ML Efforts in Google around Practical Codecs

Debargha Mukherjee, In-Suk Chong

Google

Google-Meta Workshop @ ICIP 2024, Abu Dhabi, UAE, October 30, 2024

Outline

- Introduction
- CNN Based In-loop filtering
  - Switchable-models
  - Guided CNN
  - Overall Framework
  - Hardware Design & Analysis
  - Results
- ML Based Encoder Optimizations
- Conclusion

Introduction

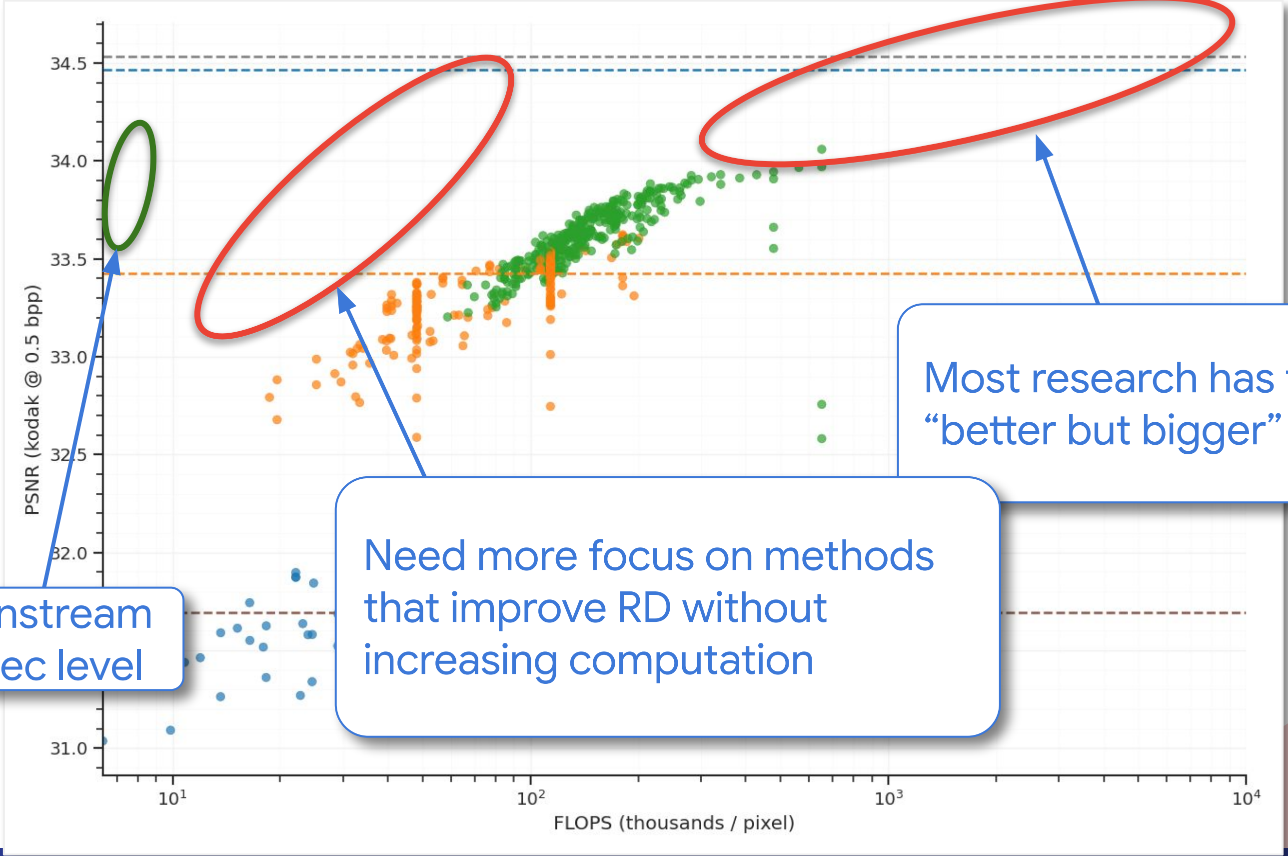
Introduction

Challenges in AI based video codecs

- Video decoder has very challenging constraints!
  - Silicon area for decoder ASIC in a mobile chipset is very limited.
  - Should support throughput to handle 4K/60fps or 8K/30fps video.
    - Throughput requirement for UHD video is similar to a large LLM (100 tokens/sec)
  - From one gen to next, no more than doubling of area is expected for 30+% coding gain
    - A small AI model that gives only 4% coding gain will blow that budget.
- Hybrid AI codecs most promising so far
  - Augment a conventional pipeline with AI tools: In-loop filtering most promising (AOM, JVET)
    - Decoder side inference must be very, very light.
- JPEG-AI - being standardized for images; first full AI image codec.
  - But … video is a different beast altogether.

