Qualcomm

CLIC 2022-06-19

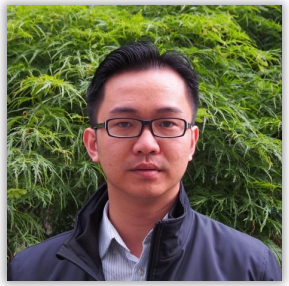
Efficient neural codecs

Auke Wiggers

Staff engineer

Qualcomm Technologies Netherlands B.V.

Collaborators


Hoang Le


Reza Pourreza


Amir Said


Guillaume Sautière


Yinhao Zhu


Yang Yang


Taco Cohen


Auke Wiggers

Covered in this talk:

- Yinhao Zhu et al., "Transformer-based Transform Coding", ICLR 2022
- Hoang Le et al., "MobileCodec: Neural Inter-frame Video Compression on Mobile Devices", ACM MMSys 2022

At Qualcomm, we are interested in creating efficient neural codecs.

We want neural codecs to be adopted in production-like settings.

If you are working on neural codecs, you should share this interest: real-world deployment often highlights practical constraints and issues.

We believe that adoption will require the codec to…

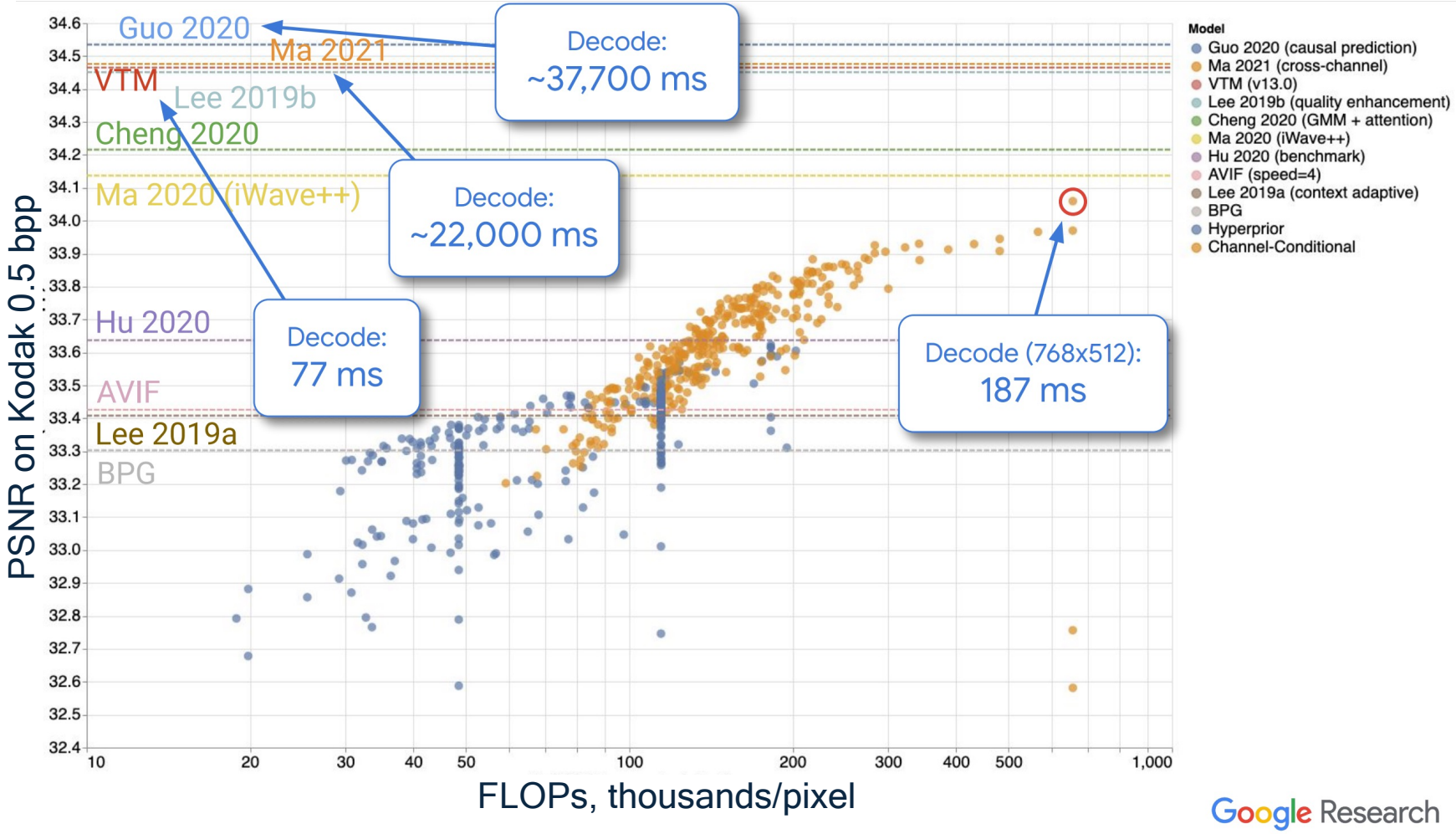
1. Be efficient: use low-compute transforms and priors
2. Be deployable: run efficiently on-device, not only on desktop GPUs
3. Be performant: outperform existing codecs in metrics that matter

We believe that adoption will require the codec to…

1. Be efficient: use low-compute transforms and priors
2. Be deployable: run efficiently on-device, not only on desktop GPUs
3. Be performant: outperform existing codecs in metrics that matter

Neural codecs should be efficient

Recent work improves RD performance, but at increasing computational cost.

