Fast Partitioning for VVC Intra-Picture Encoding With a CNN Minimizing the Rate-Distortion-Time Cost

Gerhard Tech, Jonathan Pfaff, Heiko Schwarz, Philipp Helle,
Adam Wieckowski, Detlev Marpe, and Thomas Wiegand

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC
- Requirement: Encoder selects efficient coding modes
  - Rate-distortion optimization (RDO)
    - Encode a block B with different partitioning, prediction, and transform mode combinations
    - Select the modes providing the minimal Lagrangian rate-distortion (RD) cost
  - Problem: Limited encoding time; not all combinations can be tested
  - → Some modes must be skipped without testing

Motivation

- Versatile Video Coding (VVC)
  - New coding tools e.g. for intra-picture coding:

    multi-type tree (MTT), intra sub-block partitions (ISP), matrix-based intra prediction (MIP), extended number of directional modes, multi-reference line (MRL), low-frequency non-separable transform (LFNST), multiple transform select (MTS) ...

  - 50% bit rate reduction compared to HEVC
- Requirement: Encoder selects efficient coding modes
  - Rate-distortion optimization (RDO)
    - Encode a block B with different partitioning, prediction, and transform mode combinations
    - Select the modes providing the minimal Lagrangian rate-distortion (RD) cost
  - Problem: Limited encoding time; not all combinations can be tested

  ➔ Some modes must be skipped without testing
- Which modes should the encoder skip?
  - Optimally: Reduce encoding time, while not increasing the RD cost

  ➔ Subject of this presentation: Doing this by skipping MTT partitioning modes

Outline

- How can the the partitioning be restricted?
  - VVC partitioning
  - Parameters

Outline

- How can the the partitioning be restricted?
  - VVC partitioning
  - Parameters
- How can the parameters be selected optimally?
  - Theoretical background

Outline

- How can the the partitioning be restricted?
  - VVC partitioning
  - Parameters
- How can the parameters be selected optimally?
  - Theoretical background
- How to estimate the optimal parameters practically?
  - CNN
  - Training data generation
  - Loss function in training

Outline

- How can the the partitioning be restricted?
  - VVC partitioning
  - Parameters
- How can the parameters be selected optimally?
  - Theoretical background
- How to estimate the optimal parameters practically?
  - CNN
  - Training data generation
  - Loss function in training
- How do the estimated parameters perform in encoding?
  - Evaluation

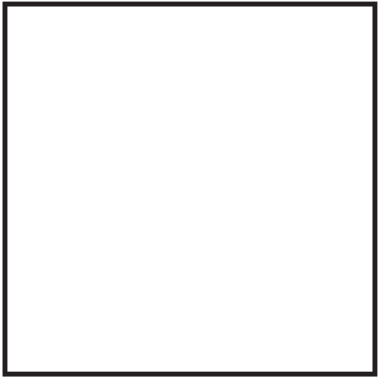
Outline

- How can the the partitioning be restricted?
  - VVC partitioning
  - Parameters
- How can the parameters be selected optimally?
  - Theoretical background
- How to estimate the optimal parameters practically?
  - CNN
  - Training data generation
  - Loss function in training
- How do the estimated parameters perform in encoding?
  - Evaluation
- Summary and Conclusion

VVC Partitioning

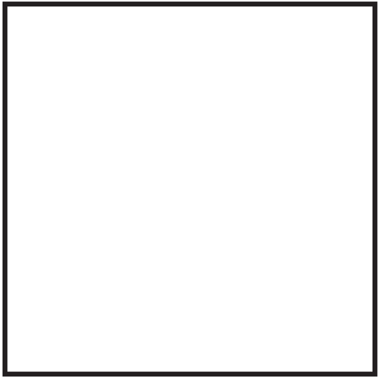
Partitioning Restrictions

- Combined quad and a multi-type tree



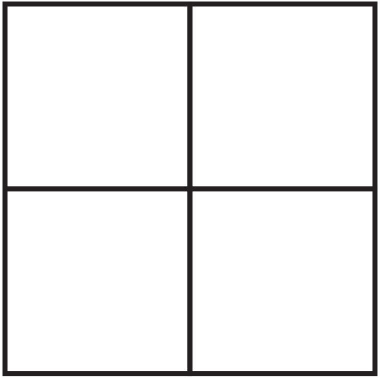
VVC Partitioning

Partitioning Restrictions



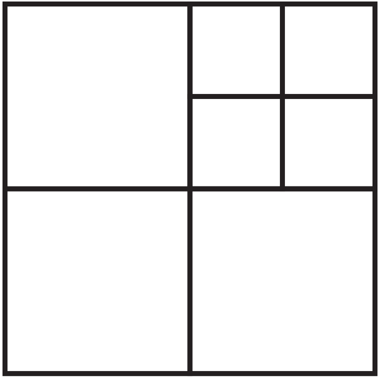
- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively

