Data driven methods in video compression. Summer school on video compression, July 2021, Berlin.

Transform coding and data driven transforms

Review of classical transform coding

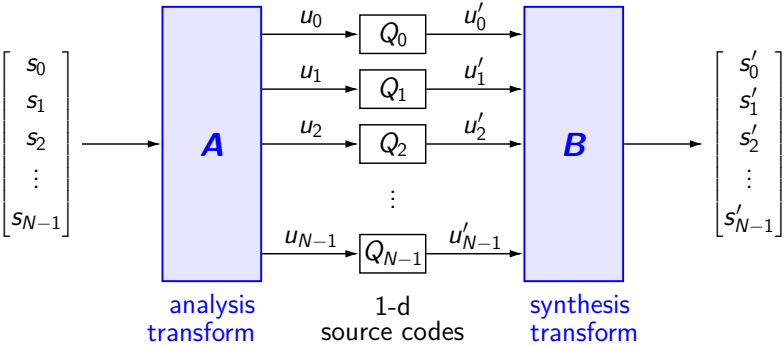
Definition (Transform coding)

Let S be an N-dimensional source. Invertible $N \times N$-matrices A, B and 1-d source codes $Q_0, \ldots, Q_{N-1}$ define the transform coding of $s \in \mathbb{R}^N$, realization of S, as the following ordered steps:

1. Compute $u = As$. The components $u_0, \ldots, u_{N-1}$ of u are called transform coefficients.
2. Code each u_i with Q_i to obtain the reconstructed transform coefficients u'_i that form the reconstructed transformed vector u'.
3. Compute the reconstructed vector s' as $s' = Bu'$.

- Transform coding is the **cornerstone of all modern image and video codecs**.
- In video coding: Transform coding is applied for **prediction residuals** (motion-compensated prediction or intra prediction).
- In most cases, one has $B = A^{-1}$ and even assumes **orthogonal** transforms, i.e. $A = B^{-1} = B^t$. If orthogonal: Quantization error in transform domain equals quantization error in sample domain; important property for efficient quantization algorithms at encoder.

Structure of Transform Coding Systems



Effect of transform coding:

- Remove/reduce dependencies before scalar quantization
- Simple alternative to vector quantization
- Simple and most relevant case: Linear transforms

Transform coding: Optimal transforms

Basic questions: Given source S . What is the optimal transform?

Karhuenen Loewe transform

- Assume S of zero mean. Autocorrelation matrix $R_S := \mathbb{E}(SS^t)$ is diagonalizable.
- Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ be eigenvalues of R_S.
- A matrix A_{KLT} with $A_{KLT}^t R_S A = diag(\lambda_1, \ldots, \lambda_N)$ is called a KLT of S.
- The i-th row of A_{KLT} is of norm one and is an eigenvector of R_S for eigenvalue λ_i. If eigenvalues are distinct, A_{KLT} is unique up to multiplication by ± 1 of its rows.

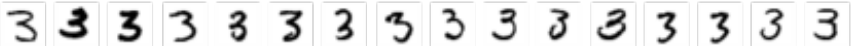
Optimality of KLT

- KLT **always** has **optimal energy compaction property**: For each $k \leq N$, subspace spanned by the first k rows of A_{KLT} is the best k-dimensional approximation for S.
- **Optimality for transform coding** [Goyal, Zhuang, Vetterli, 2000]: Assume that there exists a single function f such that for any transform T, the i-th component of $(TS)_i$ of TS can be code with distortion-rate function $D_i(R_i) = \sigma_i^2 f(R_i)$, where σ_i variance of $(TS)_i$.
  Then KLT is the optimal orthogonal transform for transform coding of S. In particular: KLT is optimal for Gaussian sources.

Example of a KLT

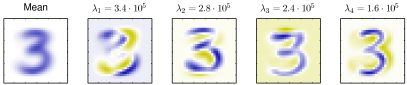
Taken from *Bishop, Pattern Recognition and Machine Learning*.

- Let S be large collection of handwritten digits of the number 3, taken from MNIST dataset.
- Each data-sample of an handwritten digit is a grey-scale image with 28×28 pixels.
- Thus: Data samples correspond to vectors $x_d \in \mathbb{R}^N$ for $N = 748$.



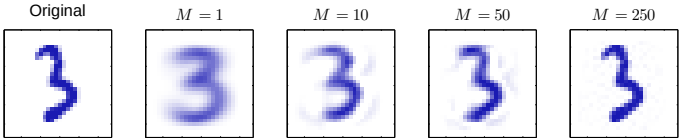
Example for handwritten digits from MNIST.

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ and $v_1, \ldots, v_N$ denote the eigenvalues and eigenvectors of the empirical covariance matrix of S.