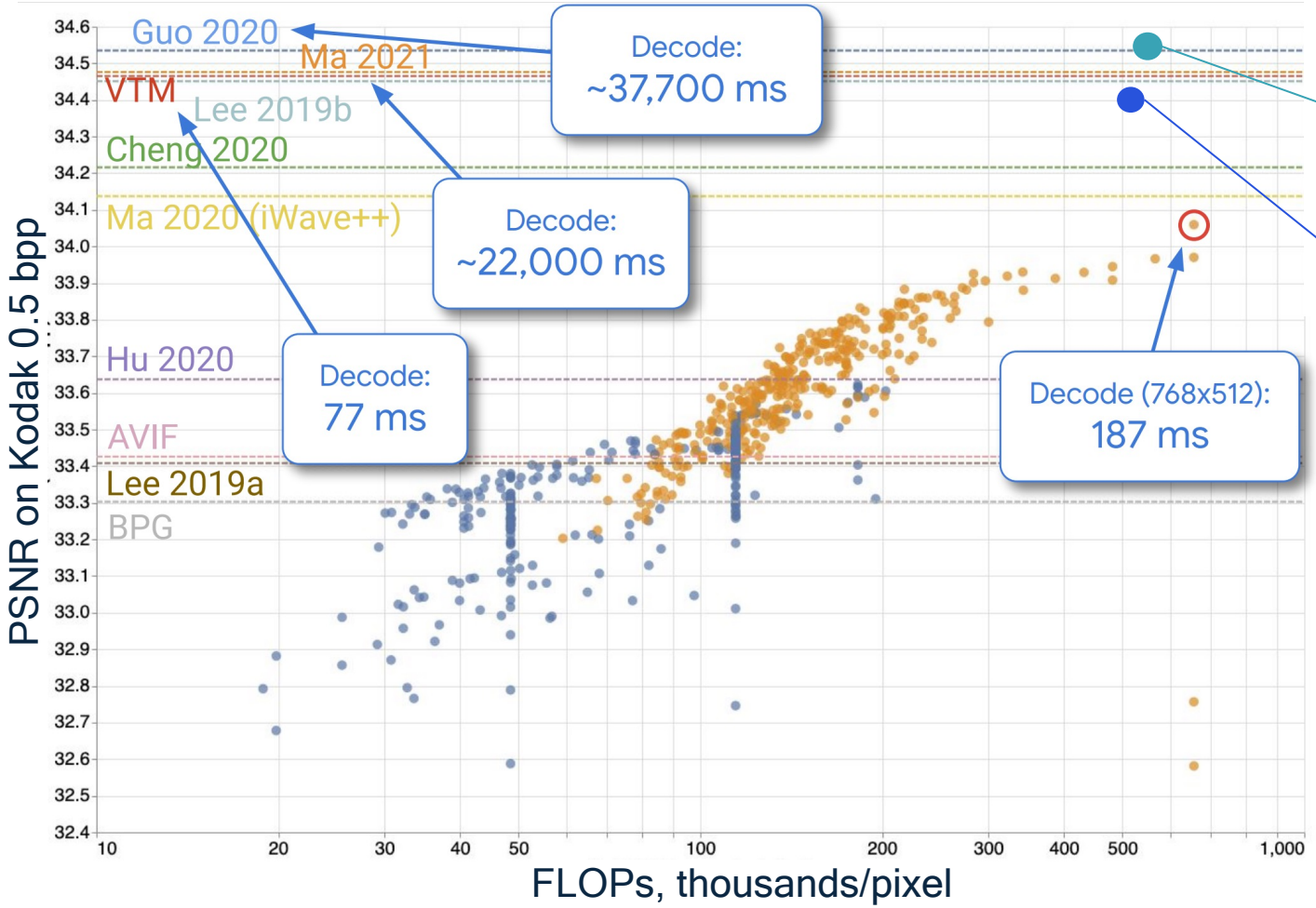


Google Research

David Minnen, “Curent Frontiers in Neural Image Compression “, ICIP 2021 plenary talk.

Neural codecs should be efficient

Recent work improves RD performance, but at increasing computational cost.



SwinT-ChARM
Decode: 167 ms

SwinT-Hyperprior
Decode: 114 ms

This is where our work comes in!