Partitioning Restrictions



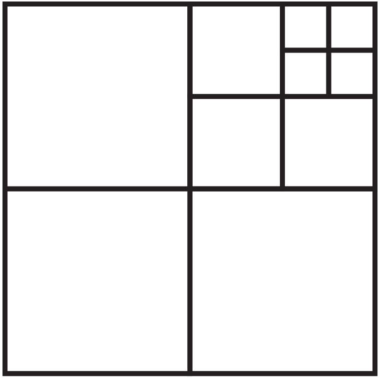
- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively

VVC Partitioning



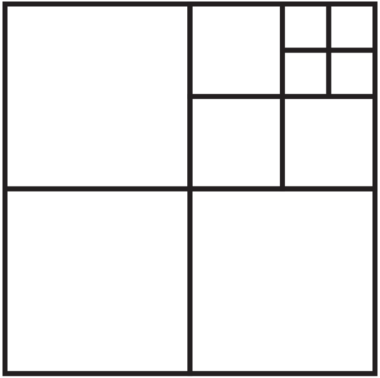
- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively

Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively

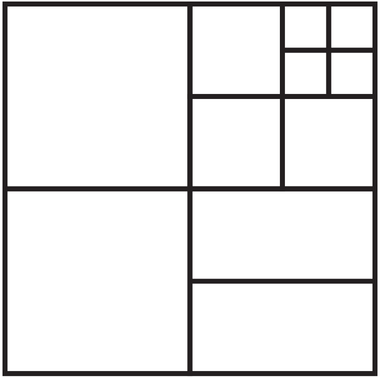
Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:

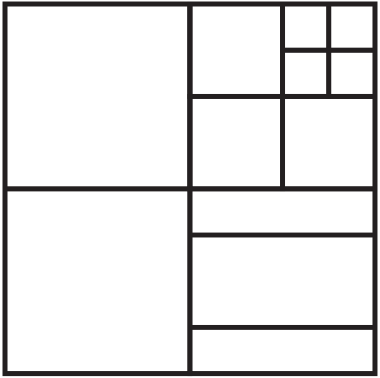
VVC Partitioning

Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Horizontal binary

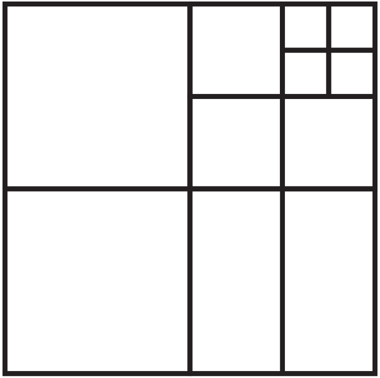
Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Horizontal ternary

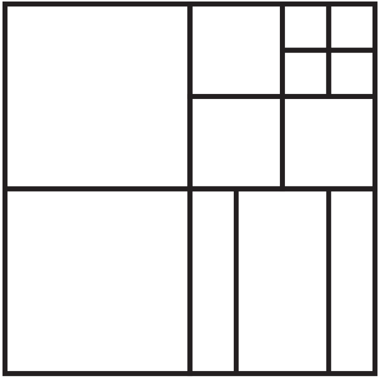
VVC Partitioning

Partitioning Restrictions



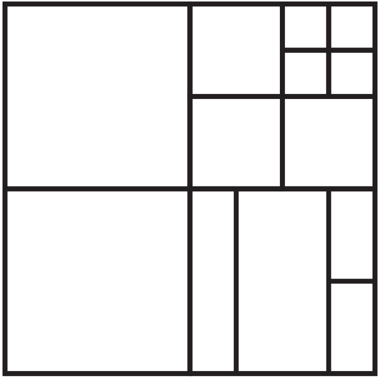
- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Vertical binary

Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Vertical ternary

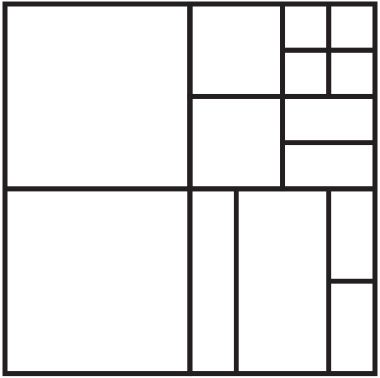
Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Recursively

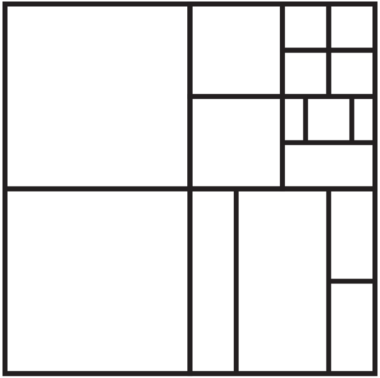
VVC Partitioning

Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Recursively

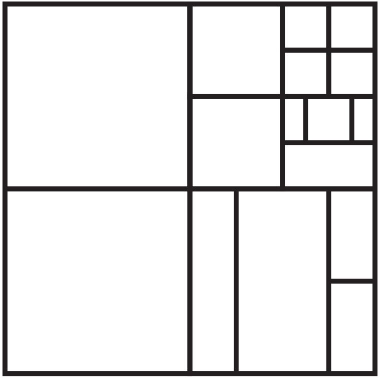
Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Recursively

