Compressed CNN for Green Versatile Video Coding

IETR - Vaader
Daniel Ménard
Wassim Hamidouche

Inria - Cairn
Olivier Sentieys
Silviu-Ioan Filip

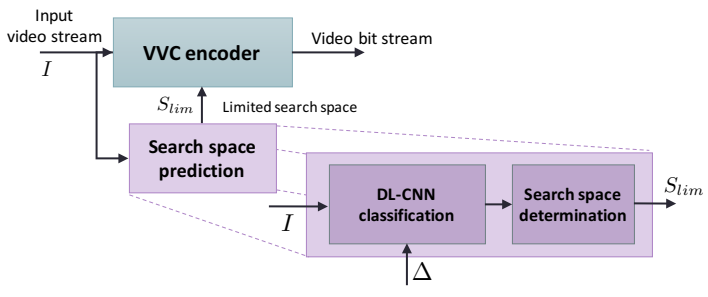
Context

- High amount of video in internet traffic
  - x4 from 2016 to 2021
  - 81% of the overall internet traffic
- New video services and format
  - VOD, webTV, video sharing, live streaming, …
  - 8K, HFR, 360° video

↪ Needs for more efficient video coding standard
↪ New MPEG standard for 2021: VVC

Objectives

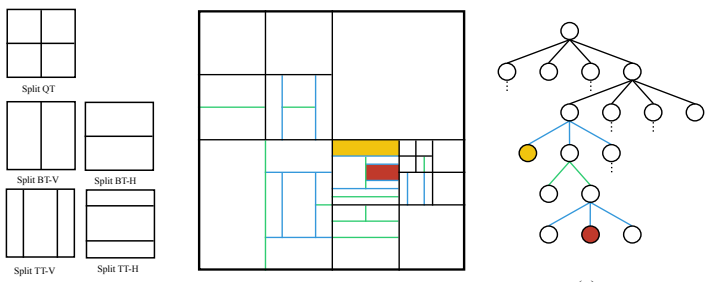
- Encoding complexity: up to x40
  - Huge increase of the RDO search space
    ↪ Need for drastic complexity reduction techniques to enable real time encoders
- Complexity reduction of encoding process
  - Prediction based on CNN



- Contributions
  1. Complexity reduction of encoding process
     - Deep learning with CNN to increase performance
  2. Techniques to reduce the CNN inference cost
     - Limitation of CNN inference overhead

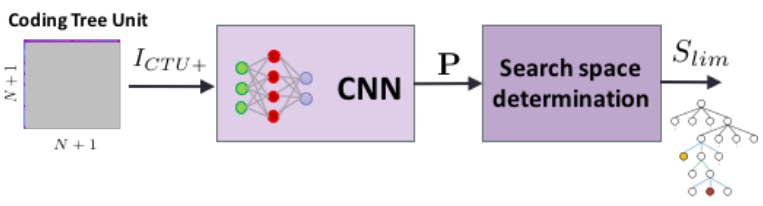
CTU Tree partitioning

- New partitioning tree: QT – BT - TT

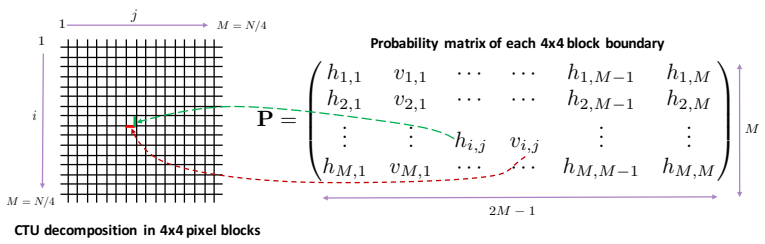


Search space reduction scheme

- Prediction of probability matrix



- Probability matrix of each block boundary



Results for complexity reduction

- Quality degradation – complexity reduction
  - Complexity reduction
    - Execution time reduction
  - Video quality reduction
    - BD-PSNR: Bjontegaard-Delta Peak-Signal-to-Noise-Ratio
    - BD-BR: Bjontegaard-Delta bit-rate (bit rate increase)

Video Class	BD-PSNR (dB)	BD-Rate (%)	Complexity reduction (%)
A1	-0,06	2,42	55,24
A2	-0,04	1,29	46,61
B	-0,06	1,42	50,32
C	-0,10	1,77	31,27
D	-0,12	1,78	27,15
E	-0,10	2,22	39,78
Mean	-0.08	1.78	41.37