Small disclaimer

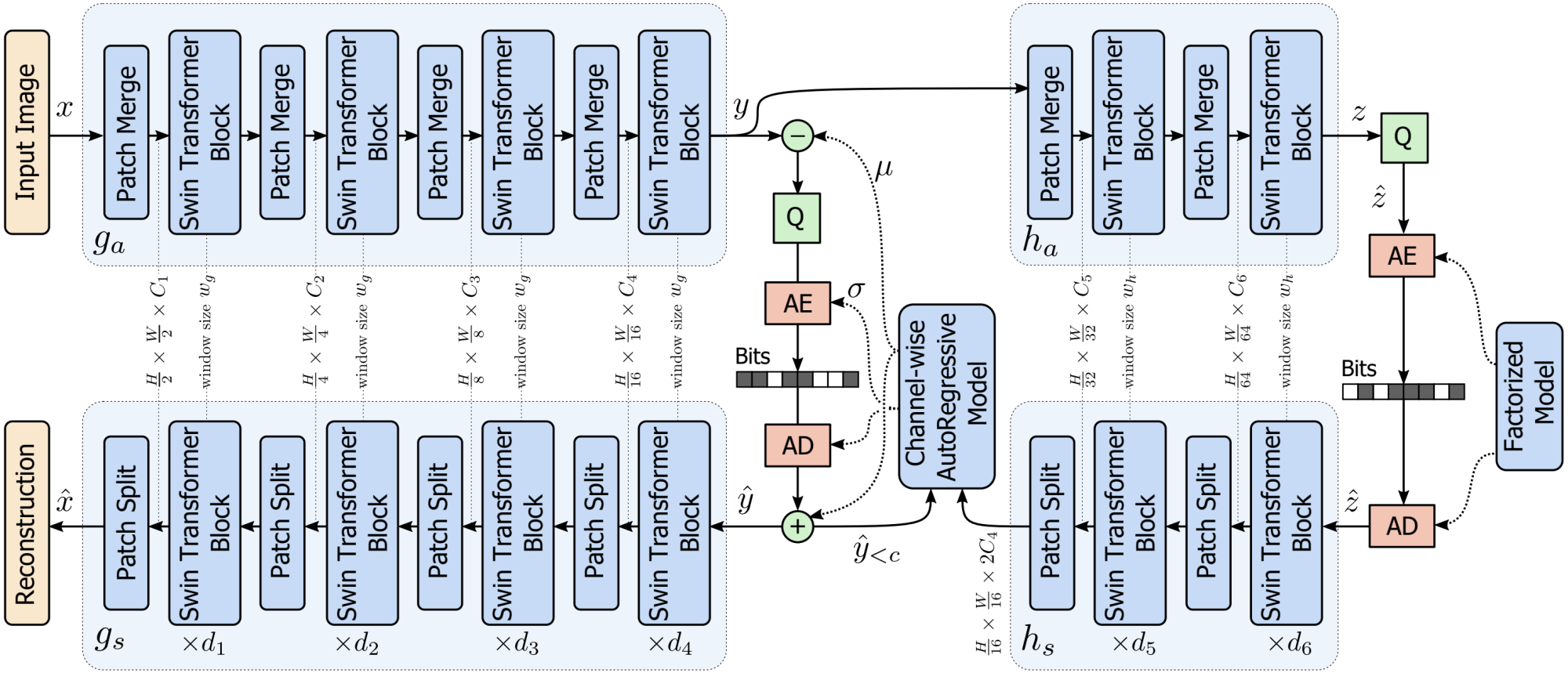
- Our SwinT models benchmarked on 2080 Ti, runtimes for neural baselines taken from papers
- None of the shown methods have been explicitly optimized for runtime

Google Research

David Minnen, “Curent Frontiers in Neural Image Compression “, ICIP 2021 plenary talk.

We focus on improving the transform

We replace strided convolutions by swin-transformer blocks [1] and custom up- and downsampling layers.

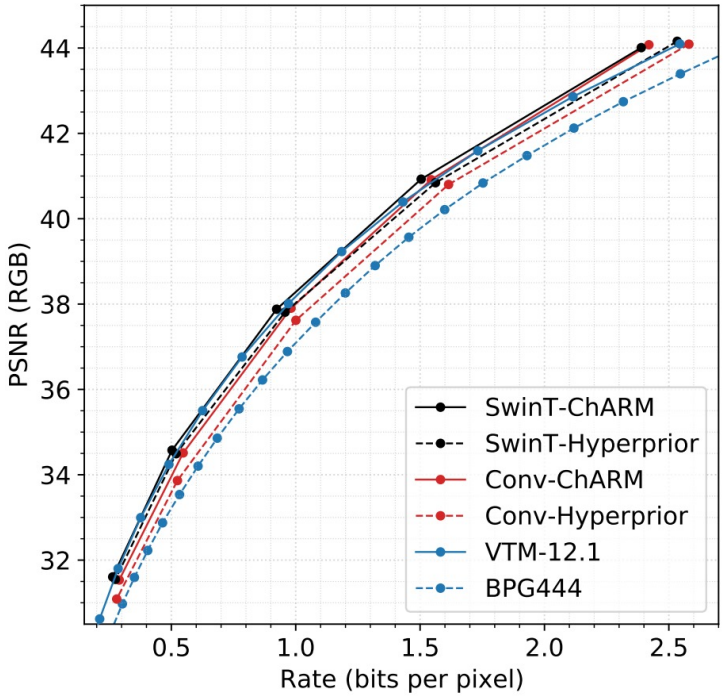


[1] Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

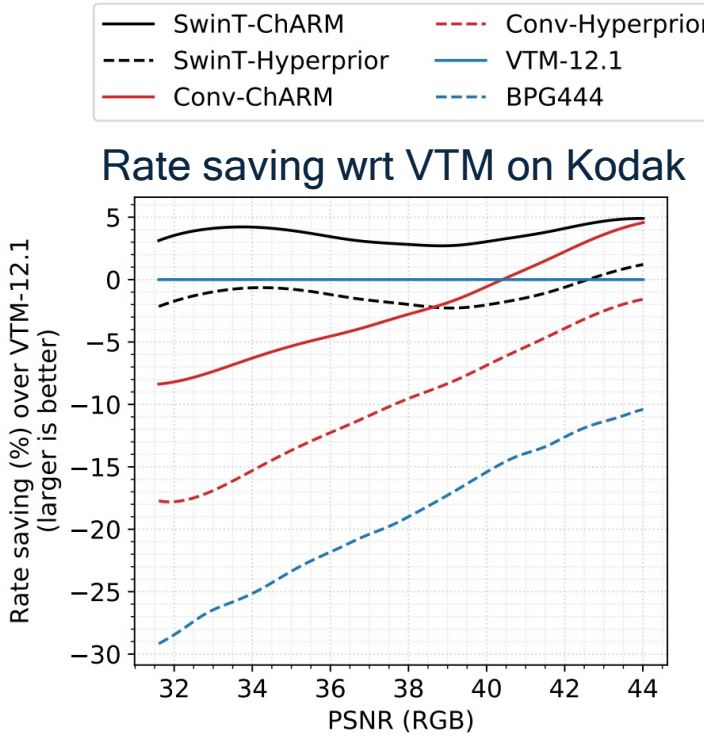
Performance: Image compression

This improves image compression performance, far outperforming convolutional hyperpriors.