VVC Partitioning

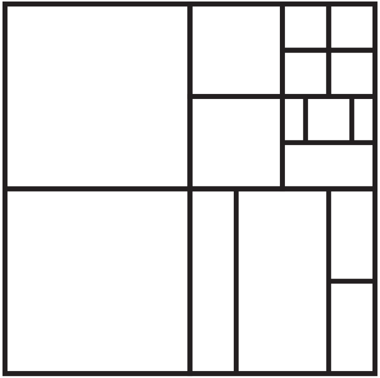
Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Recursively
- Various options to adapt to the content
  - However: Most options irrelevant

VVC Partitioning

Partitioning Restrictions



- Combined quad and a multi-type tree
- Quad tree:
  - Quad splits
  - Recursively
- Multi-type tree
  - Can start at Qt leaf nodes
  - Directional splits:
  - Recursively
- Various options to adapt to the content
  - However: Most options irrelevant

→ Idea:
  - Skip partitioning modes based on the block content without testing
  - Introduce parameters controlling which modes are skipped

Parameters

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$

Parameters

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters
  $$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$
- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node

Parameters

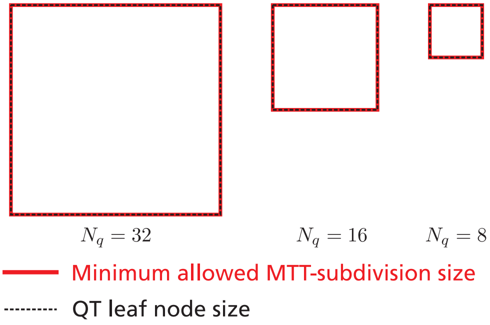
- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$

- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q / 2^{p_H}$ or
  - a height less than $N_q / 2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$



Gerhard Tech

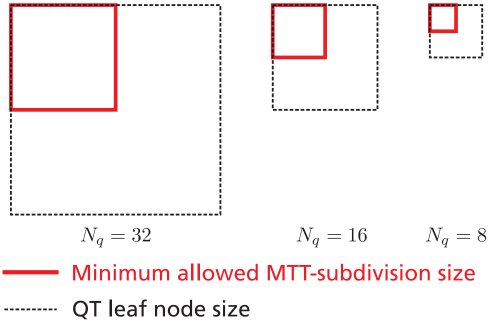
Fraunhofer HHI

Parameters

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters
  $$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$
- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



Parameters

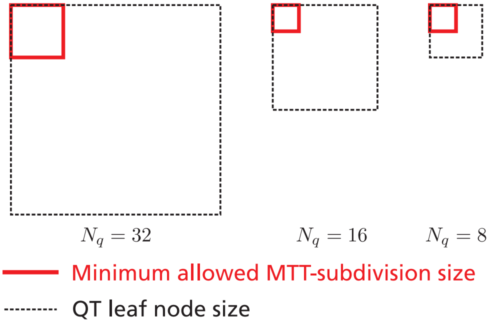
- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$

- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q / 2^{p_H}$ or
  - a height less than $N_q / 2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$



Parameters

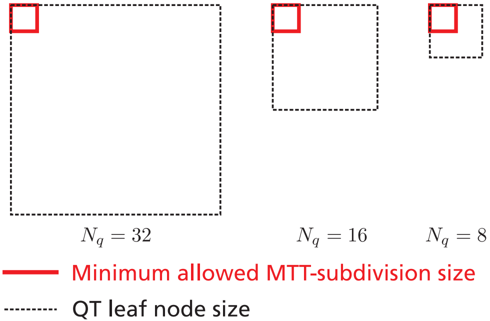
- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$

- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node
- Value range: $p_H, p_V \in \{0, 1, 2, 3\}$

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

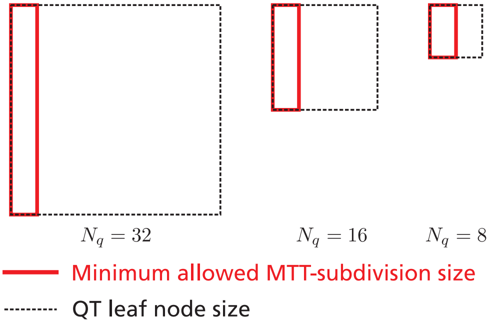


Parameters

Partitioning Restrictions

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters
  $$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$
- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node
- Value range: $p_H, p_V \in \{0, 1, 2, 3\}$

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$



Parameters

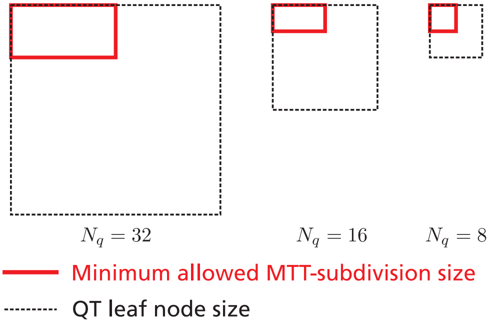
- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$

- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node
- Value range: $p_H, p_V \in \{0,1,2,3\}$

