, DCT-VIII, DST-VII (vertical != horizontal)*
  - *Secondary transform (LFNST8 or 4)*
- Filters
  - *SAO, ALF, CCALF.*
- Quality metric
  - *Standard was developed using PSNR (MSE)*
  - ***Actual encoder can use another metric*** *(even slower)*

JPEG AI performance evaluation

TEST CONDITIONS, ADOPTION CRITERIA

JPEG AI milestones

Overview of existing Technologies

2019

Q-metrics selection

2020

CfE
Call for Evidence

Exploration Experiments

2021

CfP
Call for Proposals

2022

Decoder on smartphone

2023

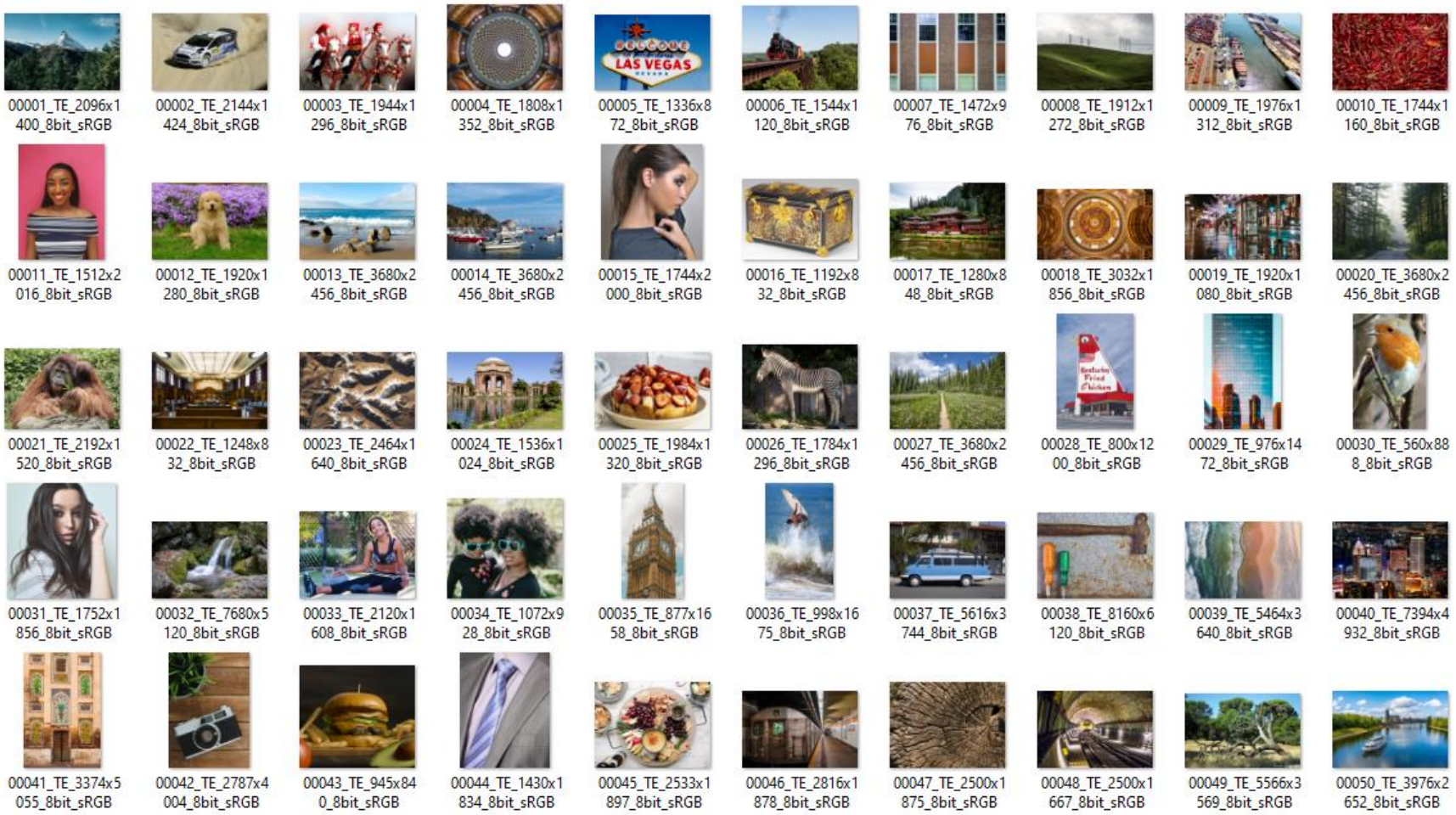
DIS
Draft International Standard

2024

IS
International Standard

2025

How to ensure against overfitting?



JPEG AI Test Set:
50 camera captured images

!=

Training Set:
120K+ patches from 5000+ camera captured images
17K+ synthetic patches from screen content and generated

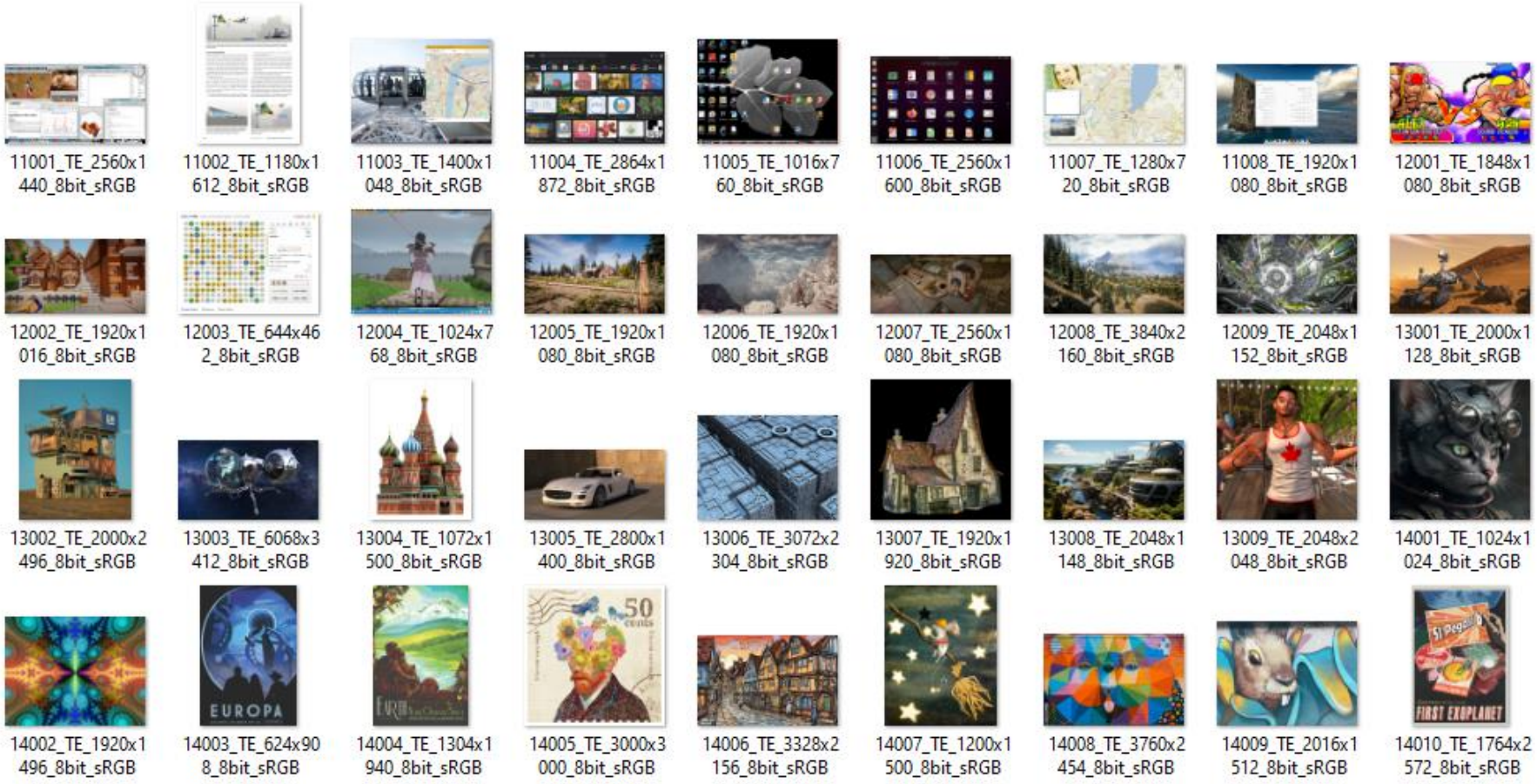
Validation Set:
350+ images

Not enough

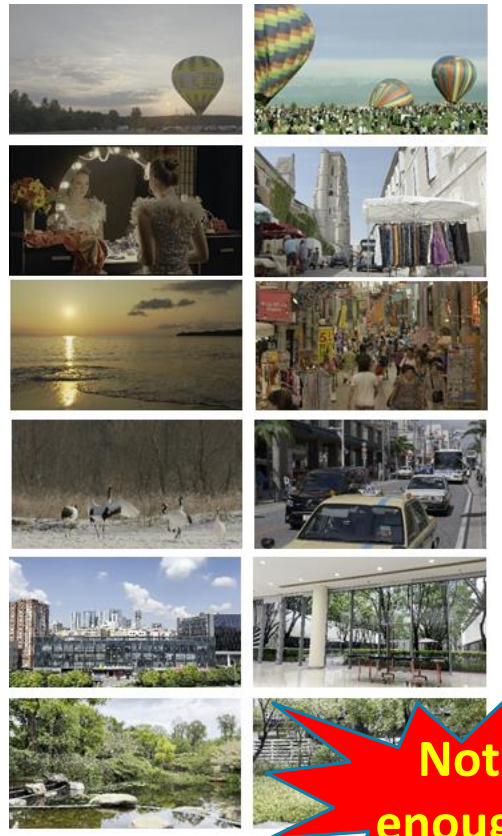
For camera captured content 7 metrics (MS-SSIM, VIF, VMAF, IW-SSIM, FSIM, psnrHVS, NLPD) measured, BD-rate computed