Rate-distortion on Kodak (RGB444)



Rate saving wrt VTM on Kodak



BD rate wrt VTM on various image datasets

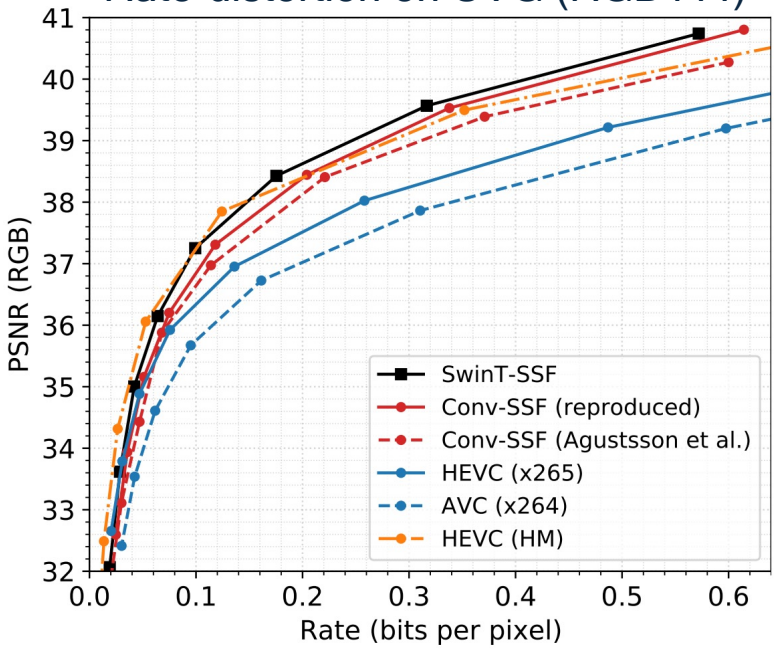
Image Codec	Kodak	CLIC2021	Tecnick	JPEG-AI
BPG444	20.87%	28.45%	27.74%	27.14%
Conv-Hyperprior	11.65%	12.23%	12.49%	20.98%
Conv-ChARM	3.44%	4.14%	3.50%	9.59%
SwinT-Hyperprior	1.69%	0.83%	-0.15%	6.86%
SwinT-ChARM	**-3.68%**	**-5.46%**	**-7.10%**	**0.69%**

Improvement is consistent across datasets, for both factorized and channel-autogressive (ChARM) models.

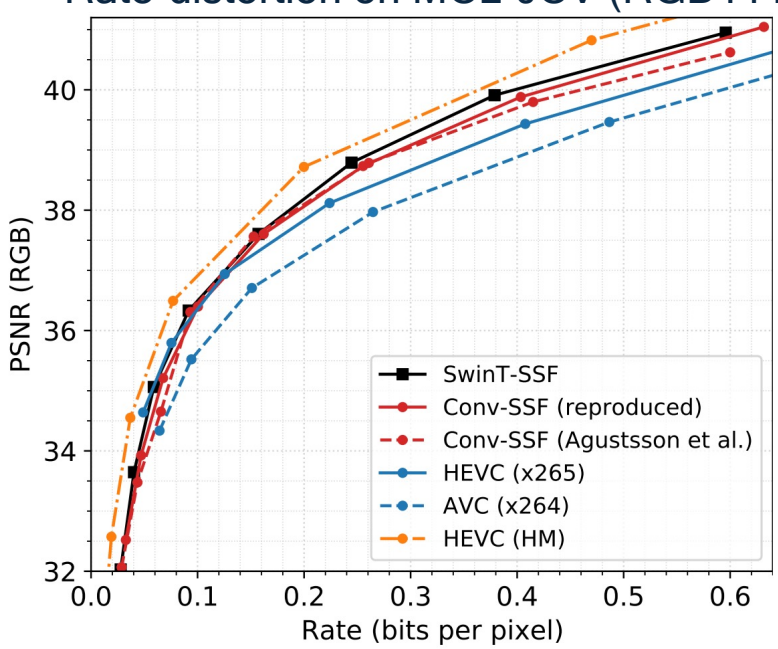
Performance: P-frame video compression

Using SwinT based encoder and decoder in flow+residual hyperpriors of a scale-space flow model [1] improves video compression performance as well.

Rate-distortion on UVG (RGB444)



Rate-distortion on MCL-JCV (RGB444)



BD rate wrt Conv-SSF (reproduced)

Video Codec	UVG	MCL-JCV
HEVC (x265)	25.97%	25.83%
HEVC (HM)	**-15.80%**	**-24.96%**
SwinT-SSF	-12.35%	-10.03%

[1] Agustsson et al. "Scale-space flow for end-to-end optimized video compression.", CVPR 2020