Introduction

Current State of Neural Codec Research



Mainstream codec level

Need more focus on methods that improve RD without increasing computation

Most research has focused on "better but bigger" models

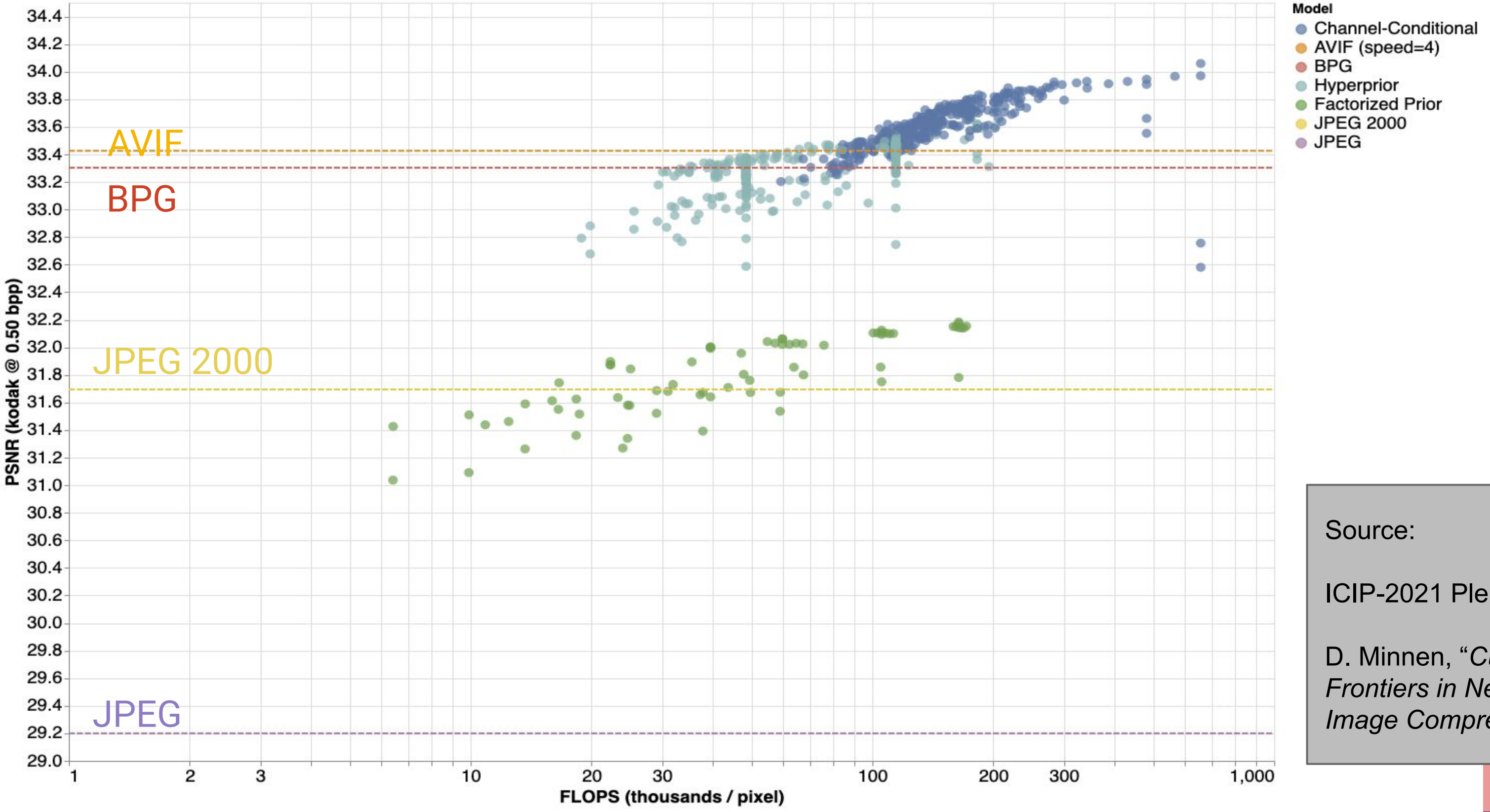
Adapted from:

ICIP-2021 Plenary

D. Minnen, "*Current Frontiers in Neural Image Compression*"

Introduction

Neural Image Codecs



Source:

ICIP-2021 Plenary

D. Minnen, "*Current Frontiers in Neural Image Compression*"

Introduction

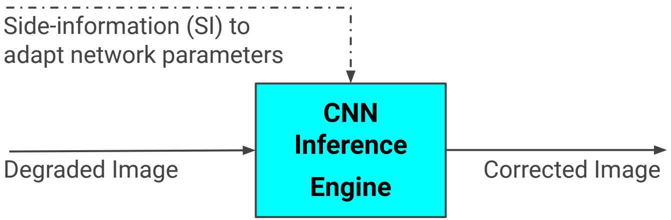
Hybrid conventional-AI codecs

- Hybrid AI codecs
  - Neural (CNN based) in-loop filtering seems to be the most promising
    - ~ **5K** MAC/pixel: 3-4% coding gain [roughly equivalent to a full AV1 decoder]
    - ~ **15-30K** MAC/pixel: 5-6 % coding gain
    - ~ **100K** MAC/pixel: 7-8% coding gain
    - ~ **500K-1M** MAC/pixel: 9-10% coding gain
  - INTRA, INTER_INTRA prediction has potential but so far seems lower
- For AVM:
  - We have focused mostly on Neural in-loop filtering (Out-of-loop is also on the table)
  - Need to get to a much lower MAC/pixel than the above
    - Order of ~ **500-600** Mac/pixel

Introduction

Instance-adaptivity

- How to achieve some of the coding gain through simpler means?
- The key is **instance-adaptivity**
  - Neural network parameters can change from frame to frame, and also within a frame based on content characteristics - overfit the network for a given instance
  - Exploit the fact that we can signal information in the bit-stream to convey the network adaptation.
  - Inference architecture per instance should remain lightweight and common.
- CNN in-loop-restoration with instance-adaptivity



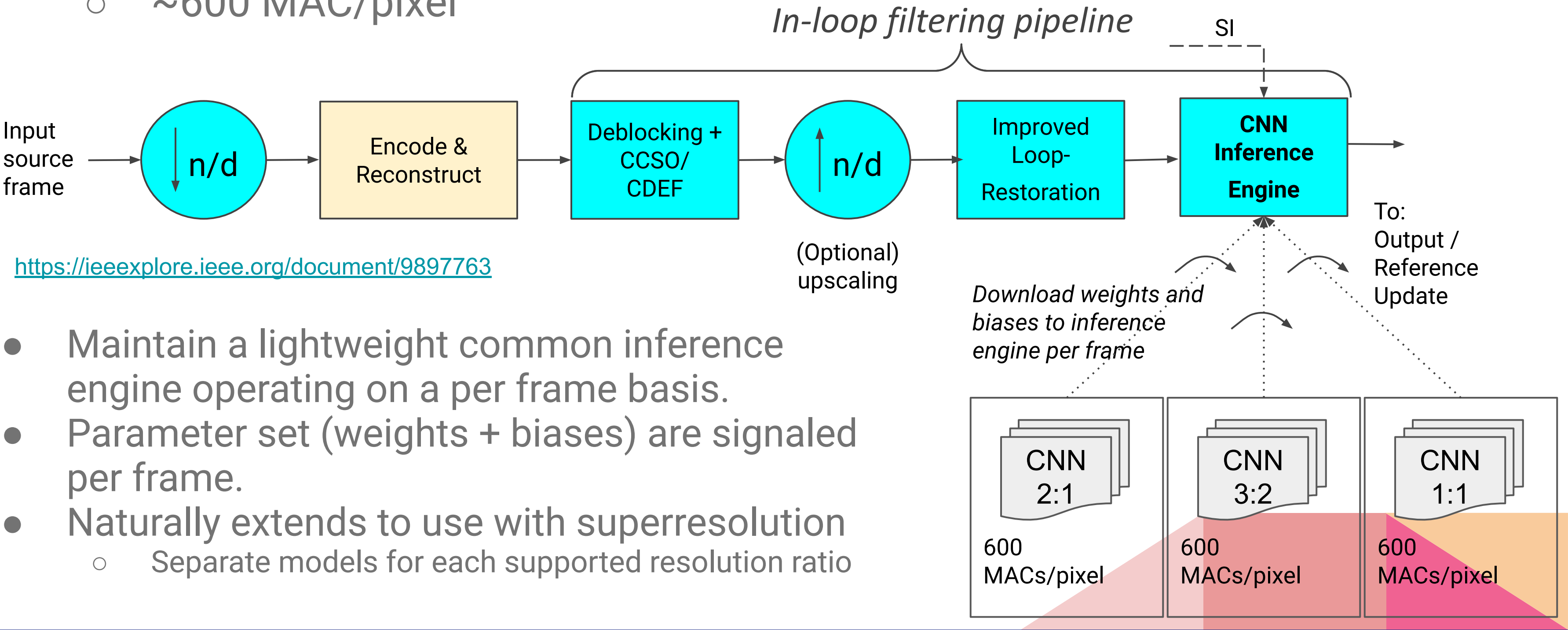
CNN based In-loop Restoration

Switchable Models

CNN Based In-loop Restoration

Switchable CNNs [Framewise Adaptivity]