Neural network approximation

- Much work on compressing feed-forward neural networks for inference
  - Weight & activation quantization
  - Network compression (i.e., smaller and or structured architectures)
- Make CNNs (and DNNs in general) more friendly for edge devices
  - Reduce memory pressure for storing networks
  - Improved energy efficiency

↪ Need for efficient methods and tools to explore neural network approximation/compression design space for inference

Quantization methodology

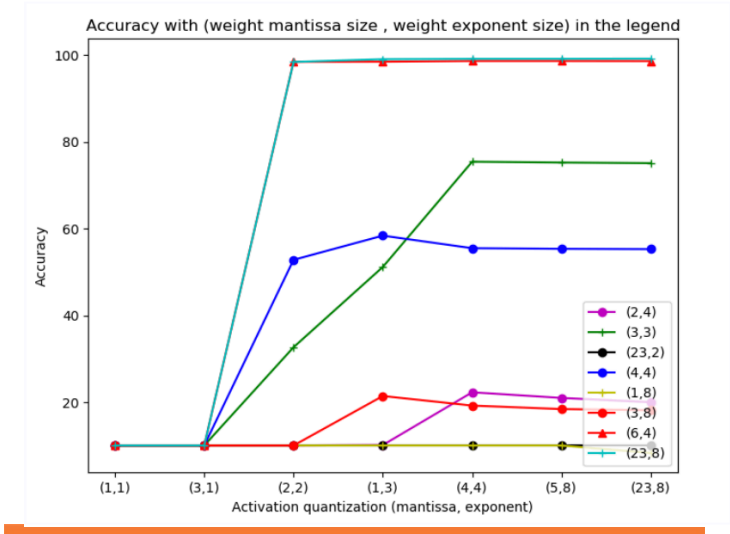
- Focus on low-precision floating-point formats for all operations inside the CNN
  - Add support for custom floating-point arithmetic inside the N2D2 framework for DNN design
  - Work on an automatic method for choosing appropriate floating-point quantization formats at each layer of the network

Custom floating-point support

- Different quantization formats at each layer
- Explore sensitivity of each layer to quantization effects

↪ Experiments on LeNet for the MNIST dataset

FormatLayer	conv1	conv2	conv3	fc1	fc2	All layers
(1, 1)	87.28%	24.11%	11.83%	11.83%	12.50%	11.83%
(23, 1)	91.67%	32.70%	11.83%	11.83%	14.62%	11.83%
(1, 8)	92.30%	89.73%	94.75%	99.33%	75.11%	31.81%
(2, 3)	98.67%	98.88%	98.33%	99.33%	97.10%	72.10%
(3, 3)	99.33%	99.00%	99.00%	99.22%	99.22%	97.32%
(4, 4)	99.22%	99.22%	99.33%	99.22%	99.22%	99.22%

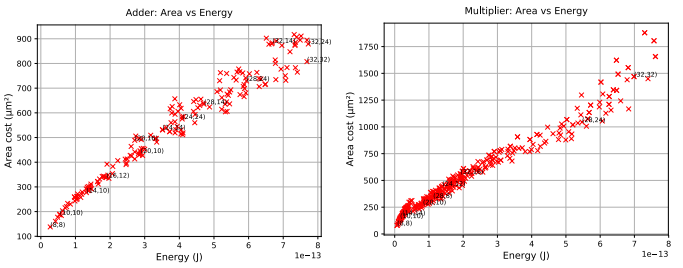


Energy efficiency improvement

Energy vs. Area of custom floating-point adders and multipliers

- Reducing computations from 32 down to 10 bits provides gains of more than 15× in energy and more than 8× in area

↪ Opens up interesting opportunities in making deep-learning inference more efficient



Automatic exploration of different quantization formats

- Initial exploration in this direction:

$$D^\star = \text{argmin}_{D \in \{D_0, D_1, \ldots, D_n\}} d_{D,\max}$$

$$d_{D,\max} = \max_{y \in H} d_{D,y}$$

$$d_{D,y} = \min_{x \in \Delta_D} ||x - y||$$

- $D_0, D_1, \ldots, D_n$ – available quantization formats
- H – list of network weights
- $d_{D,y}$ – distance between y and its closest quantization in format D
- Δ_D – set of all possible values in the format D
- $d_{D,\max}$ – maximal distance between a weight and its approx. in D