Performance: compute and complexity

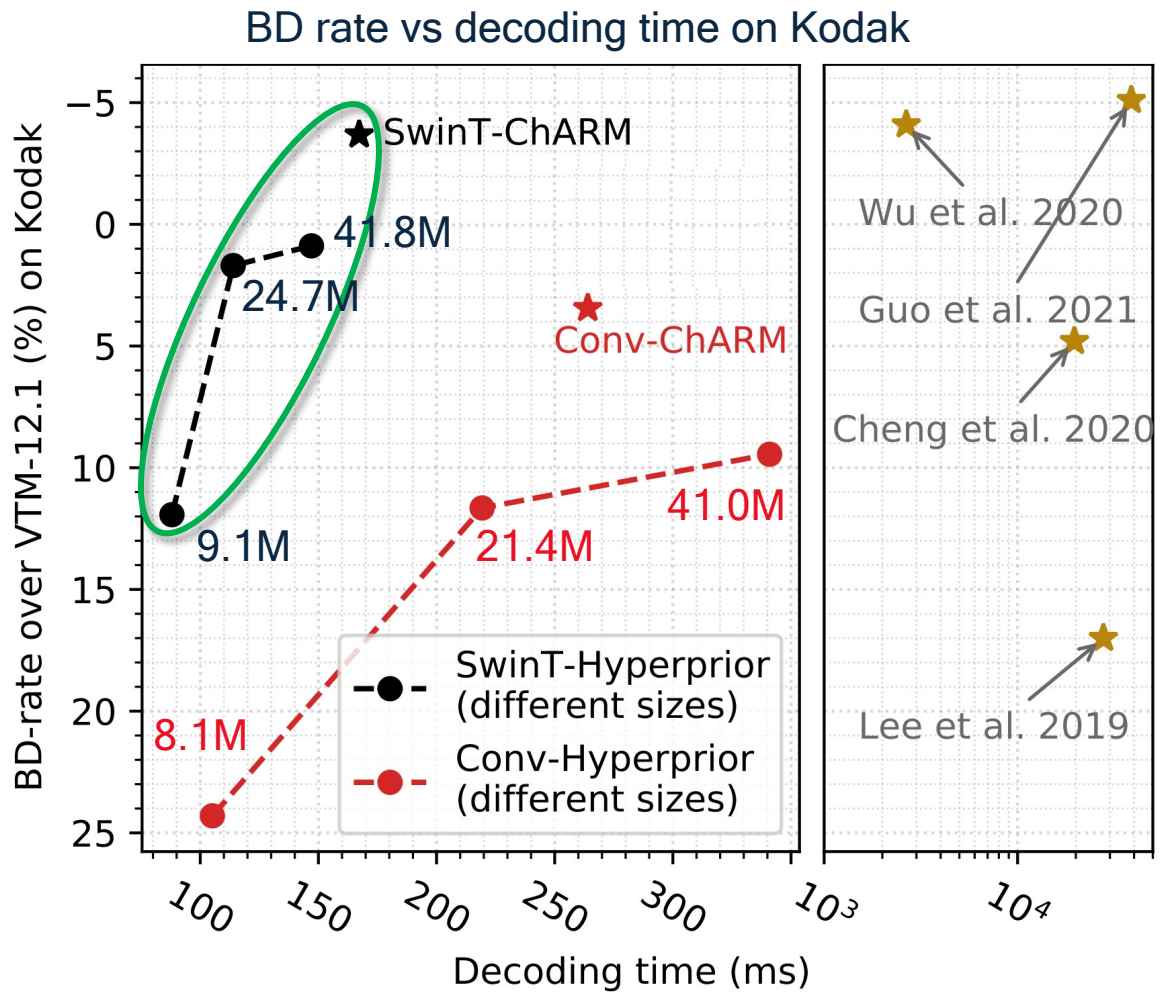
Time-to-decode for decoder g_s and GMACs are much lower for SwinT models than for conv.

Peak memory usage does increase due to the (memory-expensive) linear attention operation.

Codec	Time (ms)					GMACs	Peak memory	Model params
	$\hat{\mathbf{z}}$	h_s	$\hat{\mathbf{y}}$	g_s	total			
Conv-Hyperprior	5.5	4.0	38.2	168.9	219.3	350	0.50GB	21.4M
Conv-ChARM	5.3	4.1	82.9	168.5	264.0	362	0.53GB	29.3M
SwinT-Hyperprior	6.0	4.8	38.1	59.6	114.0	99	1.44GB	24.7M
SwinT-ChARM	5.9	4.8	90.7	60.1	167.3	111	1.47GB	32.6M

Benchmarked 0.7bpp models on Kodak (512x768) on RTX 2080 Ti GPU, deterministic convolution.

Performance: compute and complexity



Our work improves RD performance at much lower computational cost.

Small disclaimer

- Our SwinT models benchmarked on 2080Ti, runtimes for neural baselines taken from papers
- None of the shown methods have been explicitly optimized for runtime

Yinhao Zhu et al., "Transformer-based Transform Coding", ICLR 2022

In conclusion

Vision transformer-based transforms are an **efficient** alternative to convolutional ones.

In the paper, we show that…

- Turning a conv-hyperprior into a SwinT-hyperprior is straightforward
- SwinT models far outperform similar-size convolutional ones in image and video compression
- SwinT models are compute-efficient: lower runtime and MACs for same number of parameters

Consider using these in your encoder / decoder.

Yinhao Zhu et al., "Transformer-based Transform Coding", ICLR 2022

We believe that adoption will require the codec to…

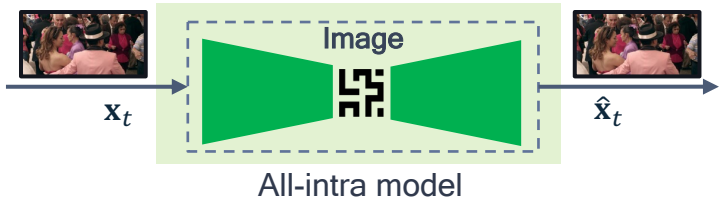
1. Be efficient: use low-compute transforms and priors
2. Be deployable: run efficiently on-device, not only on desktop GPUs
3. Be performant: outperform existing codecs in metrics that matter

Neural codecs should be deployable

We build prototypes and deploy these to mobile devices.