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



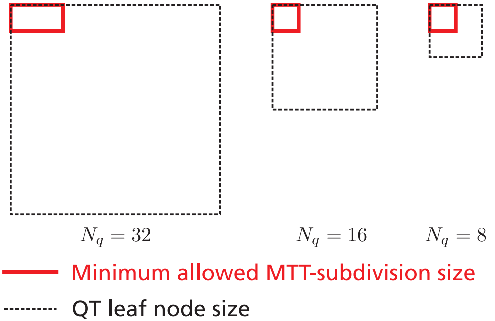
Gerhard Tech

Parameters

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters
  $$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$$
- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node
- Value range: $p_H, p_V \in \{0, 1, 2, 3\}$

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$



Parameters

- Idea: Limit the size of tested MTT partitionings for 32×32 blocks with two parameters

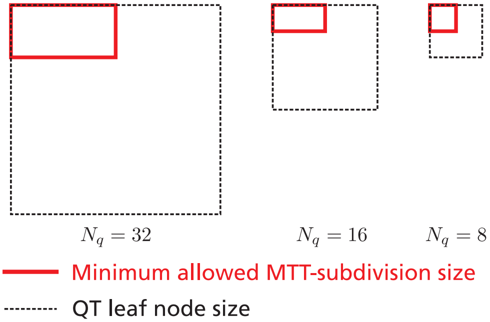
  $\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix}$

- Skip testing an MTT split if any resulting sub-block would have
  - a width less than $N_q/2^{p_H}$ or
  - a height less than $N_q/2^{p_V}$
  - with N_q denoting the size of the MTT's quad-tree leaf node
- Value range: $p_H, p_V \in \{0, 1, 2, 3\}$
- The parameters to control the size and orientation of tested MTT partitionings

→ Adaptation to block content possible

Partitioning Restrictions

$$\mathbf{P} = \begin{pmatrix} p_H \\ p_V \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



Gerhard Tech

Optimal Parameter Selection

- How to select $\mathbf{P}$ for a block B optimally?

Optimal Parameter Selection

- How to select $\mathbf{P}$ for a block B optimally?
- Objective:
  - Reduce the encoding time $T(\mathbf{P})$
  - While maintaining the RD performance, i.e. the RD cost $J(\mathbf{P})$

Optimal Parameter Selection

- How to select $\mathbf{P}$ for a block B optimally?
- Objective:
  - Reduce the encoding time $T(\mathbf{P})$
  - While maintaining the RD performance, i.e. the RD cost $J(\mathbf{P})$
- Theoretically solved—Generalized Lagrangian multipliers:
  - Per block B, select the parameter $\mathbf{P}$ that produces the minimal rate-distortion-time cost K:

$$K(\mathbf{P}) = \underbrace{D(\mathbf{P}) + \lambda \cdot R(\mathbf{P})}_{\text{rate-distortion cost } J(\mathbf{P})} + \underbrace{\mu \cdot T(\mathbf{P})}_{\text{time cost}}$$

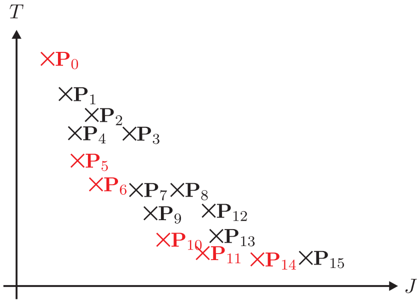
    - D distortion; R bits; T encoding time
    - λ and μ Lagrange multipliers
  - When $K(\mathbf{P})$ is independent for different B, block-wise minimization also minimizes the overall RDT cost

Optimal Parameter Selection

- How to select $\mathbf{P}$ for a block B optimally?
- Objective:
  - Reduce the encoding time $T(\mathbf{P})$
  - While maintaining the RD performance, i.e. the RD cost $J(\mathbf{P})$
- Theoretically solved—Generalized Lagrangian multipliers:
  - Per block B, select the parameter $\mathbf{P}$ that produces the minimal rate-distortion-time cost K:

$$K(\mathbf{P}) = \underbrace{D(\mathbf{P}) + \lambda \cdot R(\mathbf{P})}_{\text{rate-distortion cost } J(\mathbf{P})} + \underbrace{\mu \cdot T(\mathbf{P})}_{\text{time cost}}$$

    - D distortion; R bits; T encoding time
    - λ and μ Lagrange multipliers
  - When $K(\mathbf{P})$ is independent for different B, block-wise minimization also minimizes the overall RDT cost

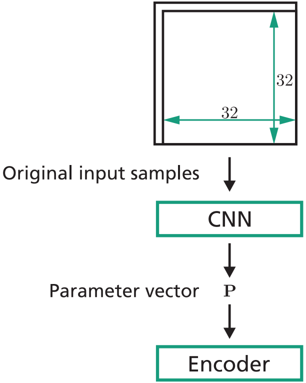


Basic Idea

- Practical Problem
  - Lagrangian method requires $J(\mathbf{P})$ and $T(\mathbf{P})$ for all parameter combinations $\mathbf{P}$
  - Not known and testing beforehand makes no sense