The mean (blue) and the first four eigenvectors $v_1, \ldots, v_4$.

Visualization of energy compaction of KLT



Original handwritten digit (left), and its representation by using only a subset $v_1, \ldots, v_M$ of the principal components. If $x \in \mathbb{R}^{748}$, the picture for value M represents

$$\mu + \sum_{i=1}^{M} < x, v_i > v_i,$$

where $\mu \in \mathbb{R}^N$ is the mean of the data.

Design of transforms for image- and video-coding

Problem: The source S a video codec has to deal with is much more **complicated and versatile/non-stationary** than above example of handwritten figure 3.

- Classical approach to transform design: 'Signal independent approximation of KLT'.
- Assume a simple AR-1 process with correlation ρ. Thus, $S = (S_1, \ldots, S_N)$ satisfies

$$\mathbb{E}(S_i \cdot S_j) = \rho^{|i-j|}.$$

- For $\rho \to 1$, eigenvectors of R_S converge to basis functions of the **Discrete Cosine Transform DCT-II** [Ray-Driver, 1970].

➡ 2-dimensional DCT-II is the basic transform for transform coding in JPEG, H.264|AVC, H.265|HEVC, H.266|VVC.

2-dimesional DCT-II:

- **Separable** application of DCT-II along horizontal and vertical direction.
- If applied on $M \times N$ blocks, requires (at most) $M + N$-multiplications per sample.
- Huge complexity reduction compared to arbitrary transform T, which requires $M \cdot N$ multiplications per sample.

Data driven transform design

How to design more general transforms for the source S of 'all natural images', 'all natural video sequences', 'all natural prediction residuals'?

- Source S is unknown, very complicated.

➡ **Data-driven approach**: Try to approximate source S by the statistics of a large set $\mathcal{T}$ of training data.

- Due to non-stationarity of source: **Clustering** of the training data $\mathcal{T}$ into multiple subsets

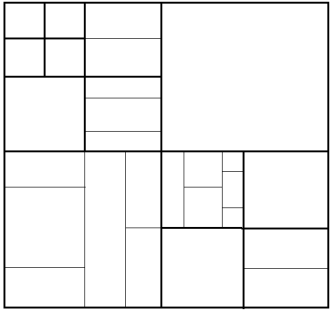
$$\mathcal{T} = \bigsqcup_{i \in \mathbb{I}} \mathcal{T}_i$$

  such that on each $\mathcal{T}_i$, statistics is 'more stationary'.
- Finding an appropriate clustering is not obvious, part of the problem.
- For compression: For a given realization s of S, need to transmit its class index i to the decoder.

➡ Try to design an RD-optimal clustering.

Data driven transforms for intra-prediction residuals in video coding

- Modern video codecs like H.264|AVC, H.265|HEV, H.266|VVC are based on a hybrid, block-based architecture.
- Partition a frame into blocks. On each block apply inter- or intra-prediction, transform coding of prediction residual and entropy coding of prediction information.

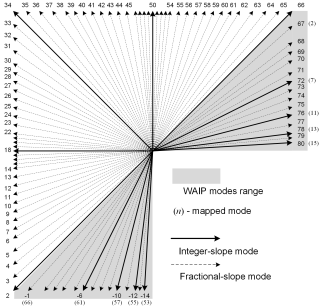


An example of a splitting of a 128×128 coding tree unit by the quad-tree plus multi-type-tree method used in VVC-partitioning.

Data driven transforms for intra-prediction residuals in video coding

- Typically: Prediction residual on intra-predicted blocks much larger than on inter-predicted blocks.
- Often, intra prediction residuals tend to follow the same directionality as the underlying intra prediction mode.

➡ Use intra prediction mode to cluster into multiple transform sets.



The intra-direction modes of VVC, including wide-angle modes. For a given block, 65 directional modes as well as the Planar, DC-mode and Matrix-Based Intra Prediction modes are applicable.

Data driven transforms for intra residuals: Complexity

Assume that **without restrictions** one attempts to determine transforms for each class of intra-prediction residuals.

Complexity: Number of multiplications.

- For block-shape $W \times H$, need $W \cdot H$ many multiplications per sample.
- ➡ For 64×64 blocks, need **4096 multiplications per sample**.
- *If use only DCT-II*: For 64×64 blocks, need (at most) **64 multiplications per sample**.

Complexity: Number of transforms and number of parameters.