May 2021

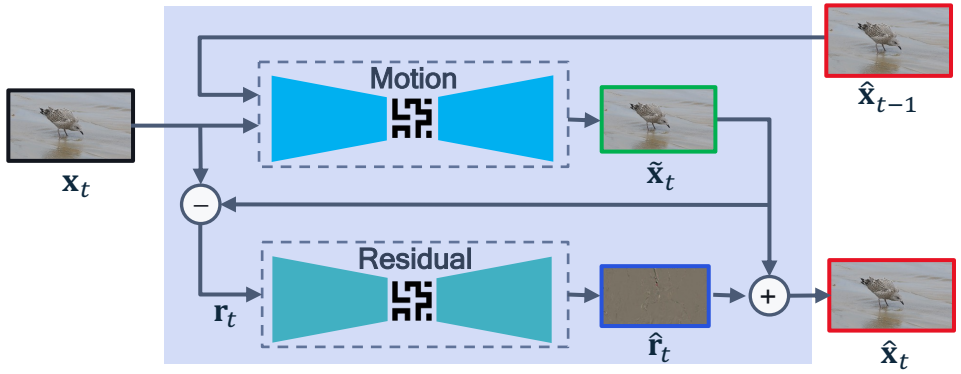
- ✓ All-intra on-device codec
- ✓ 30+ FPS 1280×704 decoding on-device
- ✓ Demo'd at CVPR21, ICML21, and ICIP21



All-intra model

November 2021

- ✓ Low-delay on-device codec
- ✓ 30+ FPS 1280×720 decoding on-device
- ✓ Demo'd at NeurIPS21, ACM MMSys 22

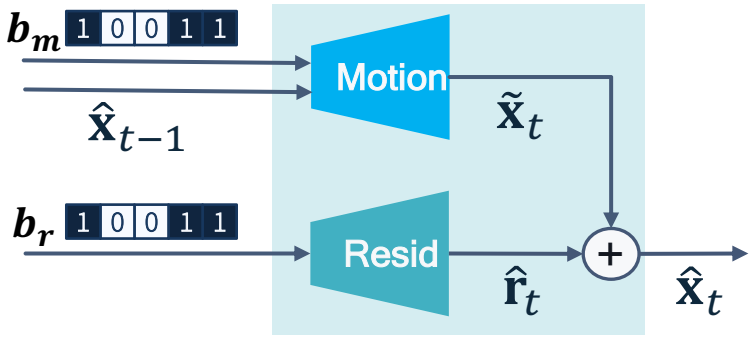


Low-delay model

Hoang Le et al., "MobileCodec: Neural Inter-frame Video Compression on Mobile Devices", ACM MMSys 2022

Solution to practical challenges

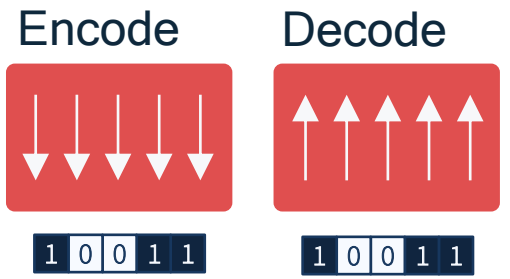
Challenge	High complexity architecture, expensive operators	Inefficient floating-point operations	Slow sequential entropy coding
Solution	Developed warping-free architecture	AIMET channel-wise quantization-aware training	Parallel Entropy Coding (PEC)



P-frame receiver

AI Model Efficiency Toolkit (AIMET)

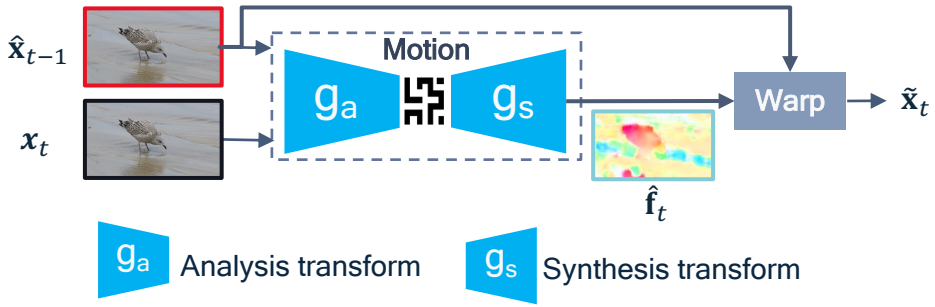
https://github.com/quic/aimet



Warping-free architecture

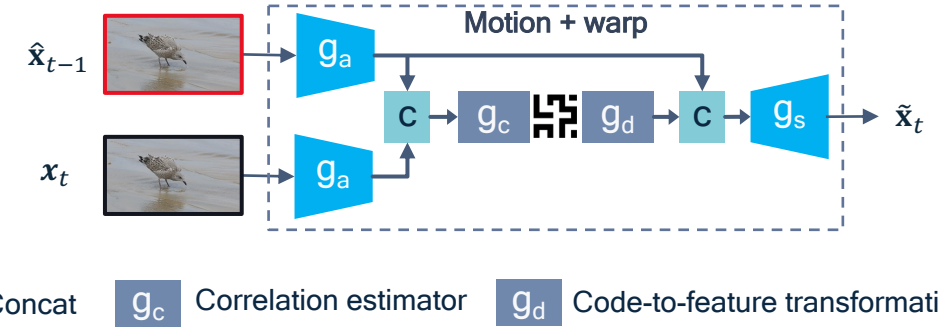
Scale-space flow architecture [1]

First transmit motion information, then perform pixel-level motion compensation (warping).



FLow-AGnostic (FLAG) architecture

Perform motion transmission and compensation jointly in feature-space, using vanilla 3x3 and 5x5 convolutions.