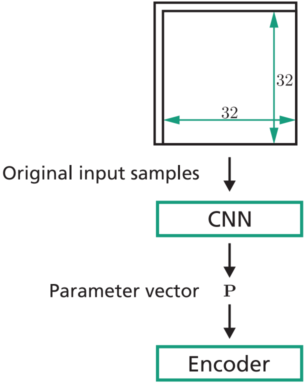
Basic Idea

- Practical Problem
  - Lagrangian method requires $J(\mathbf{P})$ and $T(\mathbf{P})$ for all parameter combinations $\mathbf{P}$
  - Not known and testing beforehand makes no sense
- Solution
  - Use a CNN to derive the parameter $\mathbf{P}$
  - Train the CNN such that $\mathbf{P}$ minimize the RDT cost



Basic Idea

- Practical Problem
  - Lagrangian method requires $J(\mathbf{P})$ and $T(\mathbf{P})$ for all parameter combinations $\mathbf{P}$
  - Not known and testing beforehand makes no sense
- Solution
  - Use a CNN to derive the parameter $\mathbf{P}$
  - Train the CNN such that $\mathbf{P}$ minimize the RDT cost
- Training Approach
  - Training data generation:
    - Perform the Lagrangian method
    - Record the occurring RDT-cost
  - In training:
    - Use the stored RDT cost to compute the training loss

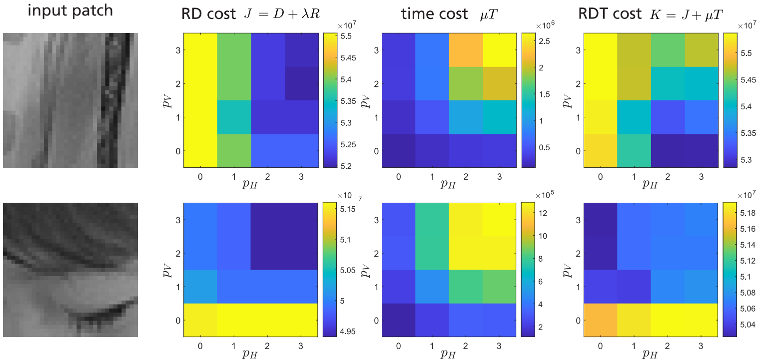


Generation of Training Data

- Encode training sequences, for each 32×32 block B:
  - Encode B with all $\mathbf{P}$
  - Store the occurring RDT cost $K(\mathbf{P})$
  - Select $\mathbf{P}$ providing the minimum $K(\mathbf{P})$ and continue based on its encoder state

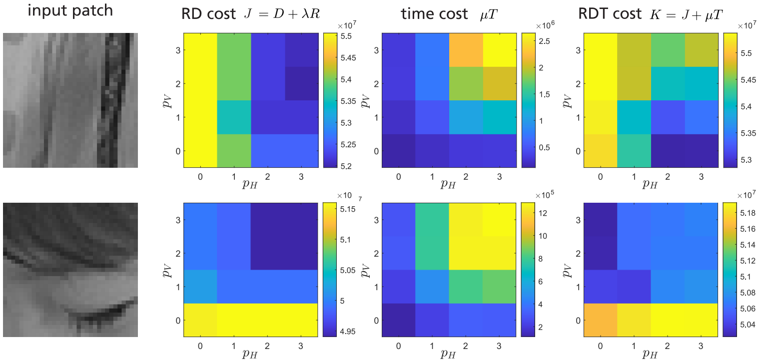
Generation of Training Data

- Encode training sequences, for each 32×32 block B:
  - Encode B with all $\mathbf{P}$
  - Store the occurring RDT cost $K(\mathbf{P})$
  - Select $\mathbf{P}$ providing the minimum $K(\mathbf{P})$ and continue based on its encoder state



Generation of Training Data

- Encode training sequences, for each 32×32 block B:
  - Encode B with all $\mathbf{P}$
  - Store the occurring RDT cost $K(\mathbf{P})$
  - Select $\mathbf{P}$ providing the minimum $K(\mathbf{P})$ and continue based on its encoder state



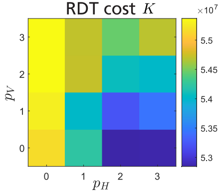
- Lagrange parameters:
  - λ: encoder default for QP
  - μ: determines encoding time

Loss Function in Training

- Objective: Minimize the RDT cost of estimated parameters

Loss Function in Training

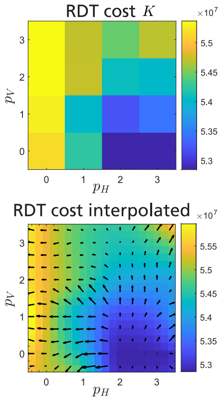
- Objective: Minimize the RDT cost of estimated parameters
- Approach
  - Interpret the two floating point CNN outputs as p_H and p_V
  - Gradient descent method
  - Requires derivatives of loss function with respect to p_H and p_V
  - → Problem: K sampled at integer values; derivatives do not exist