JPEG AI additional Test Sets

36 synthetic images



12 HDR images

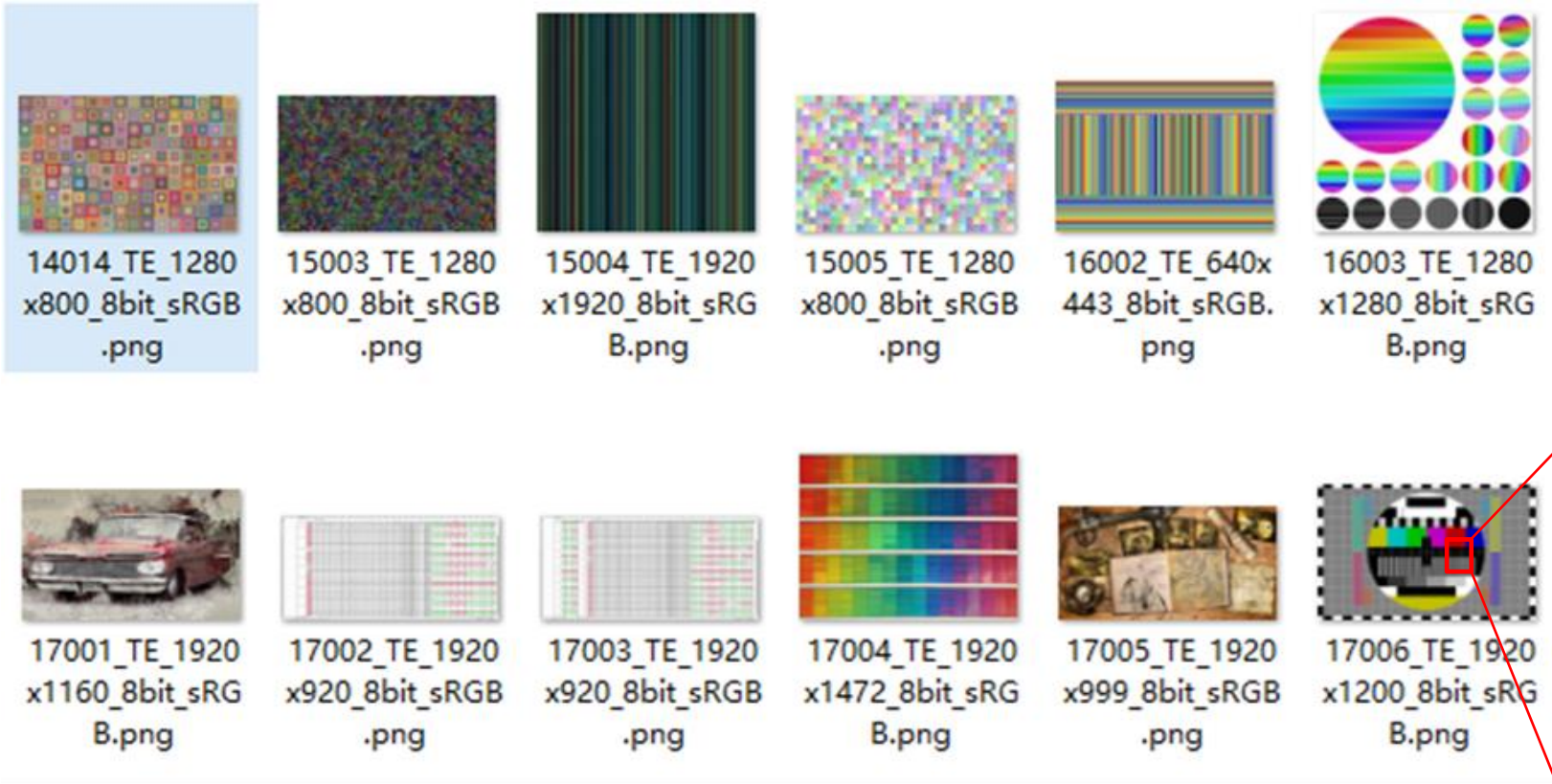


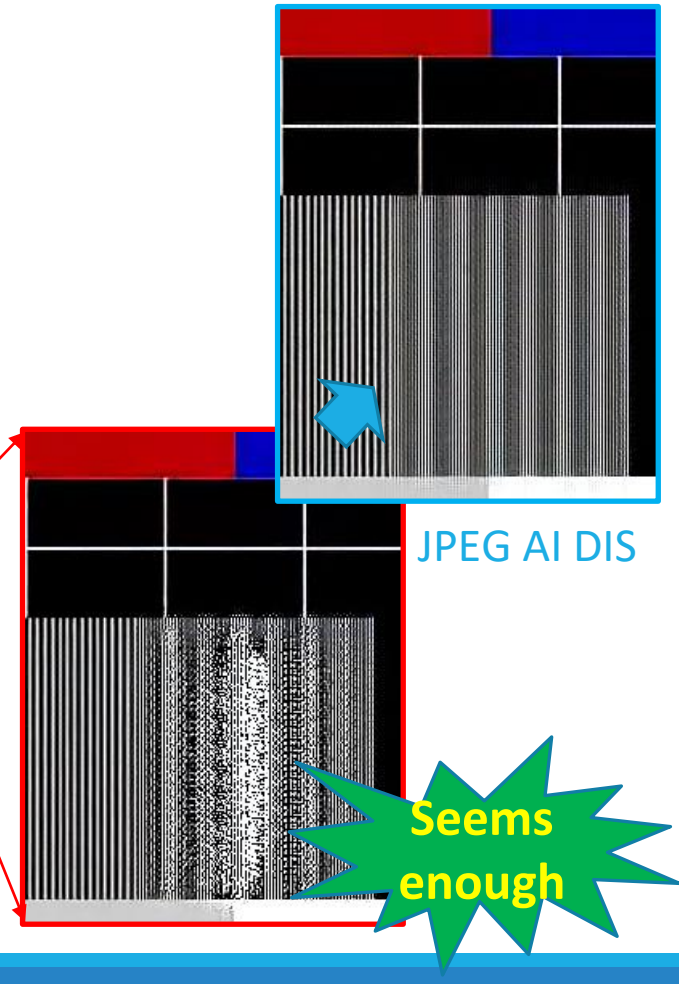
Not enough

For synthetic data: 7 metrics & visual tests, for HDR 7 metrics, HDR VDP-3 & visual tests

JPEG AI **crash** test set

12 very challenging images





How to ensure learnable codec standard is safe to use?

Common training set:
CC0 license

Common test conditions:
https://jpeg.org/jpegai/documentation.html

Unified metrics computation:
https://gitlab.com/wg1/jpeg-ai

Training cross-check:
after *independent re-training*
testing results within 0.1% BD-rate

No direct adoption:
All tentatively adopted proposals integrated and re-trained by SW coordinators →
Integration CE

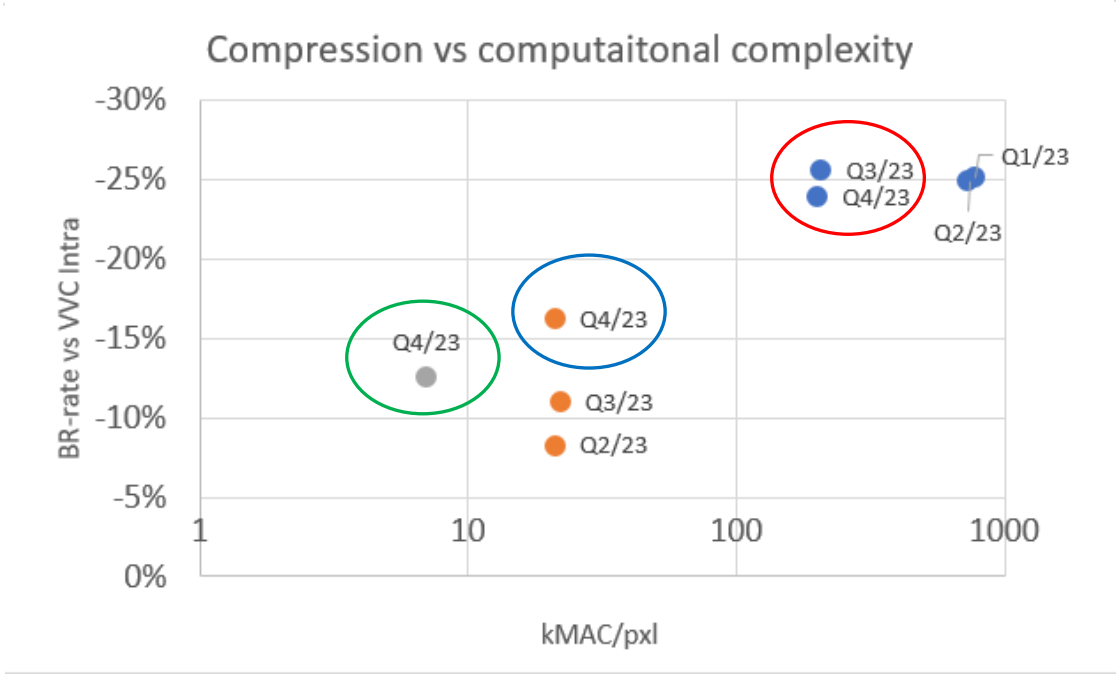
Cross-platform:
CPU, GPU, NPU, Raspberry Pi - cross platform encoding & decoding

Device interoperability check:
Strict mathematical proof of bit-exact computations in entropy part (*overflow aware NN quantizer*)

Smartphone implementation:
For Huawei (with SnapDragon) and iPhone

Mobile device run time 'prediction':
Based on profiling data for majority of commonly used NN operations (detailed!) the tool predicting the run time on mobile device created

JPEG AI evolution



JPEG AI – DIS vs VVC Intra

	BD-rate	kMAC /pxl	× DecT GPU	× DecT CPU	× EncT GPU
VM6-Ecn0Dec0	**-12.0%**	8	0.4	1	0.0004
VM6-Ecn0Dec1	**-16.7%**	23	0.4	2	0.0005
VM6-Ecn1Dec2	**-24.0%**	214	0.6	28	0.0010

Enc0Dec0 Target CPU decoding -> SIMPLE@main

Ecn0Dec1 Target Smartphone NPU decoding -> BASE@main

Enc1Dec2 Target GPU decoding -> HIGH@main

Storage scenario (visually lossless)

HEIF COMPRESSED IMAGE. TOTAL SIZE OF THE IMAGE IS 661 KB

×2…3↓

JPEG-AI IMAGE. TOTAL SIZE OF THE IMAGE IS 272 KB.

Image sharing though messengers

WECHAT TRANSFERRED IMAGE WITH SCALING AND REENCODING. TOTAL SIZE OF THE IMAGE IS 213 KB

DEMO!

JPEG-AI IMAGE WITHOUT RESCALING. TOTAL SIZE IS 272 KB.

Visual quality analysis

Enc1Dec2-tools off 0.25bpp

ORIG

VVC 0.5 bpp

Visual quality analysis

Enc1Dec2-tools off 0.5bpp | ORIG | VVC 0.5 bpp

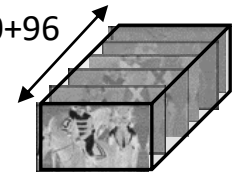

0.25 bpp
MS-SSIM = 0.9766
PSNR-Y = 33.6
VMAF = 81.6

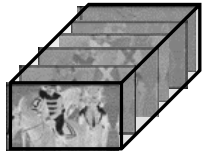

0.26 bpp
MS-SSIM = 0.9890
PSNR-Y = 34.2
VMAF = 86.3

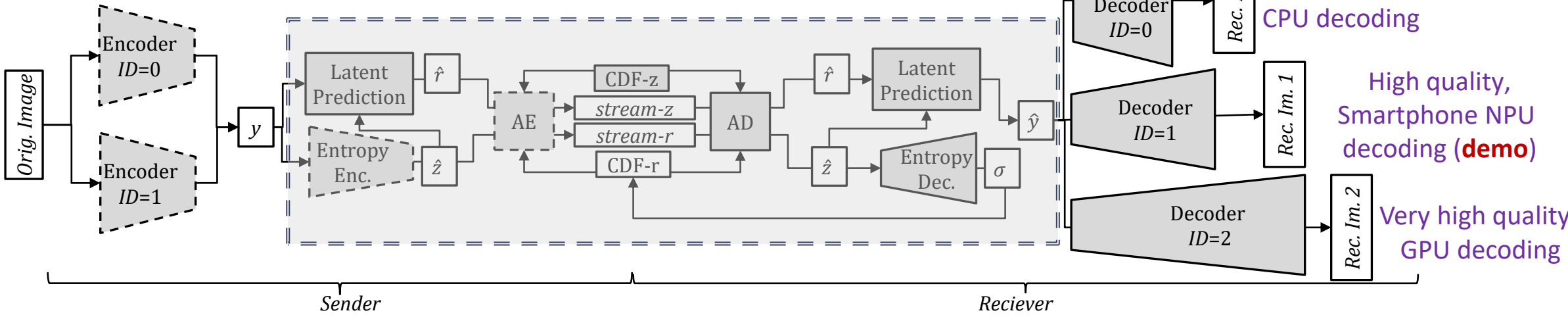
JPEG AI design

PRINCIPLES

JPEG AI: single stream multiple decodes







Good quality, CPU decoding

High quality, Smartphone NPU decoding (**demo**)