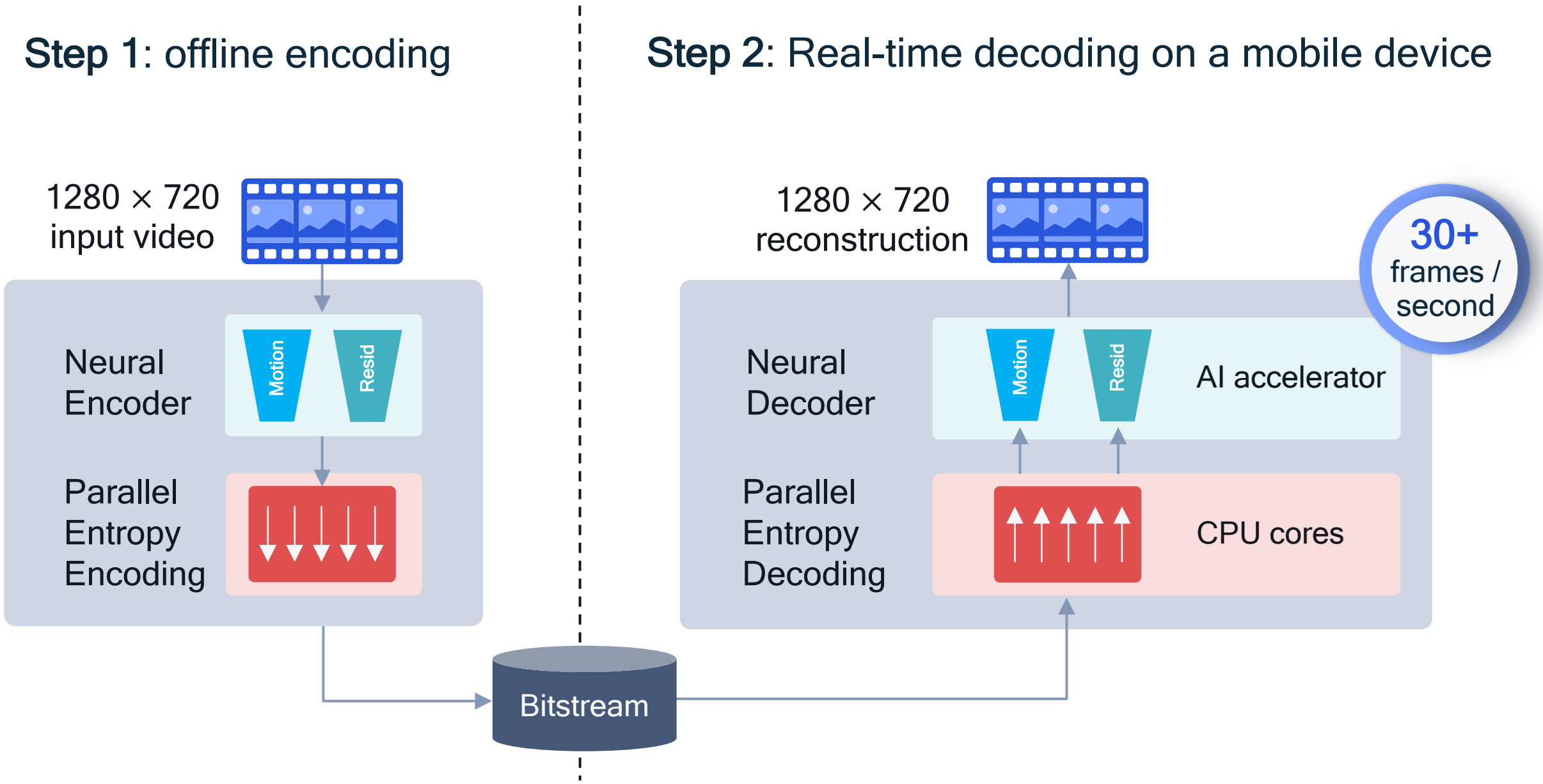


Other steps that improved efficiency:

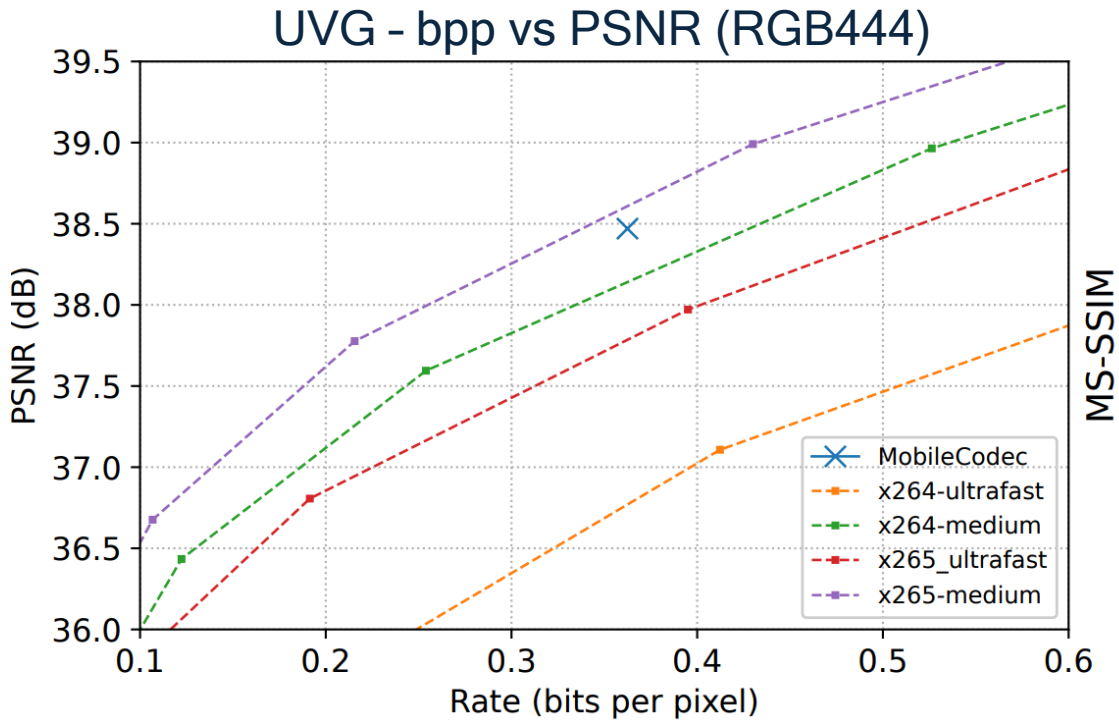
- 3x3 and 5x5 convolutions + ReLU activation throughout
- scale-only hyperpriors
- Gaussian unconditional hyperprior