- A lightweight common inference engine operating on a per frame basis.
    - ~600 MAC/pixel



https://ieeexplore.ieee.org/document/9897763

- Maintain a lightweight common inference engine operating on a per frame basis.
- Parameter set (weights + biases) are signaled per frame.
- Naturally extends to use with superresolution
    - Separate models for each supported resolution ratio

Guided CNN

CNN Based In-loop Restoration

Guided CNN [Within-frame adaptivity]

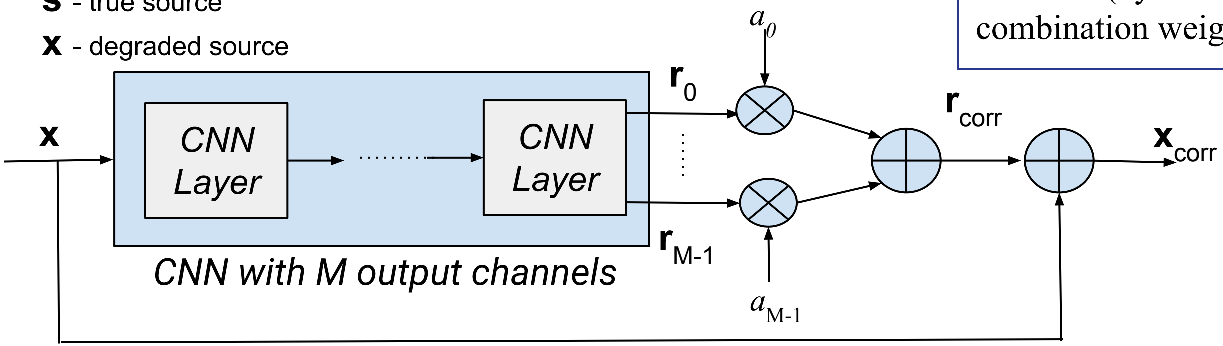
- To achieve further instance adaptivity within a frame, we need to have a mechanism to modify parameters of a neural network within a frame, using a block-level signaling mechanism.
  - Constrain the adaptation to only happen at the "last layer"
- Need to achieve a wide range of trade-offs between rate needed to signal the adaptations and distortion.
- Enter Guided CNN
  - A Convenient mechanism to achieve these objectives
  - A generalization of CNN followed by ALF

CNN Based In-loop Restoration

Guided CNN [Within-frame adaptivity]

- Guided CNN produces M (> 1) output channels instead of 1.
  - So far we have only really explored M = 2 output channels
- Final output is a weighted combination of M outputs, with signaled weights
  - Weights (a_0, a_1, ... a_{M-1}) are signaled per quadtree decomposed blocksize
  - Training loss function is modified to account for the best linear combination in a least squares sense

s - true source
x - degraded source



Evaluate (by least-squares) & explicitly signal linear combination weights: (a_0, a_1, …) of output channels

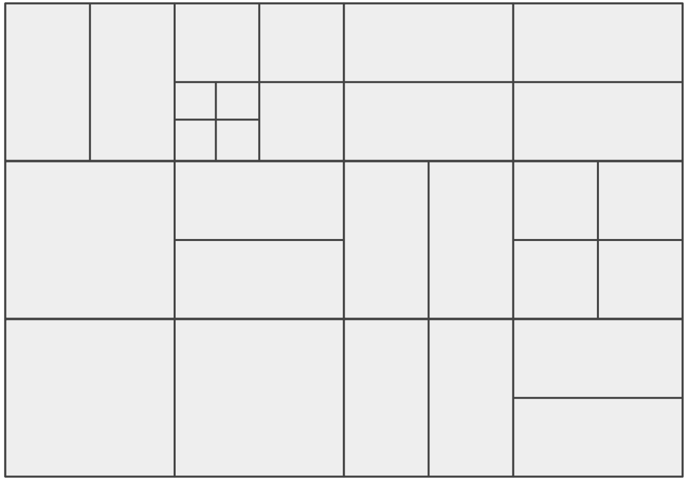
Training loss modification:
Least squares solution for weights:
$[a_0\ a_1\ \ldots\ a_{M-1}]^T = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{d}$,
where $\mathbf{R} = [\mathbf{r}_0\ \ \mathbf{r}_1\ \ldots\ \mathbf{r}_{M-1}]$
$\mathbf{d} = \mathbf{s} - \mathbf{x}$
Then loss $e = |\mathbf{d}|^2 - \mathbf{d}^T\mathbf{R}(\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{d}$

https://ieeexplore.ieee.org/document/10078282

CNN Based In-loop Restoration

Guided CNN [Within-frame adaptivity]