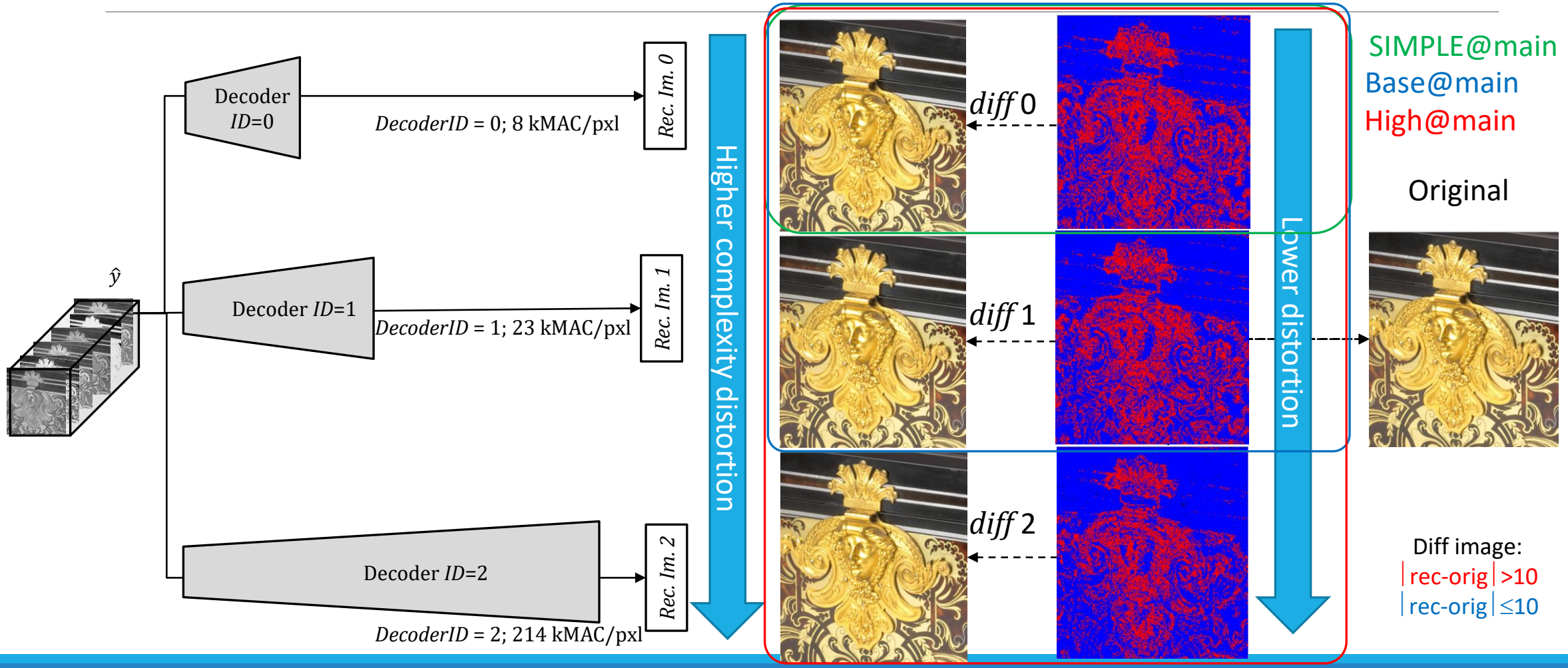
Very high quality GPU decoding

Run time: **encoder0** JPEG AI (GPU) = JPEG (CPU)

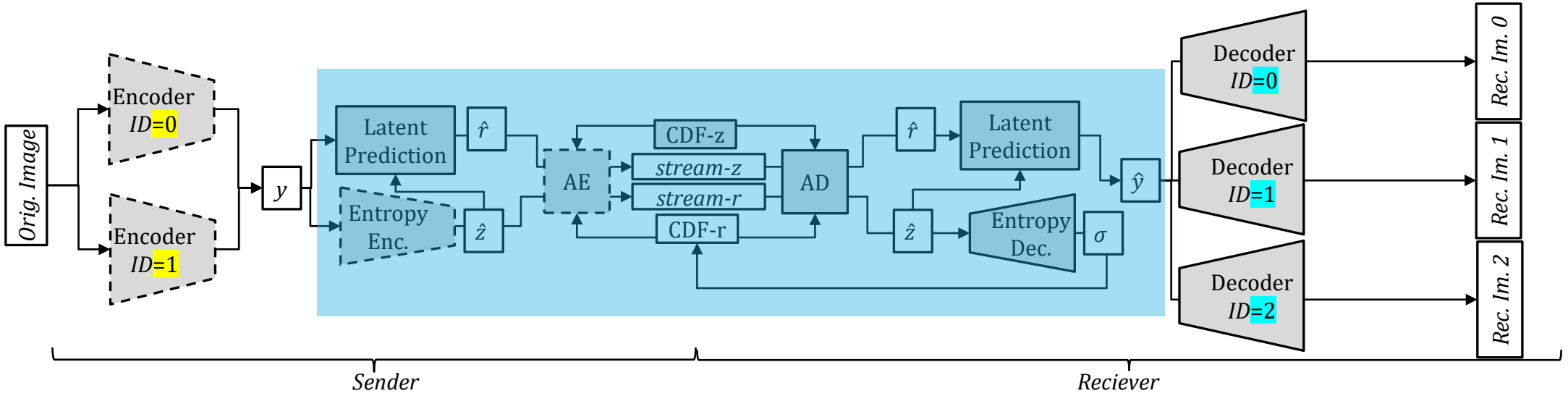
encoder1 JPEG AI (GPU) = 1/2000 · VVC/H.266 (CPU)

decoder1 JPEG AI (GPU) = ½ · H.266 (CPU)

JPEG AI: single stream multiple decodes

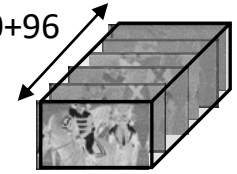


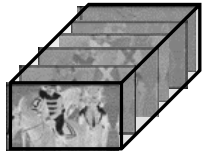
JPEG AI: single stream multiple decodes

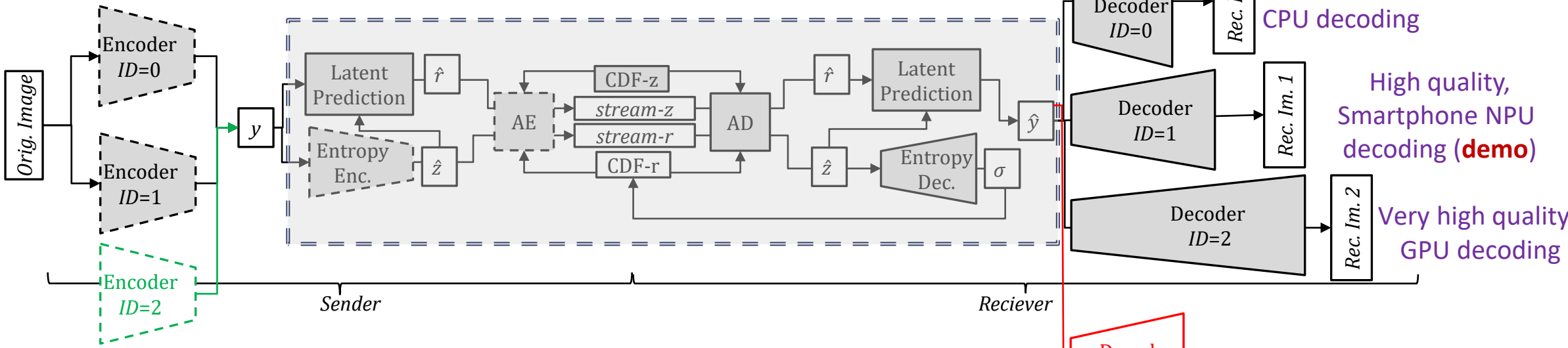


How is it possible?... Just train ‘multi-legs’ NN.

JPEG AI: what else can we do?







Good quality, CPU decoding

High quality, Smartphone NPU decoding (**demo**)

Very high quality GPU decoding

Can one add one more ‘leg’? Sure!

Can one overfit encoder for each particular image? Of cause!