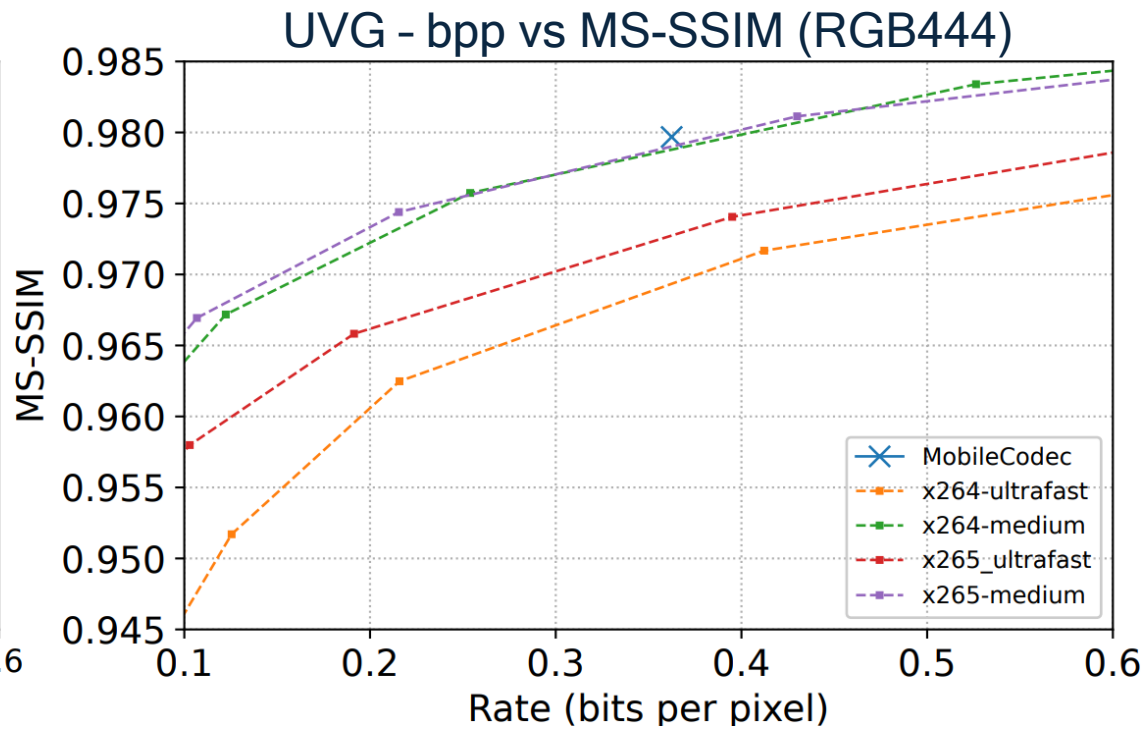
(FVC [2] and C2F [3] warp in feature space using deformable conv)

[1] Agustsson et al. "Scale-space flow for end-to-end optimized video compression.", CVPR 2020
[2] Lu et al., "FVC: A new framework towards Deep Video Compression in Feature Space", CVPR 2021
[3] Hu et al., "Course-to-fine Deep Video Coding with Hyperprior-guided Mode Prediction", CVPR 2022


Step 1: offline encoding
Step 2: Real-time decoding on a mobile device
1280 × 720
input video
Neural
Encoder
Motion
Resid
Parallel
Entropy
Encoding
Bitstream
1280 × 720
reconstruction
30+
frames /
second
Neural
Decoder
Motion
Resid
AI accelerator
Parallel
Entropy
Decoding
CPU cores

Our on-device codec is competitive with ffmpeg x265 (zerolatency) on UVG.





We provide additional results and runtimes for demo videos in the paper.

In conclusion

Neural video codecs can be **deployed** to device, and decode video in real time.

This requires:

- Replacing of expensive or difficult-to-quantize operators (such as dense warping)
- Careful model quantization
- Fast implementation of on-device entropy coding

We will continue to deploy codecs to device to understand the corresponding challenges.

Hoang Le et al., "MobileCodec: Neural Inter-frame Video Compression on Mobile Devices", ACM MMSys 2022

We believe that adoption will require the codec to…

1. Be efficient: use low-compute transforms and priors
2. Be deployable: run efficiently on-device, not only desktop GPUs
3. Be performant: outperform existing codecs in metrics that matter

Neural codecs should be performant (measured in metrics that matter)

Neural codec community mostly evaluates RGB444 PSNR and MS-SSIM. We are guilty of this as well.

This may not convince those working with video codecs ..

- Some favor YUV420 PSNR, which matches human perception more closely than Euclidean RGB
- Some favor learned distortion measures like VMAF
- We've also heard "I only trust my eyes"

For neural codecs to be adopted in realistic settings:

- They need to win in user studies (used in Image and Video compression challenge)
- We need a better metric (Perceptual metric challenge)
- I don't think they need to win on YUV420 PSNR, but I could be wrong

We are working towards a neural video codec that is…

1. Efficient: we’re developing codecs with low complexity
2. Deployable: we’re deploying codecs to identify and solve related challenges
3. Performant: in future work, not just RGB444 metrics

Measure compute in your work, as this will help make the case for adoption!

Thank you

Snapdragon

Follow us on:

For more information, visit us at:

snapdragon.com & snapdragoninsiders.com