- Signaling of weights: $(a_0, a_1, \ldots)$ is crucial for efficiency
- Use a quad-tree or similar block partitioning structure to signal the weights
- Achieves varying trade-offs between signaled rate and distortion

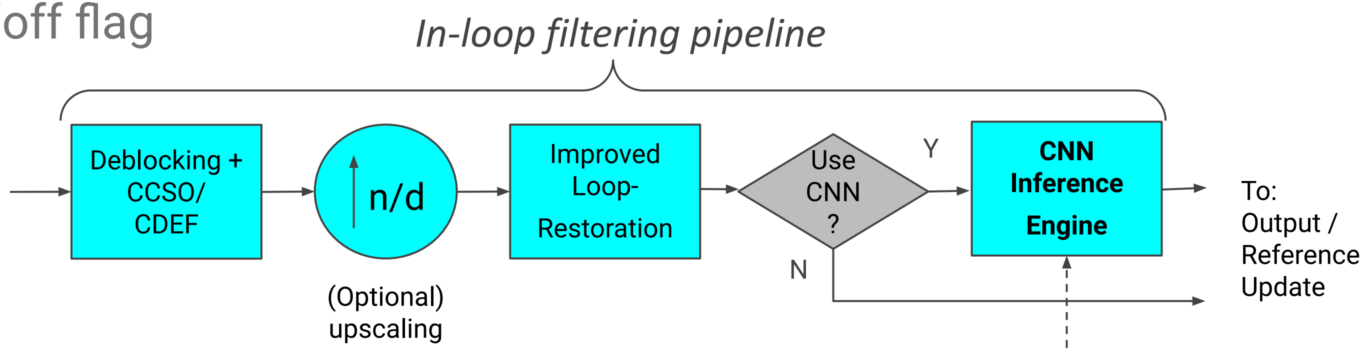


Overall Framework

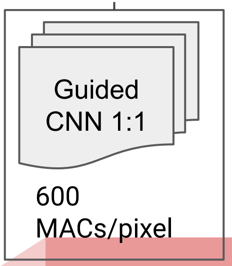
CNN Based In-loop Restoration

Overall Framework

- Frame-level on/off flag



- Multiple Guided CNN models available sharing common architecture
- Choose one Guided CNN model per frame using:
  - *Model Bucket:* Derived implicity from frame QP, frame type
  - *Within Model Bucket Index*: explicitly signaled to indicate one of a few available models within the bucket
- Apply chosen guided CNN or none to produce output frame



CNN Based In-loop Restoration

Overall Framework

- Guided CNN specifics:
  - Specific Model architecture inspired from U-Net: ~**600 MACs/pixel**
    - 2D convolutions replaced by **depthwise-separable** convolutions
    - Downscaling using a convolution layer with stride 2
    - Upscaling using a transpose convolution layer and/or depth-to-space with stride 2
  - **1 channel input**
  - **2 channel output** to use the Guided CNN method with M = 2
    - 2 channels linearly combined to generate 1 output correction channel
  - Single-level quad/bi-tree partitioning for weight signaling
    - Each square block is further partitioned once using NONE, HORZ, VERT or SPLIT
  - Total #models for 1:1 case:
    - 6 QP ranges x 2 frame types (INTER/INTRA) x 3 models per bucket = 36

Results

CNN Based In-loop Restoration

Results

- Baseline: AVM v-7 anchor on Common Test Conditions
- One of the **smallest** CNN models ever used -approaching the trade-off of a conventional tool.
- Notable points:
  - VMAF gains are higher
  - Higher resolution gains are higher

(1) float32 Models

Config	Overall (w/o B2)		Class A1_4K	
	PSNR YUV	VMAF	PSNR YUV	VMAF
AI	-1.48%	-3.78%	-2.12%	-5.93%
RA	-1.21%	-2.34%	-1.90%	-4.55%

(2) int10 Models

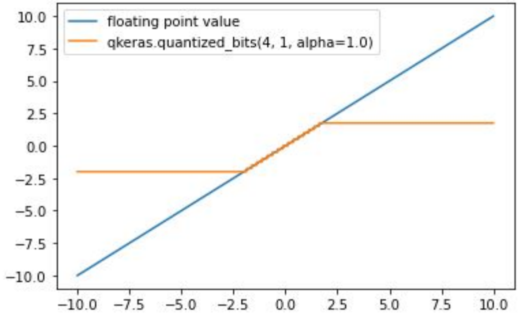
Config	Overall (w/o B2)		Class A1_4K	
	PSNR YUV	VMAF	PSNR YUV	VMAF
AI	-1.44%	-2.89%	-2.10%	-4.79%
RA	-1.17%	-2.74%	-2.21%	-5.02%