- ➡ If M transforms requried for each block shape $W \times H$, need to store $M \cdot (W \cdot H)^2$ many parameters.
- ➡ If all $W \times H$ blocks with $W, H \in \{4, 8, 16, 32, 64\}$ are supported for intra prediction and $M = 8$, then restricting to case $W \leq H$, one would need to store **120 transforms** with about **190 Million parameters** in total.
- *If use only DCT-II*: Need to store **5 transforms** with (at most) **5456 parameters** in total.

➡**Need to invoke structural/complexity constraints when designing data driven transforms.**

Complexity reduction: Row-column transforms

Row column transforms, Ye et al. 2008, Sezer et al. 2011, Egilmez et al. 2016 ...

- Given an $M \times N$-block, with M rows and N columns.
- Given M row-transforms $R_1, \dots, R_M \in \mathbb{R}^{N \times N}$ and N column transforms $C_1, \dots, C_N \in \mathbb{R}^{M \times M}$.
- ➡ Transform signals $s \in \mathbb{R}^{M \times N}$ on the $M \times N$ block by first applying each R_i to the i-th row and then each C_j to the j-th column.
- Generalization of a separable transform. Huge complexity reduction. But: Obtaining data-driven row-column transforms not trivial.

Determination of row column transforms: Algorithm of Egilmez et al.:

- Given an arbitrary non-separable/non-structured transform $H \in \mathbb{R}^{(M \cdot N) \times (M \cdot N)}$ on the $M \times N$-block.
- Try to approximate permutation of H (i.e. re-ordering of the rows of H):

$$\underset{G,P}{\text{minimize}} \, \|PH - G\|_F^2 \,, \quad P \text{ a permutation matrix, } G \text{ a row-column transform}$$

- For fixed P, minimum can be computed explicitly. For fixed G, problem is a combinatorial problem solvable in polynomial time.
- Proceed alternatingly.

Complexity reduction: Layered Givens rotations

- Approximation of an arbitrary orthogonal transform by a **short** sequence of parallel 2d-rotations in permuted coordinates.
- Assume that 2d-dimensional signal space is given.
- **Givens layer**: $2d \times 2d$-matrix

$$G = P \cdot T \cdot P^t,$$

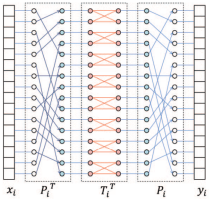
  where
  - P a permutation
  - T direct sum of planar rotations matrices T_k, $1 \leq k \leq d$. Each T_k rotation in the 2-dimensional plane spanned by $2k-1$-st and $2k$-th standard basis vector.
- Algorithm of Li et al, 2017: Given orthogonal matrix $H \in \mathbb{R}^{(2d)\times(2d)}$, for each $N << 2d(2d-1)$ find Givens layers $G_1, \ldots, G_N$ so that

$$\|H - G_1 \cdot \cdots \cdot G_N\|_F^2$$

  is minimized.

Layered Givens rotations

- Decomposition into Givens layers is a generalization of the butterfly decomposition for fast Fourier transform, DCT-II algorithms.
- Each Givens layer requires 1 multiplication per sample and the storage of d rotation-angles and one permutation.
- Secondary transforms implemented by Givens layers were part of the **Joint Exploration Model (JEM)**, see Chen et al., 2020.



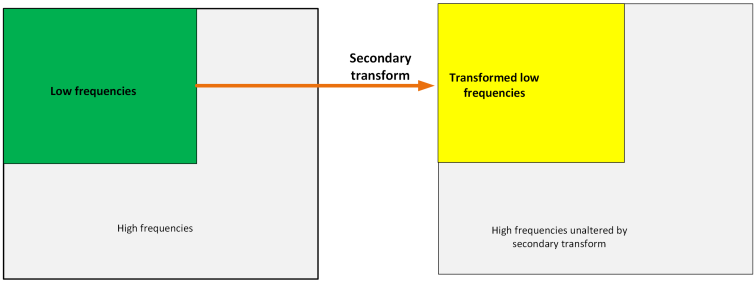
Li et al.: Example of an inverse Givens layer G_i^t. The input vector x_i is permuted by P_i^t. Then planar rotations are applied in parallel by the matrix T_i^t. The result is permuted back by P_i.

Complexity reduction: Secondary transforms with restricted domain.

Invoke data-driven transforms as **secondary transforms**:

- Transforms operate on a k-dimensional subspace of the full d-dimensional residual space, $k << d$.
- This subspace is spanned by the first k basis functions of the DCT-II.
- Energy compaction property of DCT-II: This subspace should already contain most energy of the residuals.

➡ Transforms operate only on low frequencies.