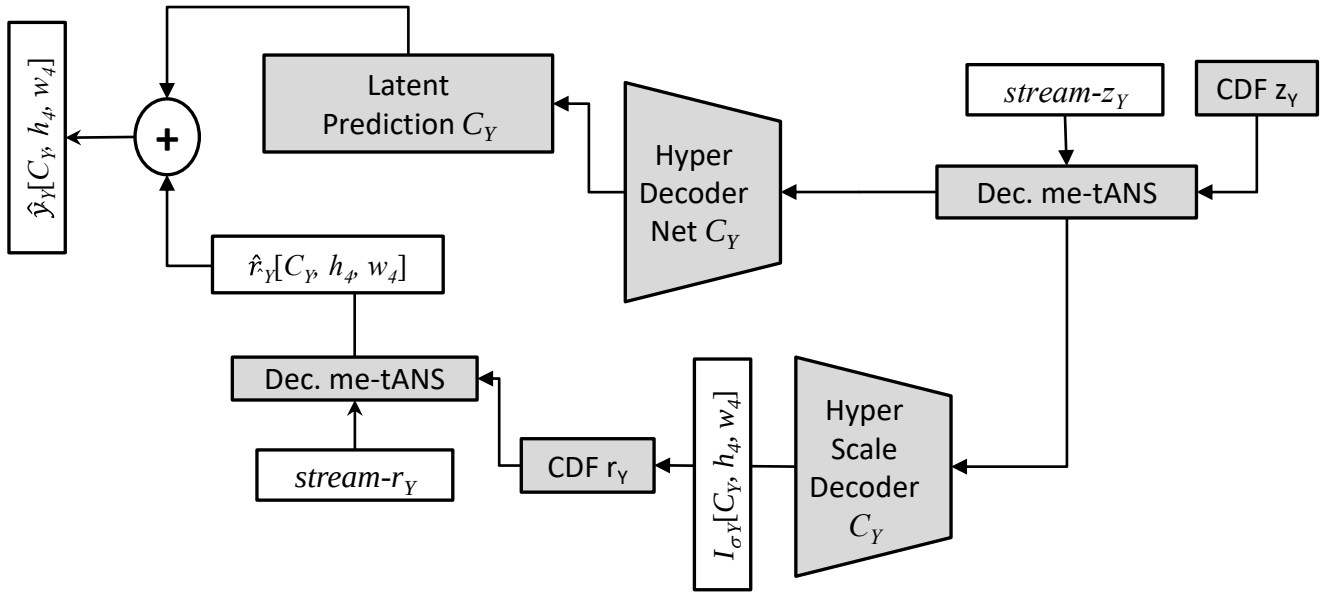
Extendable

Object detection
image classification
HDR to SDR conversion

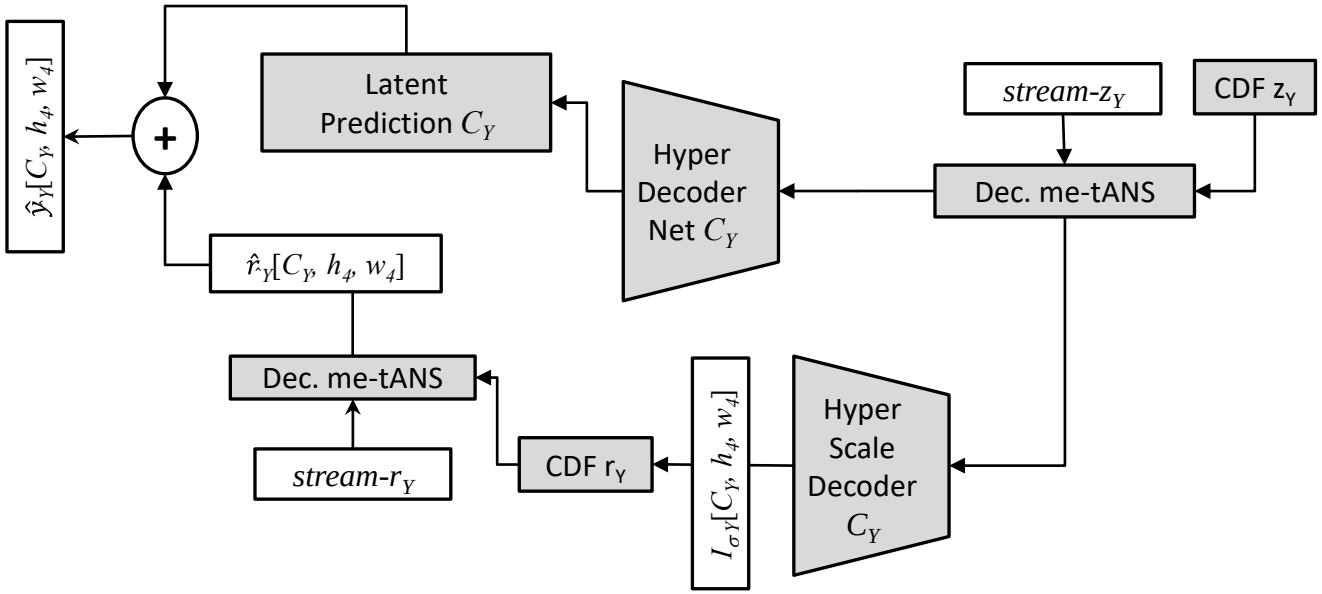
JPEG AI decoder

STEP BY STEP

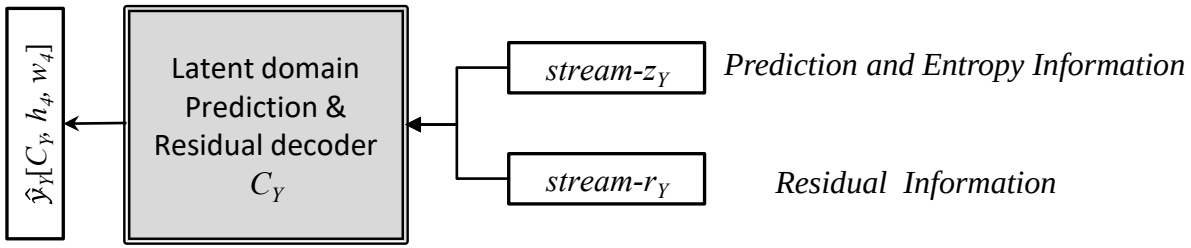
High level decoder diagram



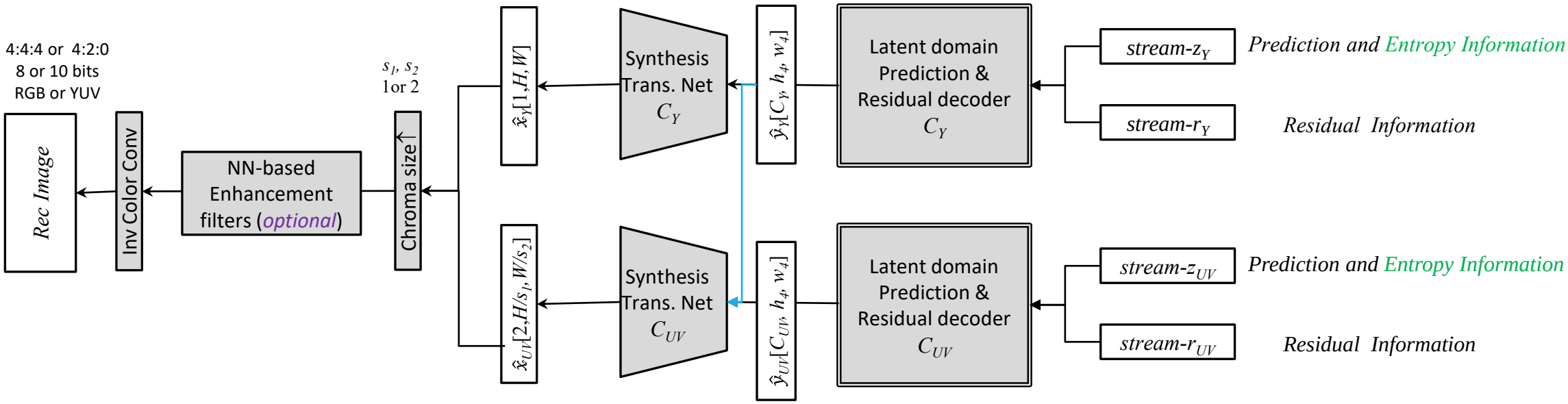
High level decoder diagram



High level decoder diagram

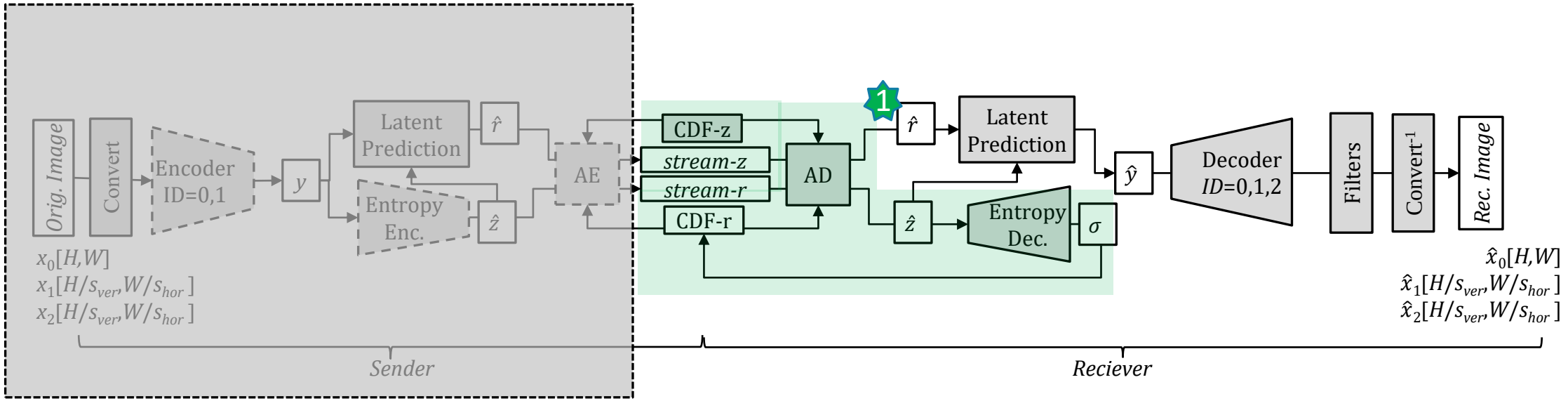


High level decoder diagram



No ‘CANAC initialization’ and from let-top to right –bottom context modeling in entropy part. Entropy parameters for each residual tensor element derived locally from z-stream

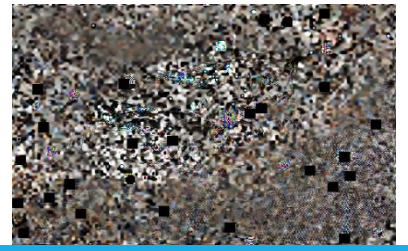
JPEG AI STRONG conformance



In case of violation:

Encoded decoded on same device

Encoded decoded on different devices

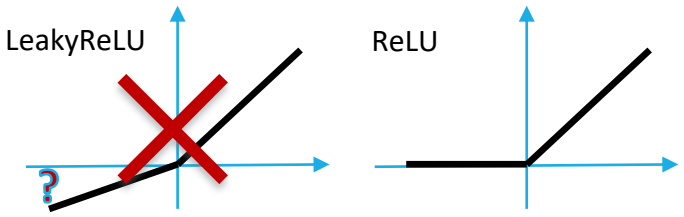


Bit.-exact behaviors required!!!

- Only integer
- Overflow aware NN quant
- Only controllable operations

Entropy network with bit-exact behavior

$\hat{z}[C, h_6, w_6]$
$qCONV(1\times1, C, C, d_1, p_1)$
$ReLU()$
$qCONV(3\times3, C, C, d_2, p_2)$
$ReLU()$
$qCONV(1\times1, C, 4\cdot4\cdot C, d_3, p_3)$
$Shuffle\ (4,4)$
$Crop(h_4, w_4)$
$abs()$
$clip(0, ((N_\sigma - 1) << sigmaPrecision)) - 1)$
$I_\sigma[C, h_4, w_4]$



convolution layer $CONV$

…

$$out[c_{out}, i, j] = bias[c_{out}] + \sum_{c_{in}=0}^{C_{in}} weigth[c_{in}, c_{out}] \star input[c_{in}, s \cdot i, s \cdot j];$$
$$i = 0, \dots, h_{out} - 1; j = 0, \dots, w_{out} - 1; c_{out} = 0, \dots, C_{out} - 1$$

where "$\star$" is 2D **cross-correlation operator** with kernel size $K_{ver} \times K_{hor}$

….

quantized convolution layer $qCONV$

… three-steps operation:

$$temp[c_{in}, i, j] = clip(-d, d-1, input[c_{in}, i, j]),$$
$$i = 0, \dots, h_{in} - 1; j = 0, \dots, w_{in} - 1; c_{in} = 0, \dots, C_{in} - 1;$$

$$R[c_{out}, i, j] = bias[c_{out}] + \sum_{c_{in}=0}^{C_{in}-1} weigth[c_{in}, c_{out}] \star temp[c_{in}, s \cdot i, s \cdot j];$$

where "$\star$" is 2D **cross-correlation operator** with kernel size $K_{ver} \times K_{hor}$.

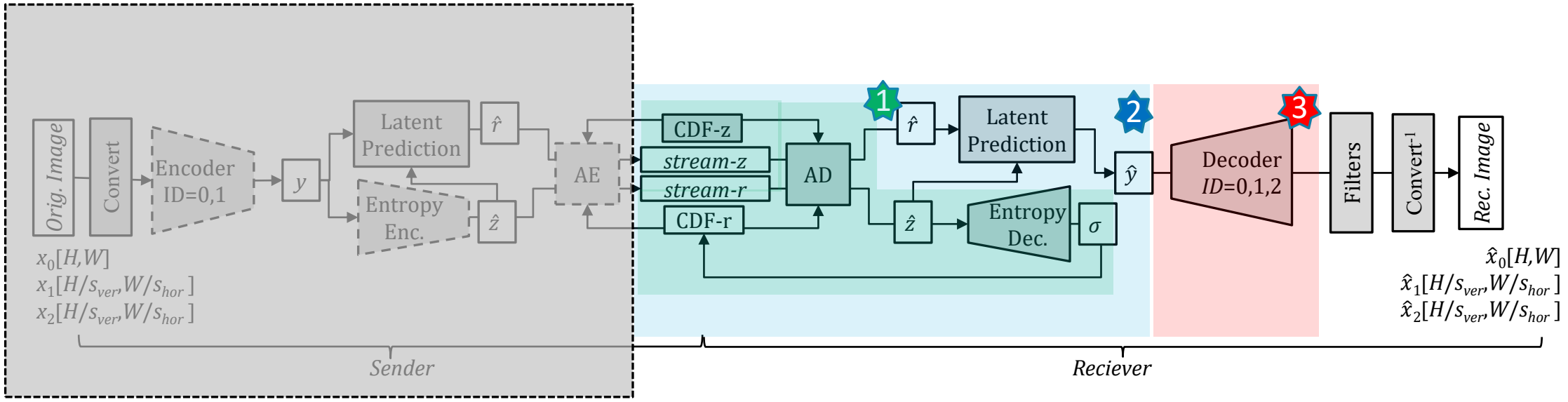
$$out[c_{out}, i, j] = (R[c_{out}, i, j]) \gg p[c_{out}];$$
$$i = 0, \dots, h_{out} - 1; j = 0, \dots, w_{out} - 1; c_{out} = 0, \dots, C_{out} - 1.$$

The tensor $weigth$ of shape $[C_{in}, C_{out}, K_{ver}, K_{hor}]$ contains learnable ***8-bit integer*** weights, the tensor $bias$ of shape $[C_{out},]$ contains learnable ***31-bit integer*** biases. All parameters $weigth$ and $bias$ are part of learnable quantized model.

The combination of clipping value *d*, de-scaling shifts $p[c_{out}]$ and magnitude for the quantized model parameters allows control over bit depth of register $R[c_{out}, i, j]$ (guaranteed to be within ***32 bits***).

…

JPEG AI weak conformance

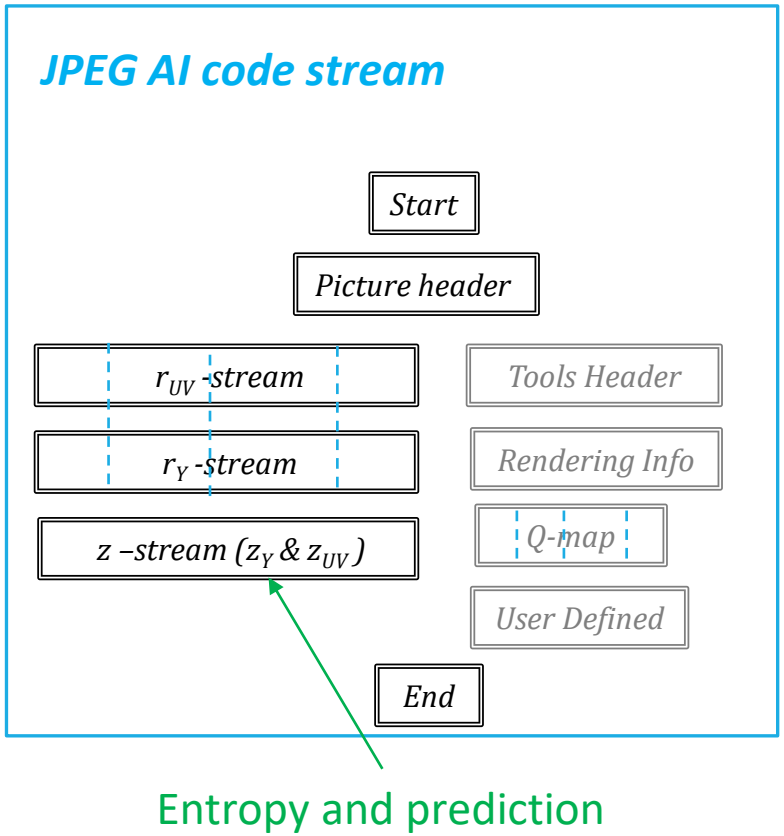


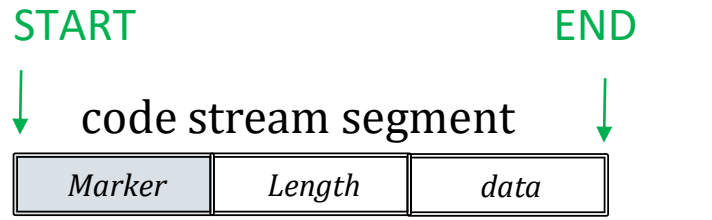
Strong: Bit.-exact behaviors required!!!
- Only integer
- Overflow aware NN quant
- Only controllable operations