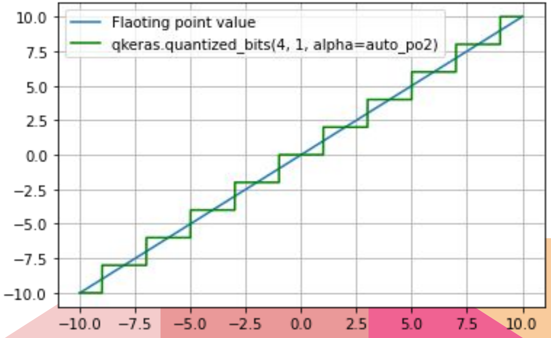
Hardware Design & Analysis

CNN Based In-loop Restoration

Model Integerization

- Quantization - crucial for fixed point hardware implementation
- HW complexity reduced by quantizing different aspects of the model:
  - Weights Quantization: Quantize the weights and any other storable params across layers.
  - Activation Quantization: Quantize inter-connects between model layers with activations quantization.
- Maintain performance while reducing complexity in two-ways:
  - Quantization aware training (QAT):
    - Train model weights while being aware of quantization.
    - Open sourced QKeras framework used for QAT.
  - Heterogeneous quantization:
    - Individual layers are optimally quantized to maintain model accuracy

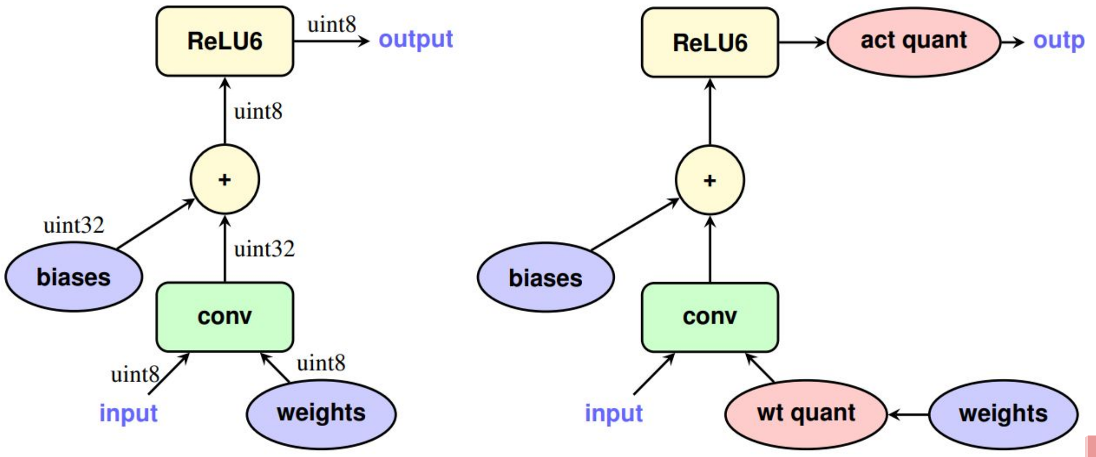




CNN Based In-loop Restoration

Model Integerization

- Quantization Aware Training:
  - Simulated Quantization during Training



Inference

Training

CNN Based In-loop Restoration

Model Integerization

- Optimize bit allocation for weights and activations on a per layer basis
- Example quantization schema:
  - **All layer activations and most weights are allocated 10-bits**
  - 16-bits allocated to pointwise layer weights in the UNet encoder (x4 layers)
  - 20-bits allocated to transposed conv2D layer weights (x4 layers)
- Different architecture study.

CNN Based In-loop Restoration

Hardware Analysis

- 10-bit quantized models are implemented with TSMC 5nm technology

- The HW synthesis is performed based on pre-set throughput requirements

Clock Frequency (GHz)	1.200
Pixel Rate (pixel/clk)	1.000
64x64 Block Rate (blk/sec)	292,968.75
4K FPS (frame/sec)	144.68