Gerhard Tech

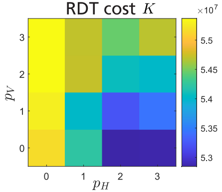
Loss Function in Training

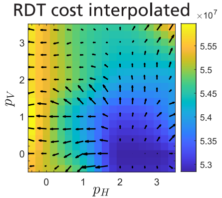
- Objective: Minimize the RDT cost of estimated parameters
- Approach
  - Interpret the two floating point CNN outputs as p_H and p_V
  - Gradient descent method
  - Requires derivatives of loss function with respect to p_H and p_V
  - → Problem: K sampled at integer values; derivatives do not exist
- Solution
  - Model derivatives using an interpolated loss function



Loss Function in Training

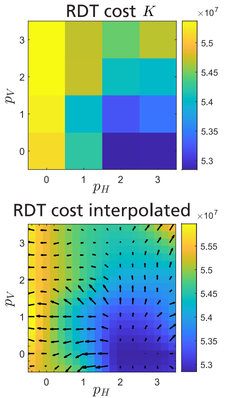
- Objective: Minimize the RDT cost of estimated parameters
- Approach
  - Interpret the two floating point CNN outputs as p_H and p_V
  - Gradient descent method
  - Requires derivatives of loss function with respect to p_H and p_V
  - → Problem: K sampled at integer values; derivatives do not exist
- Solution
  - Model derivatives using an interpolated loss function
- If $0 \leq p_H \leq 3$ and $0 \leq p_V \leq 3$
  - Horizontal and vertical interpolation using cubic hermite B-splines
  - Advantage: shape preserving; produces no local minima





Loss Function in Training

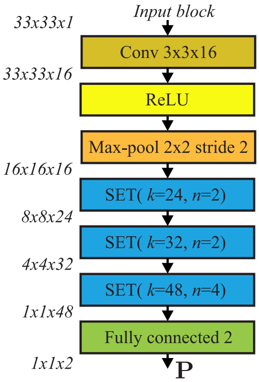
- Objective: Minimize the RDT cost of estimated parameters
- Approach
  - Interpret the two floating point CNN outputs as p_H and p_V
  - Gradient descent method
  - Requires derivatives of loss function with respect to p_H and p_V
  - → Problem: K sampled at integer values; derivatives do not exist
- Solution
  - Model derivatives using an interpolated loss function
- If $0 \leq p_H \leq 3$ and $0 \leq p_V \leq 3$
  - Horizontal and vertical interpolation using cubic hermite B-splines
  - Advantage: shape preserving; produces no local minima
- Otherwise
  - Artificial gradient pointing away from the sampled area
  - Ensures that the gradient descent method converges back



Gerhard Tech

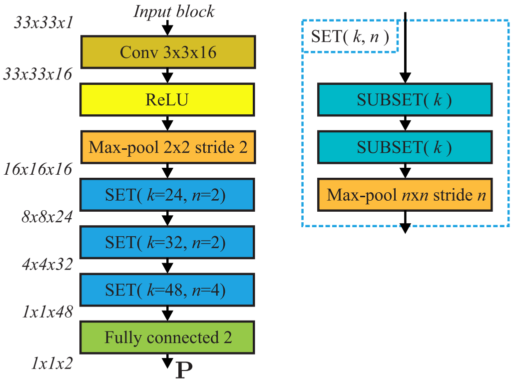
Fraunhofer HHI

The CNN



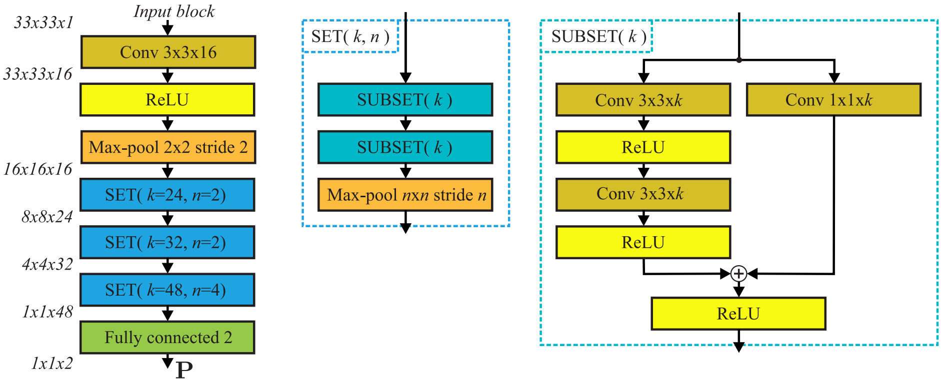
- Corresponds to CNN of Galpin (2019) and Tissier (2020) [inspired by ResNet]