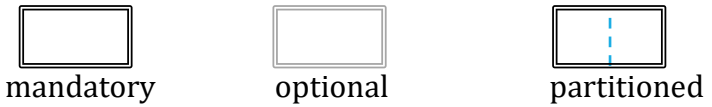
‚**Stronger**‘ conformance point after merging prediction and residual in order all ‚profiles‘, (including future extension) recieve the same ***tensor representaiton for image***

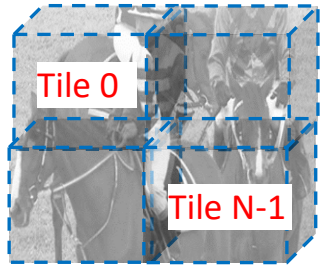
‚**Weaker**‘ conformance point after synthesis transfrom allows custom quantizer for NN

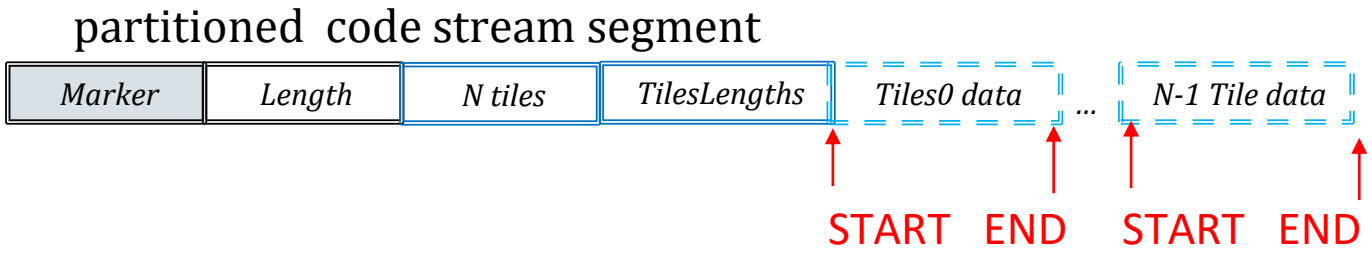
JPEG AI stream structure





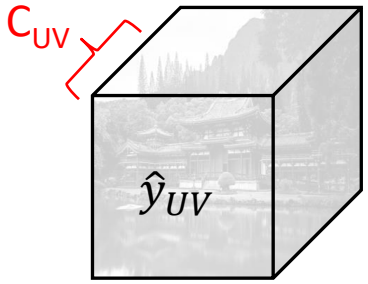




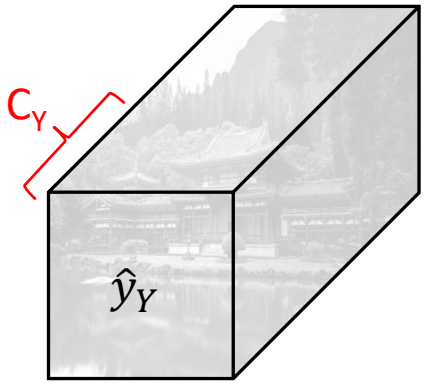


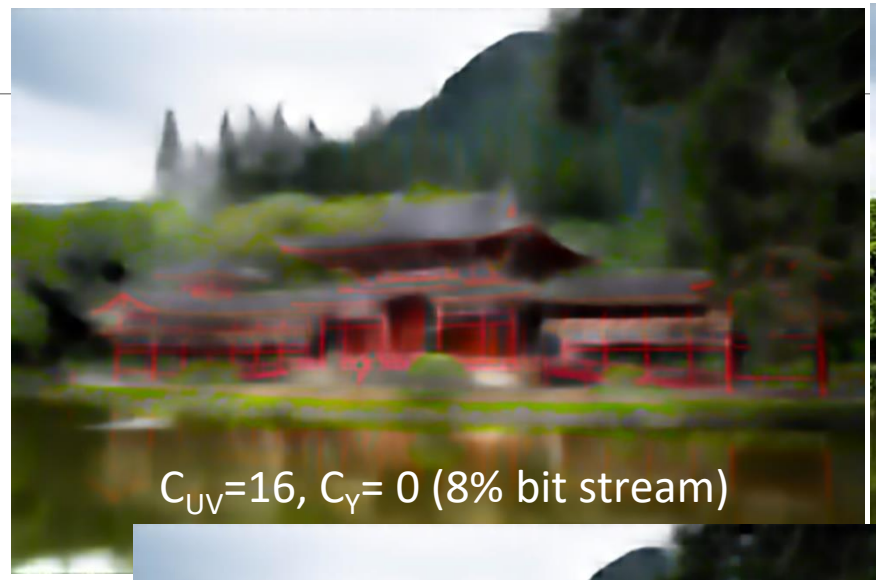
JPEG AI: progressive decoding

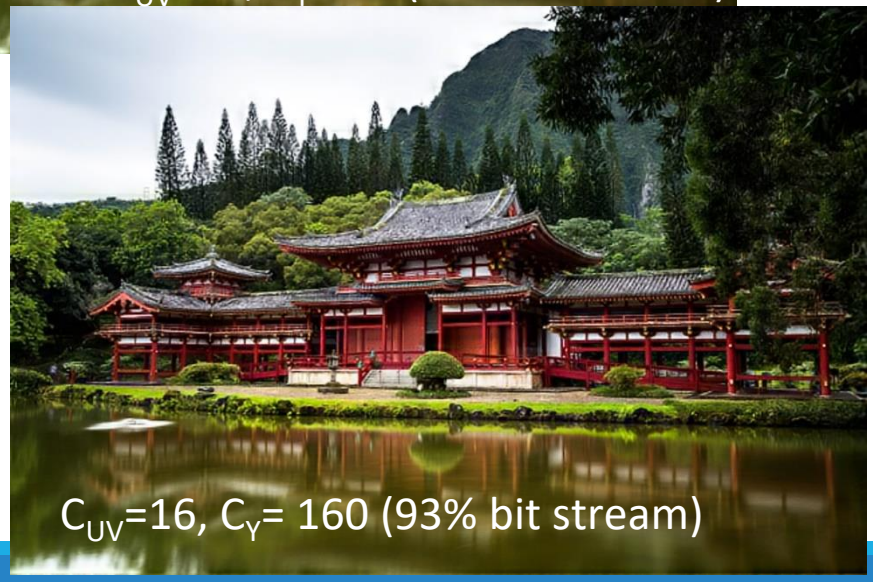
Chroma residual
(total C_{UV}= 96)



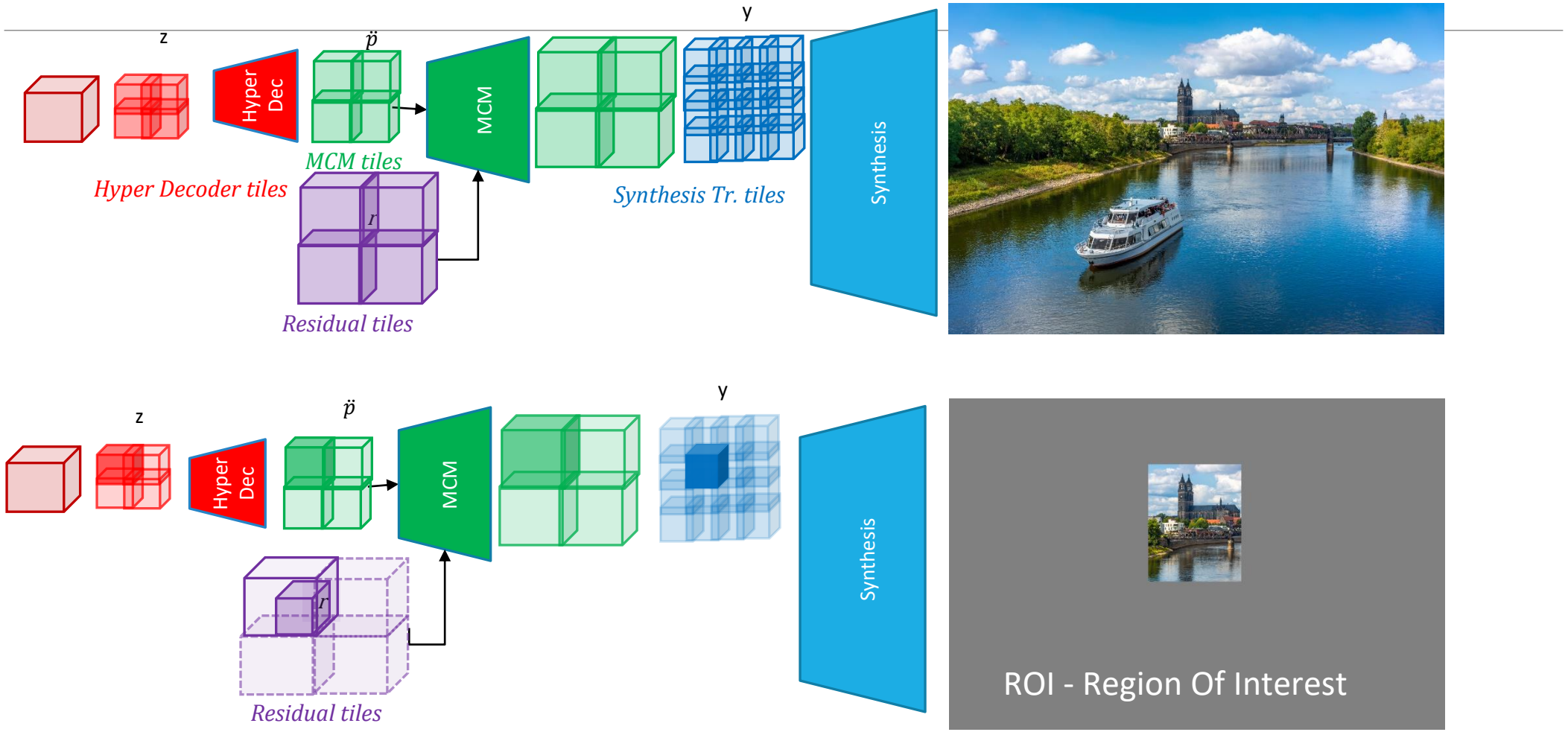
Luma residual
(total C_Y= 160)







Latent domain tiles and ROI decoding



Random access inside picture is possible ***after*** ***<u>parital</u>*** ***residual*** parsing

Local quality control

DecoderID = 1

DecoderID = 2

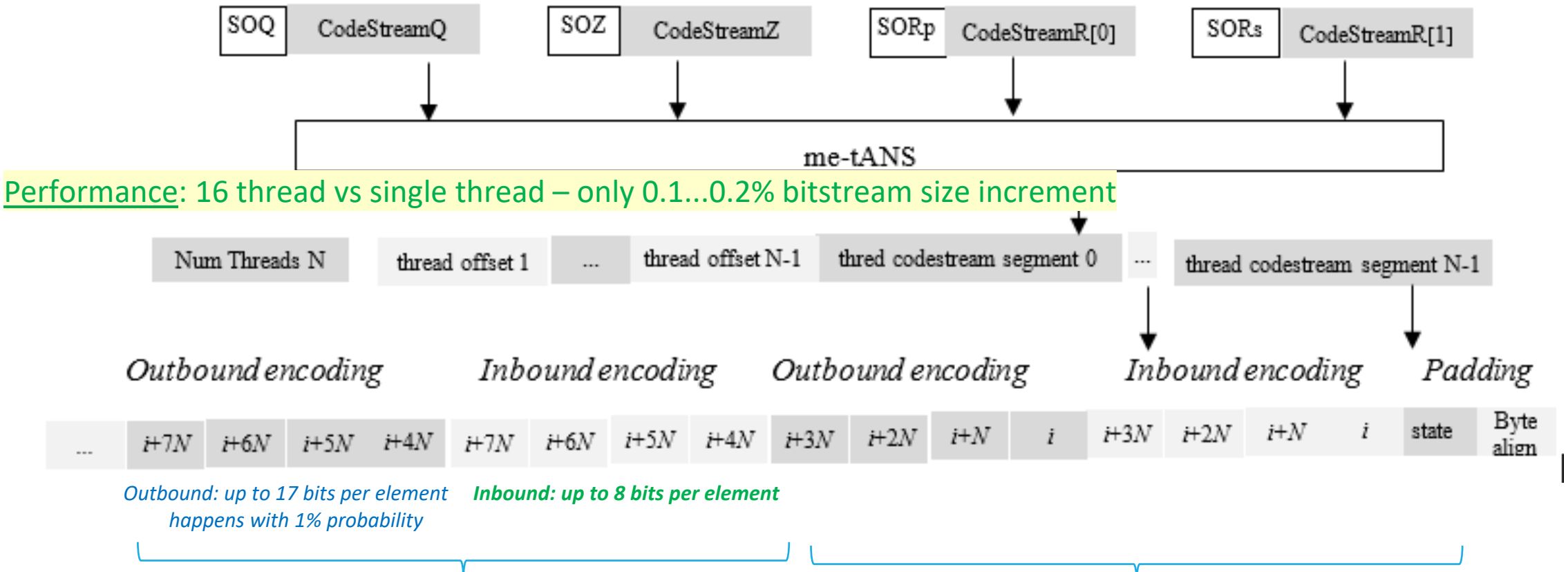
W/o Q-map

Original:

W/ Q-map

me-tANS: memory efficient tabulated Asymmetric Numeral Systems

Multi-thread coding with me-tANS



Performance: 16 thread vs single thread – only 0.1...0.2% bitstream size increment

4 elements in each ‚round', inboudn part can be efficiently ‚packed' into 32 bits register

Bits allocation by JPEG AI and VVC

Original image
1336x872

distortion

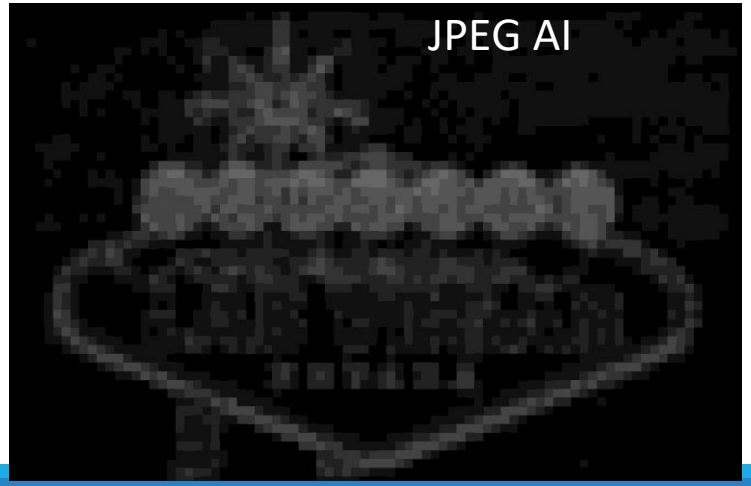
High

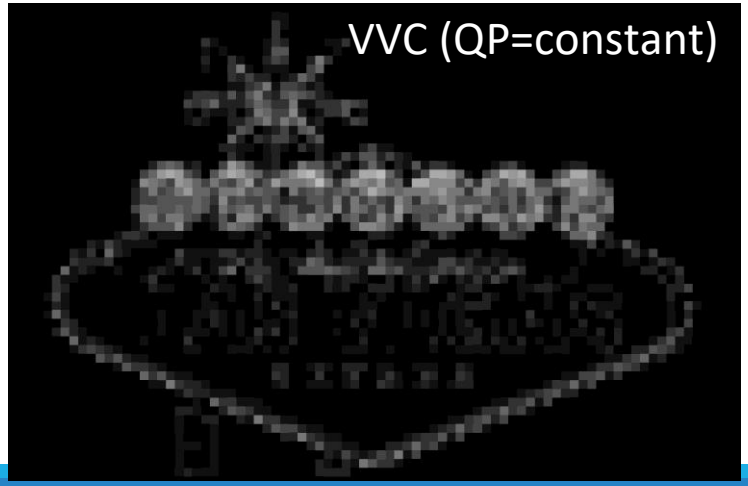
Low

bits

High

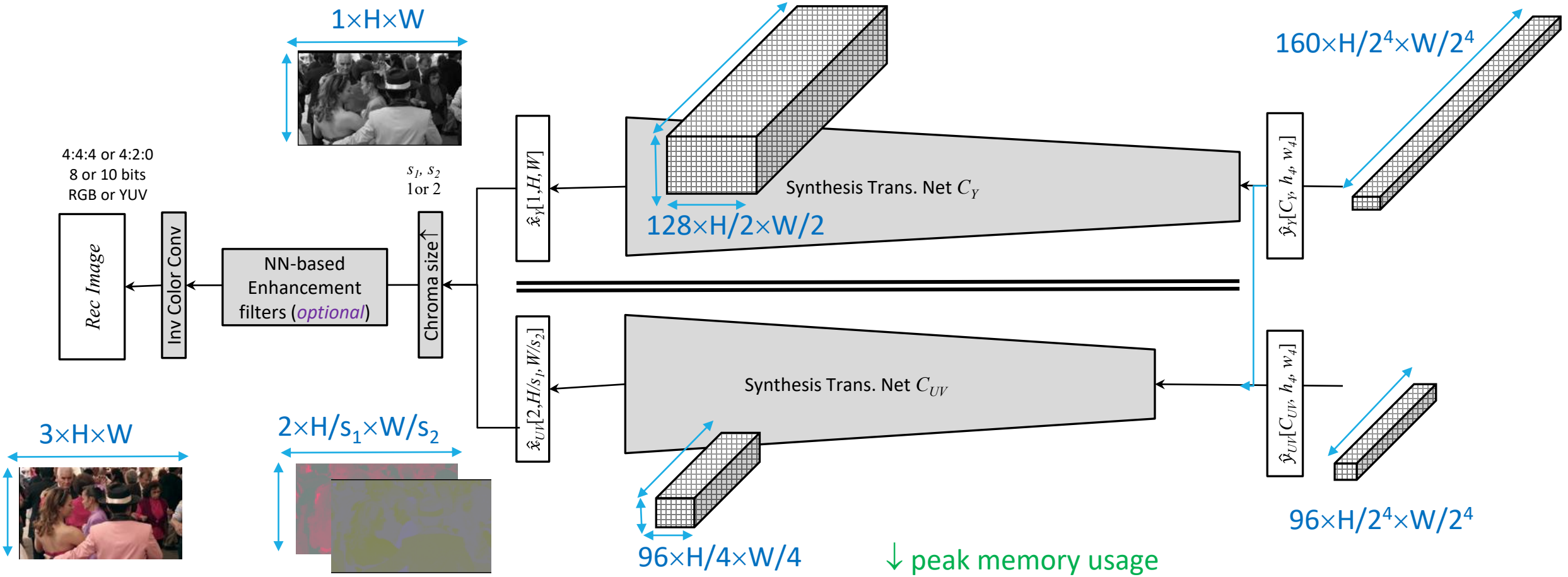
Low





Design aspects motivated by implementation cost

Why colors are separated?

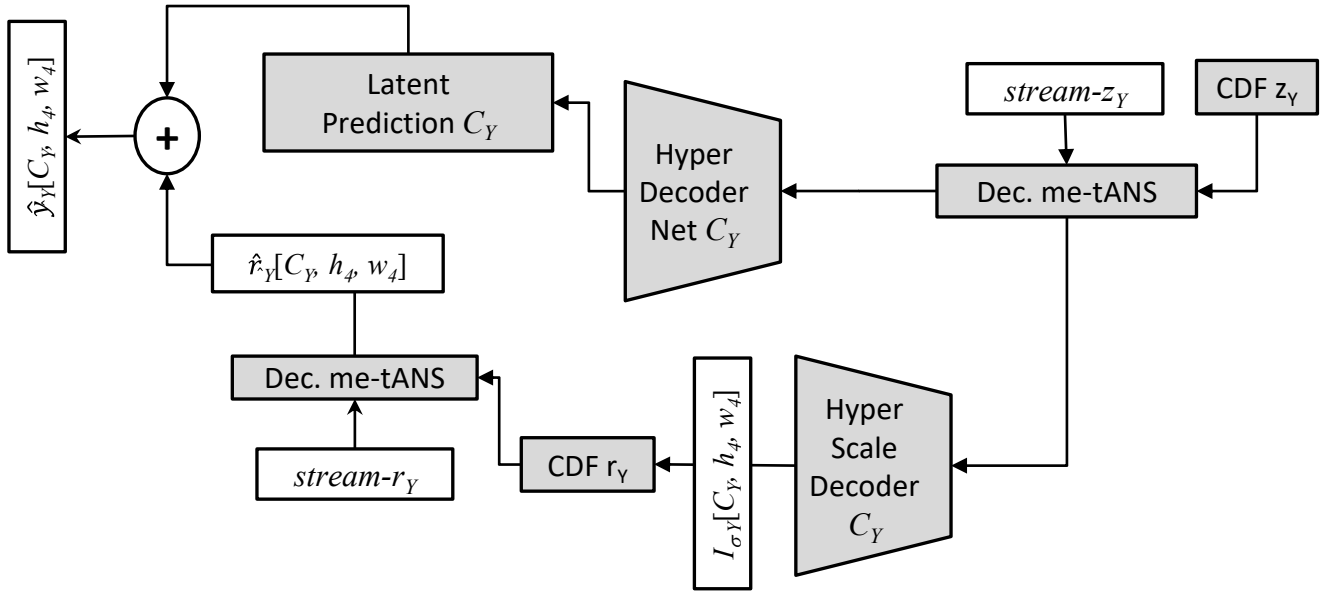


Peak memory usage for 3 colors joint coding for Autoencoder with 256 channels is 256 × W/2 × H/2 → twice higher

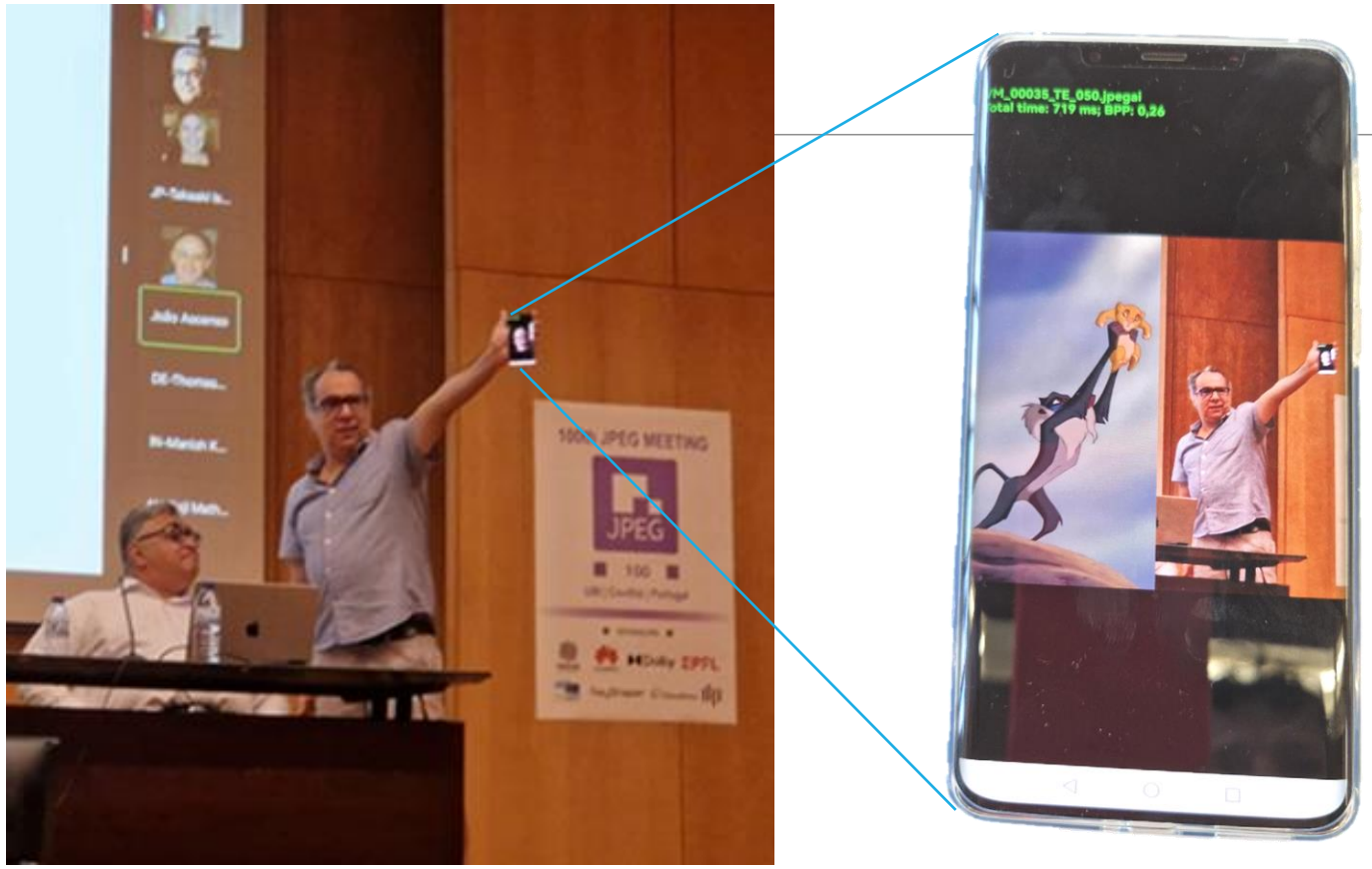
Why prediction / residual coding?