CNN Based In-loop Restoration

Hardware Analysis

Synthesis Results

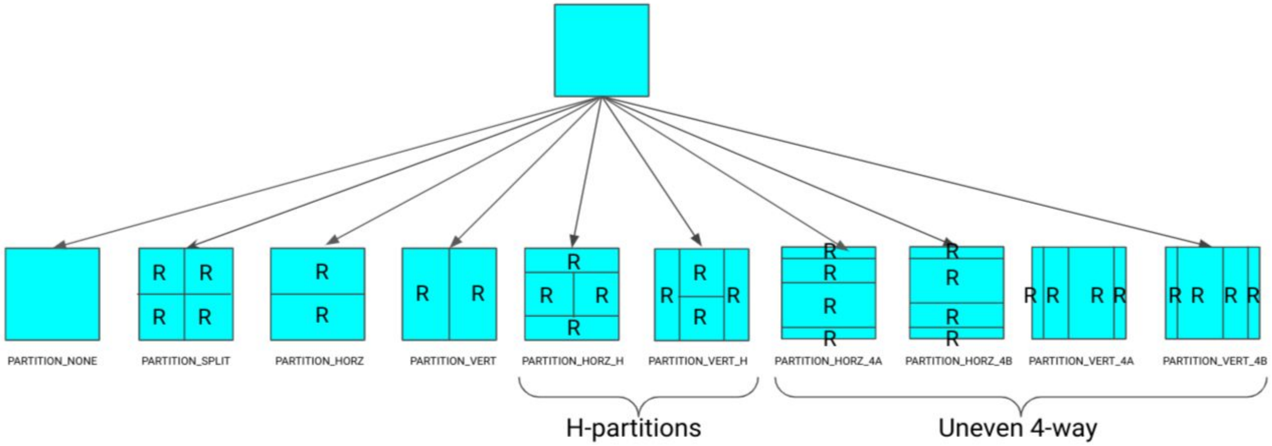
	Logic gates	SRAM (bit)	Total Area in #Gates
Logic Design	1,362,433	0	1,362,433
FIFO Connection	170,808	0	170,808
Internal Storage	121,656	118,504	240,160
Total (gates)	1,654,897	118,504	**1,773,401**
Total (um^2)	53,172	3,808	56,979

ML Based Encoder Optimizations

Partition Search

ML Based Encoder Optimizations

AVM partitioning scheme

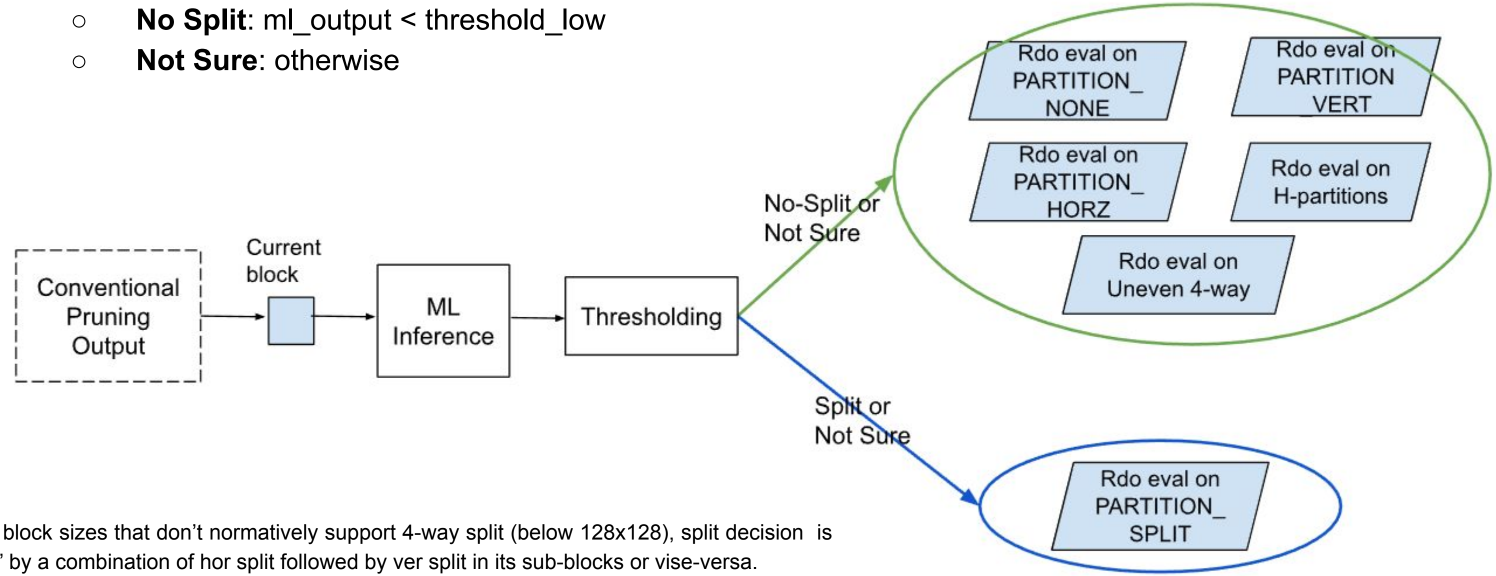


- Recursive partitioning scheme in AVM is expensive!
  - Brute-forth search + ad hoc pruning
  - AVM anchor_v7 vs AV1:
    - 38x enc-time in AI ,23x enc-time in RA
- Long pole:
  - In lower QP, partition search can reach leaf nodes, encoding time could grow ↑10x (qp 110 vs 235)

ML-based Pruning, 4-Way Split Detection

- **ML task**: Predicting if a given block is a 4-way split (both hor and ver split)