The CNN



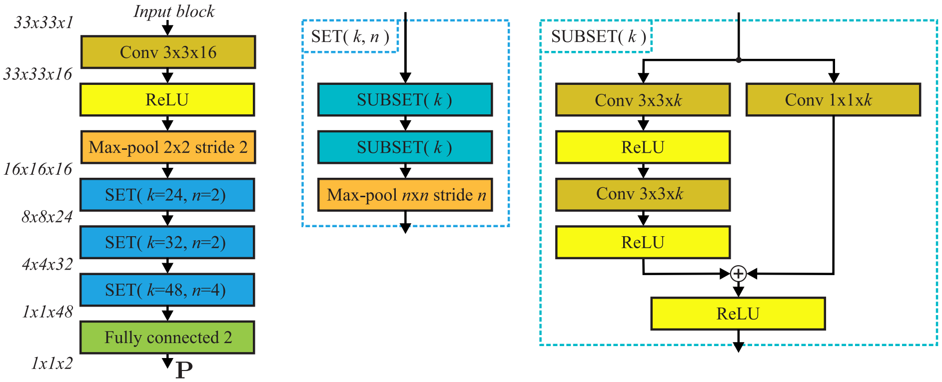
- Corresponds to CNN of Galpin (2019) and Tissier (2020) [inspired by ResNet]

The CNN



- Corresponds to CNN of Galpin (2019) and Tissier (2020) [inspired by ResNet]

The CNN



- Corresponds to CNN of Galpin (2019) and Tissier (2020) [inspired by ResNet]
- Modifications:
  - Max pooling size changed to input 33×33 instead of 65×65 blocks
  - 2 outputs instead of 480

Conditions

- Training
  - 24 million training patches
  - 22 epochs
  - Adam optimizer; Learning rate of 10^{-4}; mini-batch size of 512

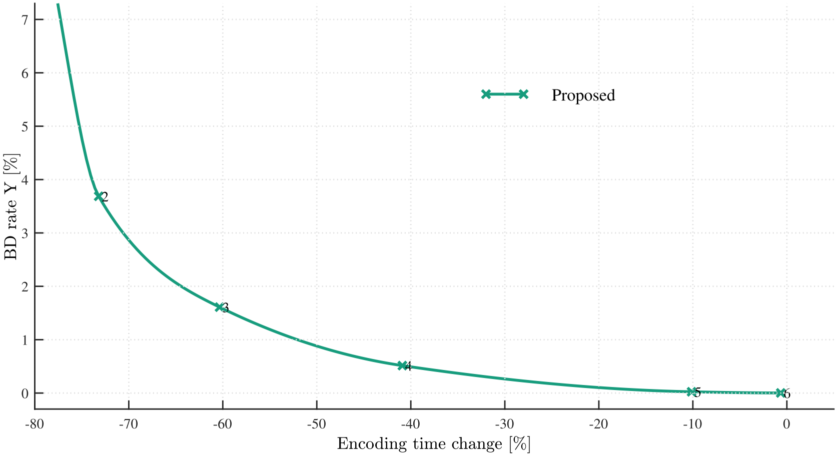
Conditions

- Training
  - 24 million training patches
  - 22 epochs
  - Adam optimizer; Learning rate of 10^{-4}; mini-batch size of 512
- CNNs for 24 test points
  - 6 Encoding time points
  - 4 QPs: 22, 27, 32, and 37
  - An individual μ for each CNN, such that
    - for all QPs of an encoding time point:
    - approximately the same encoding time reduction (in percent)

Conditions

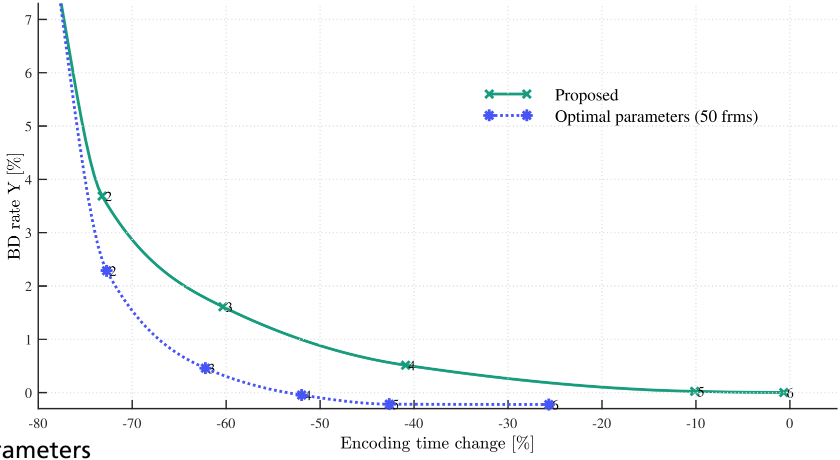
- Training
  - 24 million training patches
  - 22 epochs
  - Adam optimizer; Learning rate of 10^{-4}; mini-batch size of 512
- CNNs for 24 test points
  - 6 Encoding time points
  - 4 QPs: 22, 27, 32, and 37
  - An individual μ for each CNN, such that
    - for all QPs of an encoding time point:
    - approximately the same encoding time reduction (in percent)
- Coding conditions
  - JVET's test conditions for All-Intra Coding
  - CNN run-times included
  - Basis and anchor: VTM-7.0
  - No default optimizations disabled