CDF modeled by Gaussian distribution
(μ, σ) - 2 parameters
CDF tables are 2D

For residual μ = 0
CDF tables are 1D



World’s first smartphone implementation of JPEG AI



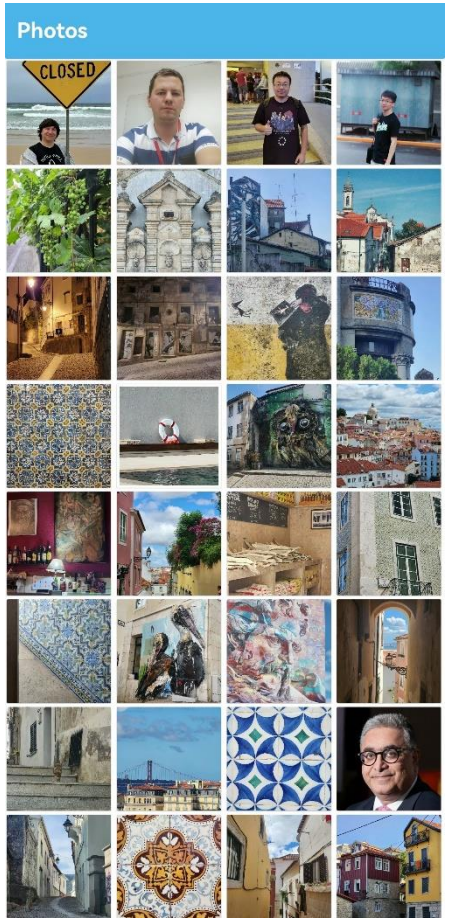
Main targets:

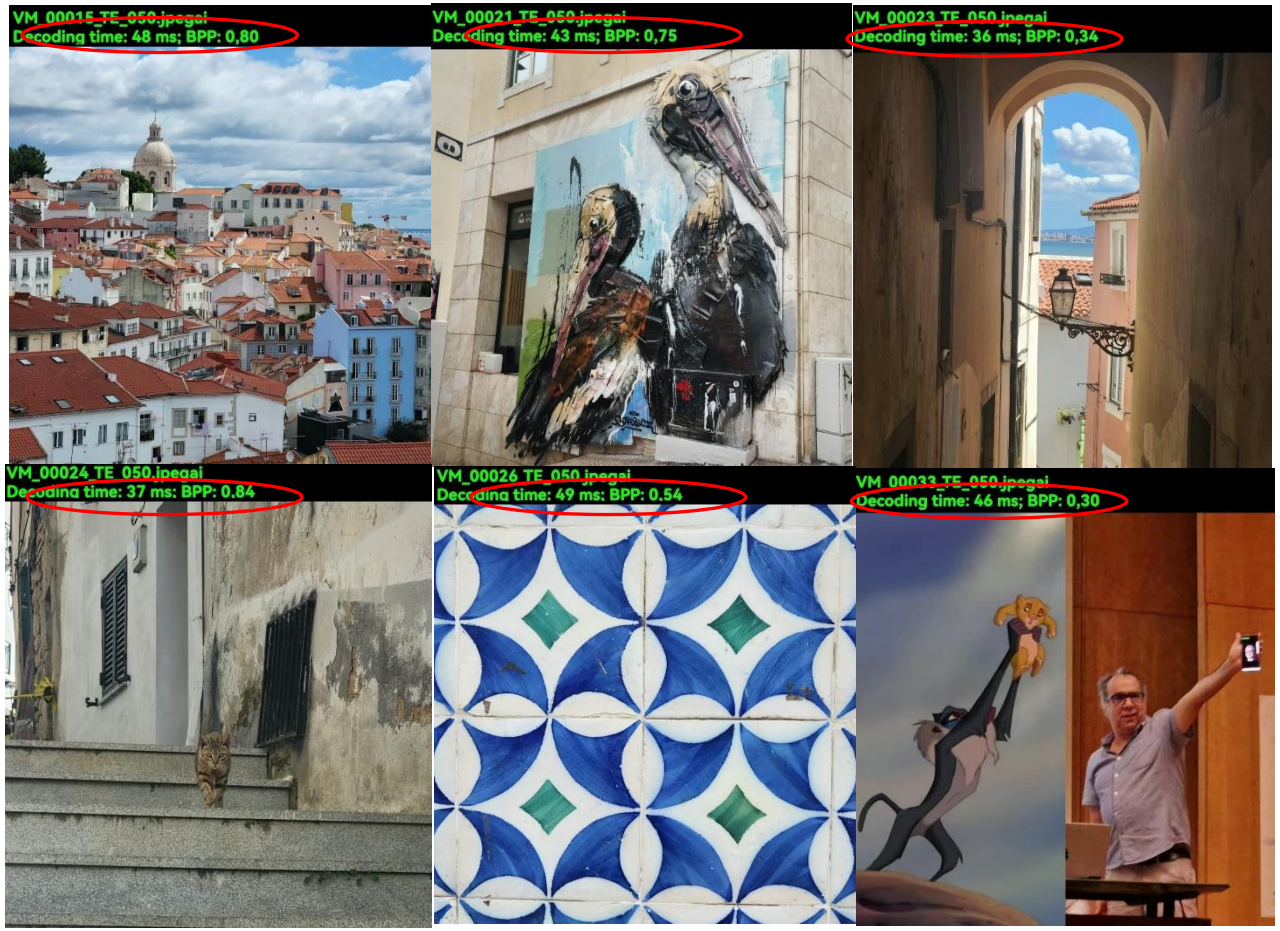
1. Demonstrate to the world that JPEG AI can fly on smartphone right now even without dedicated chip
2. Identify JPEG AI design issues preventing deployment on mobile platform as early as possible
3. Verify device interoperability of JPEG AI standard

- Device: Huawei Mate50 Pro with Qualcomm Snapdragon 8+ Gen1
- Technology stack: C++, Java, SNPE AI acceleration framework, Bolt CPU inference framework
- Features: JPEG AI base profile decoding, high resolution support (4K), tiling

JPEG AI on smartphone in ICIP 2023

Huawei P60 pro





Gallery size

.jpg	~2.6 bpp
.jpegai	~0.6 bpp

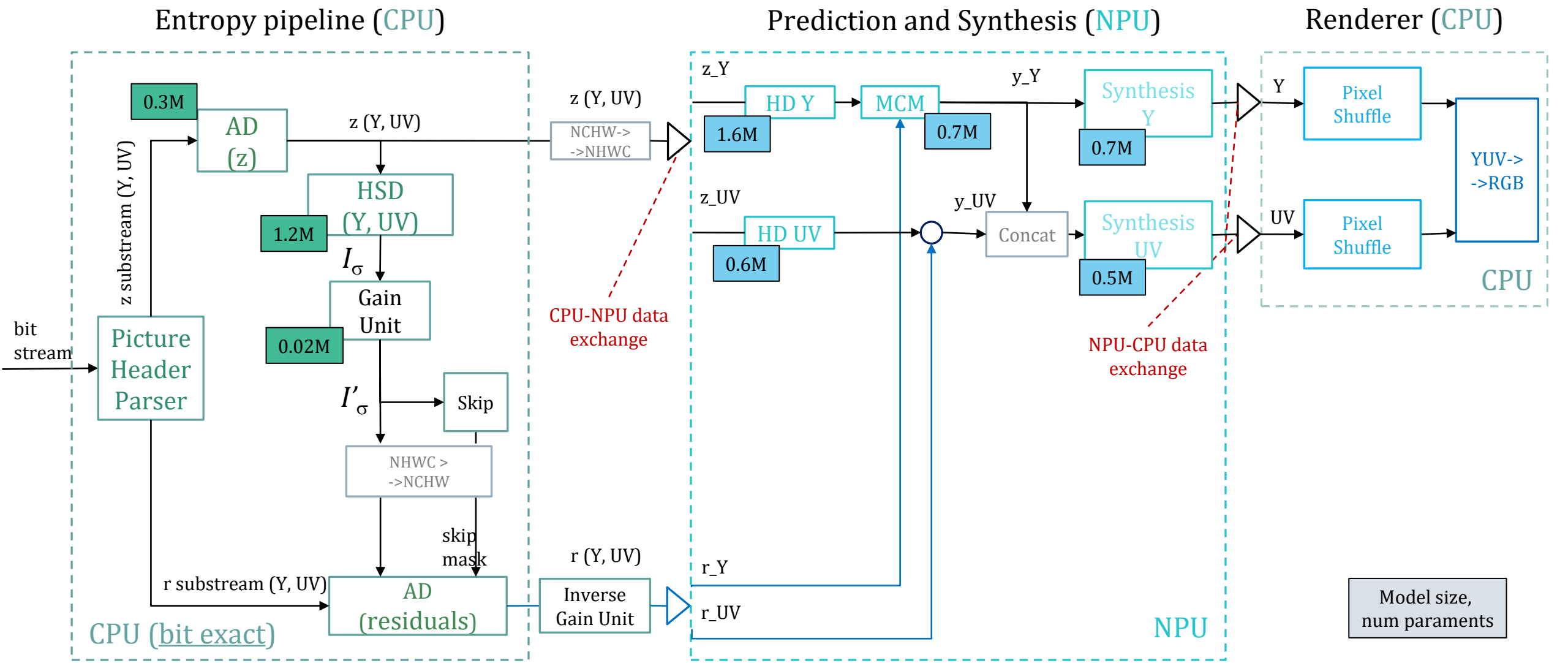
Demo creators (*)

- **Timofey Solovyev**
- Tiansheng Guo
- Alexander Karabutov
- Kang Ning
- Kejie Shao
- Dequan Yu
- Johannes Sauer

Photos captured Portugal, July 2023

JPEG AI mobile device implementation (DecID=1)

AD – Arithmetic Decoder
HSD – Hyper Scale Decoder
HD – Hyper Decoder
MCM – Multistage Context Modelling



Performance test results

5 points BD-rate (0.12, 0.25, 0.5, 0.75, 1.0)

		BD rate vs VVC-012-025-050-075-100							Dec. complexity			Enc.
Test	AVG	MS-SSIM	VIF	FSIM	NLPD	IW-SSIM	VMAF	psnrHVS	kMAC/pxl	Time GPU, x	Time CPU, x	Time GPU, x
VVC-012-025-050-075-100	**0.0%**	0%	0%	0%	0%	0%	0%	0%	1	1	1	1
VM6.1-Enc0Dec0-tools-off	**-12.0%**	-31%	7%	-15%	-12%	-26%	-7%	1%	8	0.36	1	0.0005
VM6.1-Enc0Dec0-tools-on	-16.2%	-31%	7%	-22%	-13%	-27%	-30%	3%	14	0.41	2	0.0011
VM6.1-Enc0Dec1-tools-off	**-16.7%**	-33%	1%	-20%	-16%	-29%	-15%	-4%	23	0.38	2	0.0005
VM6.1-Enc0Dec1-tools-on	-20.2%	-33%	1%	-27%	-17%	-29%	-35%	-2%	28	0.41	3	0.0011
VM6.1-Enc1Dec2-tools-off	**-24.0%**	-38%	-9%	-29%	-23%	-34%	-24%	-11%	214	0.61	28	0.0012
VM6.1-Enc1Dec2-tools-on	-27.0%	-38%	-8%	-35%	-24%	-34%	-42%	-9%	215	0.64	29	0.0018

Reference SW

c++

Python →

Codec	8K Encoding, s
JPEG	5
HEVC / H.265	2689
VVC / H.266	18725
JPEG AI	5 (Enc0) or 20 (Enc1)