- **Ternary Pruning Decision**:
    - **Split**: ml_output > threshold_high
    - **No Split**: ml_output < threshold_low
    - **Not Sure**: otherwise



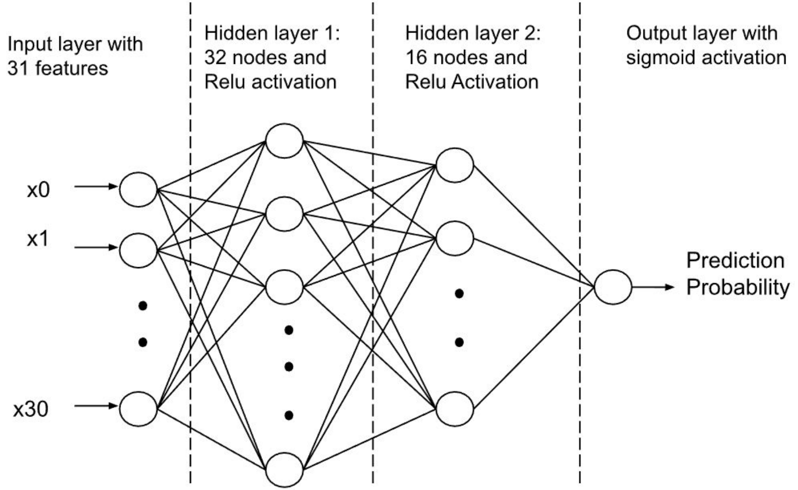
Note: For block sizes that don't normatively support 4-way split (below 128x128), split decision is 'emulated' by a combination of hor split followed by ver split in its sub-blocks or vise-versa.

ML-based Pruning, Small DNN Architecture (Intra)

- Small DNN with 2 hidden layers.
  - 4 separately trained DNNs invoked for block sizes
    - 64x64 / 32x32 / 16x16 / 8x8
  - Other block size
    - No ML, and encoder remains unmodified.

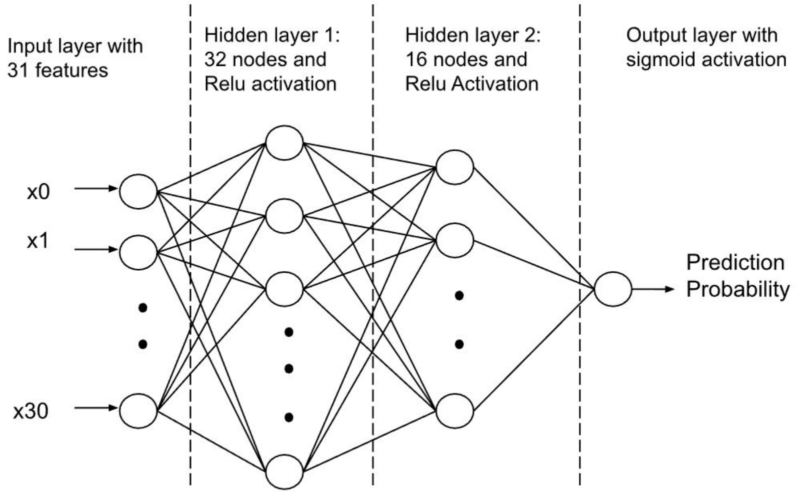
- Compute 37 Input Features
  - For 13 primary primary intra prediction modes
  - For the current / 4 sub blocks
  - SSE and Variance (VAR) of the top 3 modes
  - QP, neighbor size and availability information for the block

- 30% long pole speed up with 0.05% loss



ML-based Pruning, Small DNN Architecture (Inter)

- Small DNN with 2 hidden layers.
  - 4 separately trained DNNs invoked for block sizes
    - 64x64 / 32x32 / 16x16 / 8x8
  - Other block size
    - No ML, and encoder remains unmodified.
- 31 Input Features:
  - For the current / 4 sub blocks
  - NNZ (# of nonzero coefficients)
  - NZMAX (maximum level of nonzero coefficient)
  - PSNR/ SATD
  - Magnitude and angle of the motion vector
  - RD multiplication
- 35% long-pole speed up with 0.06% loss



Conclusion

Conclusion

Summary

- Coding Tools
    - Constraints in prevalent video decoder HW architectures make incorporating AI based tools extremely challenging
    - We have taken the first steps into bringing neural AI tools into a mainstream video codec at complexity approaching that of a conventional tool
    - Developed one of the smallest neural models reported in literature providing 1+ % gain, combining multiple switchable models/frame with guided CNN within frame
    - WIP - improving gains and reducing hardware footprint further
- Encoder Optimizations
    - ML methods shown to be useful in bypassing complex RD search in modern codecs
        - Partition search speed-up
        - Many more opportunities

Thank You