Results



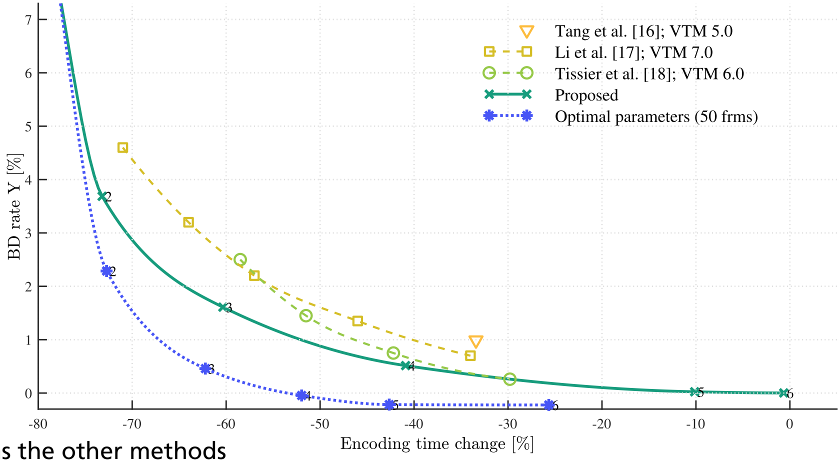
- Results
  - 10% encoding time reduction, neglectable BD rate increase
  - 50% encoding time reductions, 0.9% BD rate increase
  - 73% encoding time reductions, 3.7% BD rate increase

Results



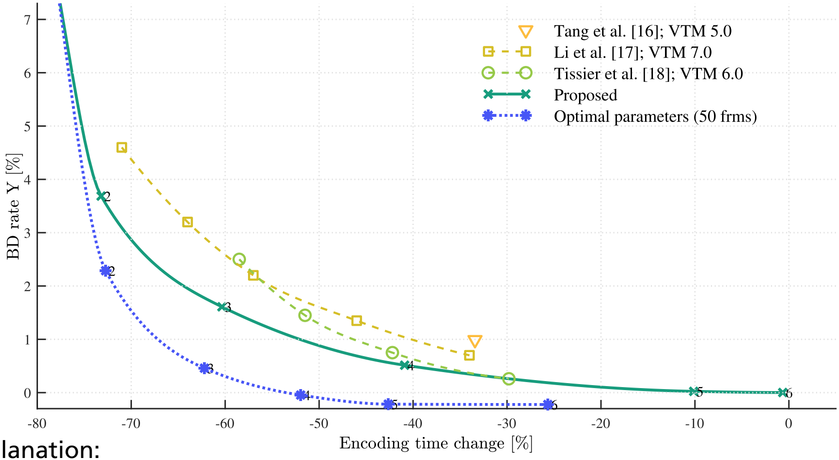
- Optimal parameters
  - Hypothetical result
  - Gap: CNN estimation not perfect
  - Potential for further improvements, e.g. advanced CNN layouts

Results



- Outperforms the other methods
  - Other: great number of parameters with much finer granularity.
  - Our: only estimates two parameters for a 32×32 block

Results



- Possible explanation:
  - Advanced loss function
  - Can explore the actual RDT cost for the full parameter space offline
  - Trades off estimation flexibility against estimation accuracy

Summary and Conclusions

- CNN-based method for fast VVC intra-picture encoding

Summary and Conclusions

- CNN-based method for fast VVC intra-picture encoding
- How can the partitioning be restricted?
  - Two parameters: Defining the minimal width and height of MTT subdivisions
  - → Allow to skip testing blocks based on their size and orientation

Summary and Conclusions

- CNN-based method for fast VVC intra-picture encoding
- How can the partitioning be restricted?
  - Two parameters: Defining the minimal width and height of MTT subdivisions
  - → Allow to skip testing blocks based on their size and orientation
- How can the parameters be selected?
  - Lagrangian multiplier method considering the RDT cost
  - → Theoretically optimal solution

Summary and Conclusions

- CNN-based method for fast VVC intra-picture encoding
- How can the partitioning be restricted?
  - Two parameters: Defining the minimal width and height of MTT subdivisions
  - → Allow to skip testing blocks based on their size and orientation
- How can the parameters be selected?
  - Lagrangian multiplier method considering the RDT cost
  - → Theoretically optimal solution
- How to approximate the optimal solution practically?
  - CNN with original samples as input
  - Training data generation using the Lagrangian multiplier method
  - Interpolation of the loss function from the recorded data
  - → CNN estimates the optimal parameters

Summary and Conclusions

- CNN-based method for fast VVC intra-picture encoding
- How can the partitioning be restricted?
  - Two parameters: Defining the minimal width and height of MTT subdivisions
  - → Allow to skip testing blocks based on their size and orientation
- How can the parameters be selected?
  - Lagrangian multiplier method considering the RDT cost
  - → Theoretically optimal solution
- How to approximate the optimal solution practically?
  - CNN with original samples as input
  - Training data generation using the Lagrangian multiplier method
  - Interpolation of the loss function from the recorded data
  - → CNN estimates the optimal parameters
- Conclusions
  - We outperform other methods although using only two parameters for large blocks
  - → Our loss function models the RDT cost accurately
  - → Might be better to optimize fewer parameters, but with an exact loss function

Thank you for your attention!

- Any questions?

→ gerhard.tech@hhi.fraunhofer.de

Gerhard Tech