Many thanks to Alexander Karabutov – JPEG AI project SW coordinator

Comparison with VVC Intra

y:31.4
u:38.8
v:39.4

y:29.7
u:39.8
v:40.3

VTM 7 KB

Encoding time: 0:00:**12.9**57

QP = 36

JPEG AI VM6.1 7KB

Encoding time: 0:00:00.081

025 (CTTC)

Comparison with VVC Intra

VTM 7 KB

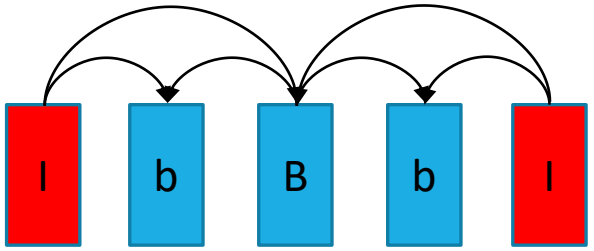
Encoding time: 0:00:**12.9**57

JPEG AI VM6.1 7KB

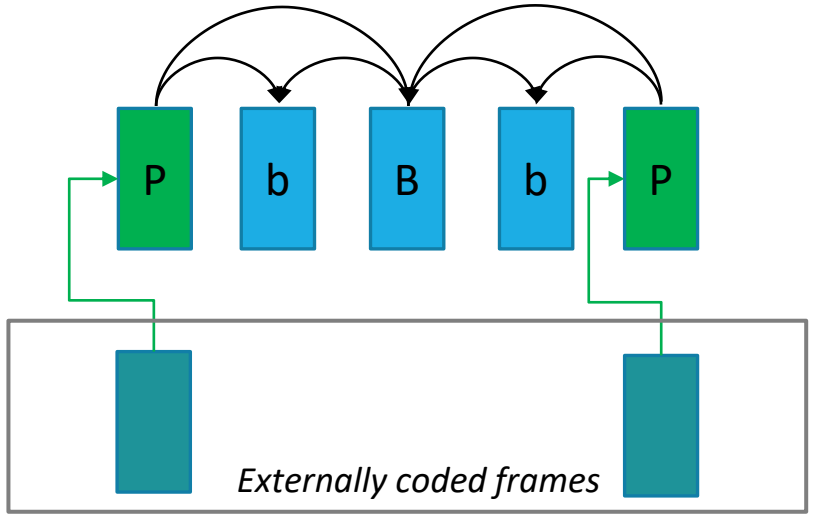
Encoding time: 0:00:00.081

Easy way to incorporate to video coding

HEVC v1 coder



HEVC v3 (scalable and multi-view)



HEVC v1 coder

HEVC v3 + E2E AI coded Intra frame

AI coded refPic can be added to any traditional codec

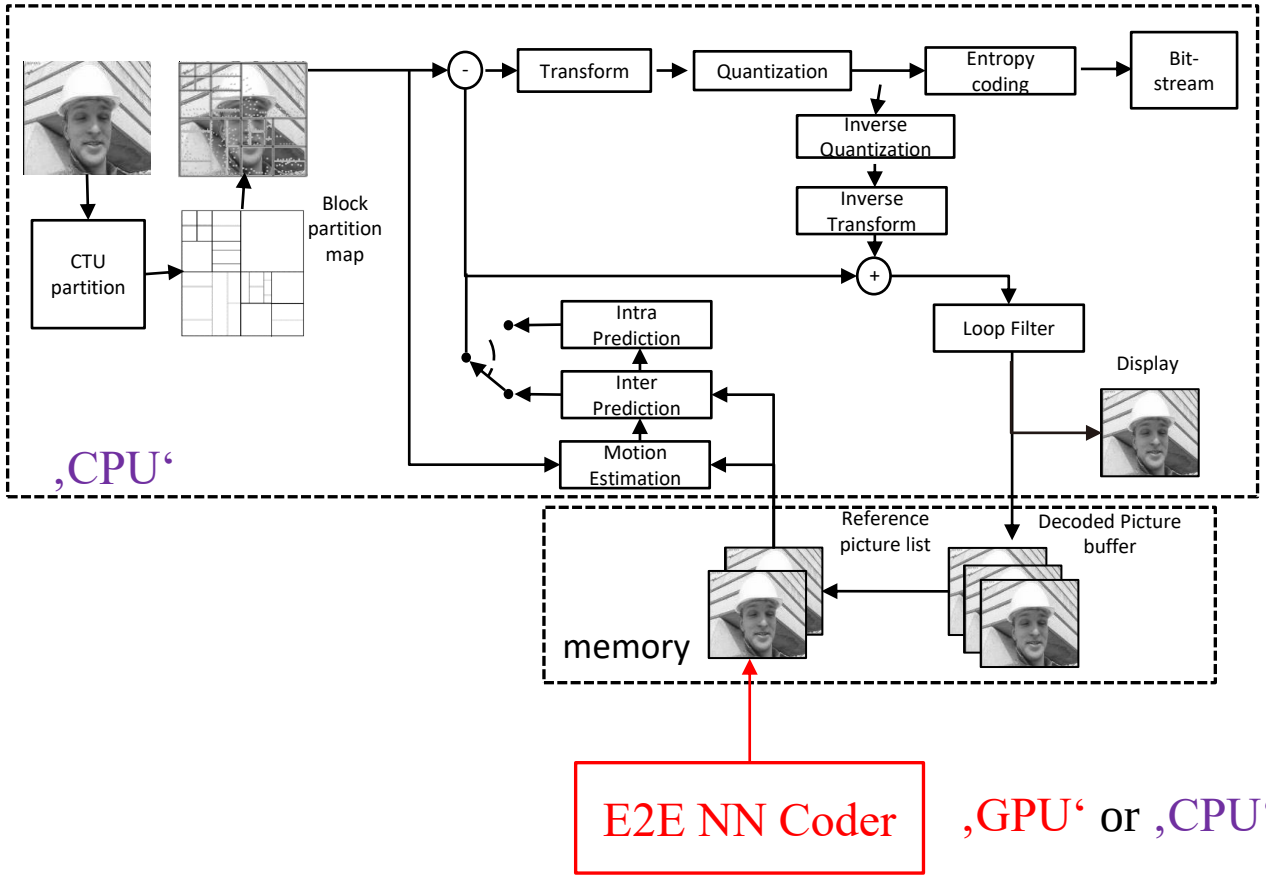
JVET-AJ0208: VVC + DCVC-FM I-frame

JVET-AJ0208 vs VTM (PSNR)

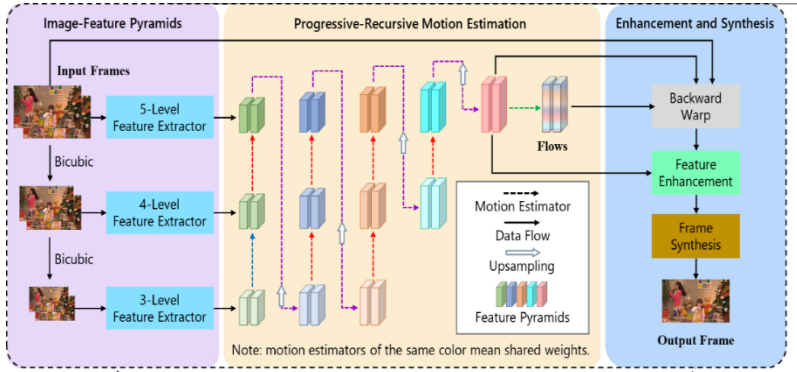
All Intra (AI)				GPU		CPU	
	Y	U	V	Enc	Dec	Enc	Dec
Class C	-5.0%	-10.6%	-8.8%	100%	87%	113%	180%
Class D	-6.4%	-15.1%	-12.4%	100%	93%	108%	133%

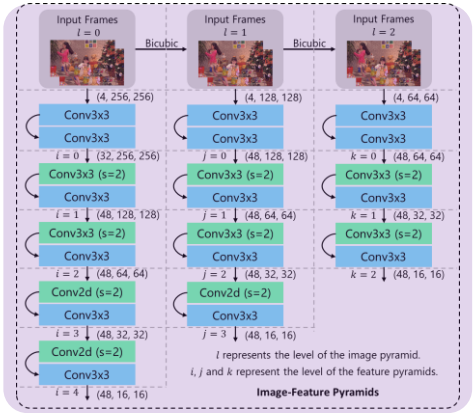
Random Access(RA)				GPU		CPU	
	Y	U	V	Enc	Dec	Enc	Dec
Class C	-0.7%	-4.6%	-3.2%	100%	100%	100%	295%
Class D	-0.9%	-6.5%	-5.9%	100%	100%	100%	222%

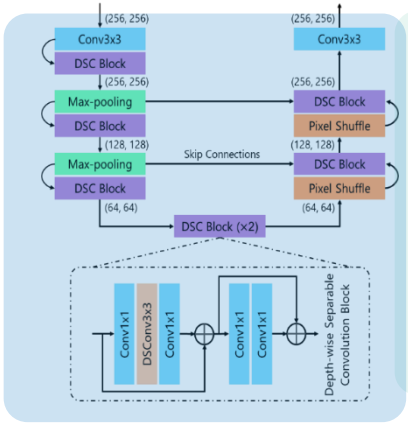
Complexity DCVC-FM I-frame 525 kMac/pxl

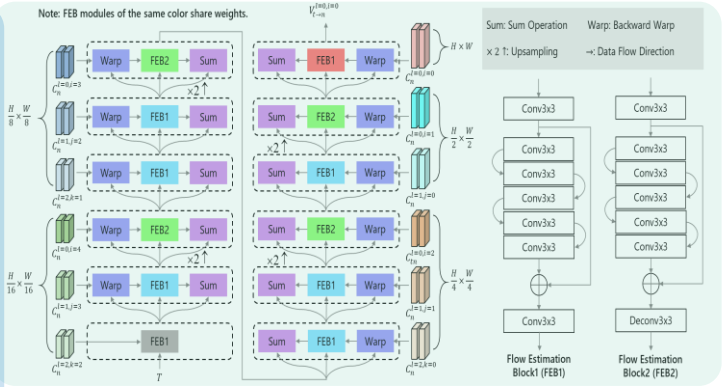


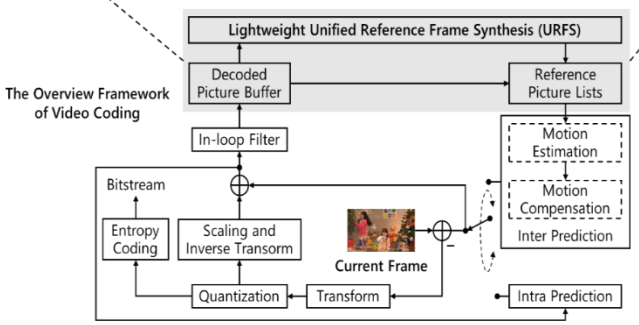
More example of ‘AI synthesized’ RefPicture











Source	BD-rate Random Access			kMAC/pxl	Param, M	Training set
	Y	U	V			
JVET-AG0122	-0.6%	-0.3%	-0.3%	69	2.9	HAA500
JVET-AH0107	-1.7%	-1.6%	-1.3%	69	2.9	Vimeo-90K triplet
JVET-AH0107	-2.6%	-1.8%	-1.5%	182	2.9	Vimeo-90K triplet
JVET-AJ0099	-3.5%	-4.4%	-4.3%	727	2.9	Vimeo-90K triplet

IFRNet variations …

Very strong dependency on training data set…

Take away…

- AI codec standard development requires slightly ***different methodology***
  - Not only CPU run time, tools with million parameters – training cross-check, kMAC/pxl is very rough complexity measure - real mobile device implementation in parallel with standardization – great help!
- **AI coding** tools coding can become a ***reality*** (as standard or/and product)
  - E2E AI image codec (20 kMAC/pxl) can work on Smartphone
  - Remember: 'CPU-GPU' data transfer to minimize, overflow aware NN quant, not every NN operation can fly…
- Some ***attractive features*** come almost for fee with E2E AI codec (more painful for traditional codec)
  - Region of interest or partial decoding, progressive decoding, single stream multiple decoders, vision for machine and human from same bit-stream, easier perceptual optimization, ultra-fast encoder …
- No need to choose ***'traditional' or 'E2E AI'*** codec! ***Both***!
  - Just an example: AI coded external reference picture (can combine with HEVC v3 right now)…

I'm deeply grateful for the wonderful team we have!!!

Thank you for attention!!!