The Low frequency non-separable transform (LFNST) from H.266|VVC

- For each intra-prediction mode, VVC supports 2 non-separable transforms.
- These transforms operate as secondary transforms.
- Index is signaled in bit-stream that determines whether LFNST used or not.
- For a given block-size, 4 sets of transforms, each consisting of 2 non-separable transforms.
- Transform set determined by intra prediction mode:

intra-mode m	LFNST-set
$m \leq 1$	0
$2 \leq m \leq 12$ or $56 \leq m \leq 80$	1
$13 \leq m \leq 23$ or $45 \leq m \leq 55$	2
$24 \leq m \leq 54$	3

.

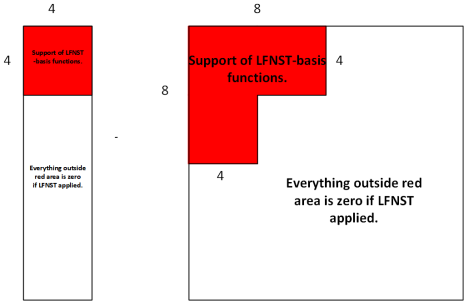
  Index that determines the transform within the transform set is signaled in bit stream.
- Sets of transforms are shared between different block-shapes:
  - $LFNST4$: 4 transforms sets of 16×16 matrices, used as inverse transforms for $4 \times H$ and $W \times 4$ blocks. Only first 8 columns used for 4×4 blocks.
  - $LFNST8$: 4 transforms sets of 48×16 matrices, used as inverse transforms for all other blocks. Only first 8 columns used for 8×8 blocks.

The Zeroing process in the LFNST

LFNST-transforms of VVC are coupled with a **zeroing** in the DCT-II-domain:

- Whenever an LFNST-transform is applied on an $W \times H$-block, a prediction residual that belongs to a d-dimensional subspace of $U \subset \mathbb{R}^{W \times H}$ of the full residual space $\mathbb{R}^{W \times H}$ is transmitted.
- One has $d = 8$ for 4×4 and 8×8 blocks and $d = 16$ in all other cases.
- U is spanned by d basis functions which depend on the LFNST, but which always belong to the p-dimensional subspace V of the full residual space $\mathbb{R}^{W \times H}$ spanned by the p lowest DCT-II basis functions.
- For $4 \times H$ and $W \times 4$ blocks, one has $p = 16$ and V is the space generated by the first 4×4 DCT-II basis functions.
- For all other blocks, one has $p = 48$ and V is the space generated by the the first three 4×4 quadrants in the DCT-II domain.

The Zeroing process of LFNST: Illustration



The p-dimensional subspace (red) V in the DCT-II domain that always contains the d-dimensional subspace U spanned by the LFNST-basis functions.

Left: Case of $4 \times H$ and $W \times 4$ blocks: Here, always $p = 16$ and $d = 8$ for 4×4-blocks and $d = 16$, else.

Right: Case of all other blocks. Here, always $p = 48$ and $d = 8$ for 8×8 blocks and $d = 16$, else.

LFNST and MIP in VVC

Complexity of LFNST:

- Due to dimension reduction, worst case number of multiplications per sample from LFNST is **8 multiplications per sample**.
- Total number of matrix entries of all LFNST-matrices is 8192.

➡ **Huge complexity reduction compared to full non-separable transforms**.

- Assumption that residual is representable in a fixed 16-dimensional space may not always hold, particularly for small quantization step-size.
- LFNST is *switchable*, can be switched off if assumption does not hold

Data driven intra prediction modes:

- Similar objective than for data-driven transforms: Design multiple intra prediction modes for application in video coding.
- Complexity problems similar as for transforms: Memory and number of multiplications. Intra-loop very tight, even tighter than transform-loop, sequential processing of blocks.
- **Matrix based intra prediction (MIP):** Low complexity variant of data-driven intra prediciton.
- MIP requires 4 multiplications per sample, same as other intra modes. Memory: 4500 parameters.

Non-linear transform coding. End-to-end image compression

Motivation

- Data driven transforms presented so far only determine the transform itself.
- **But**: A full transform coder also consists of entropy coding of transform coefficients.
- Assuming ideal arithmetic coding: Problem of entropy coding is equivalent to model the (conditional) **probability distribution of the transform coefficients**.
- An 'ideal' transform would be a transform that minimizes the average expected codeword length of the quantized transform coefficients at a given distortion.
- Energy compaction or sparsity are only coarse approximations for the rate in a transform coder. KLT essentially optimal only for Gaussian sources.
- Training **non-linear** transforms may lead to more flexible transforms, possibly better compression efficiency.

Basic idea [Balle, Minnen, Toderici and others, starting 2016]:

- Data-driven approach to jointly optimze a non-linear transform with the parameters of an entropy coding scheme for the transform coefficients.

➡ End to end compression.

Outline of the transform architecture

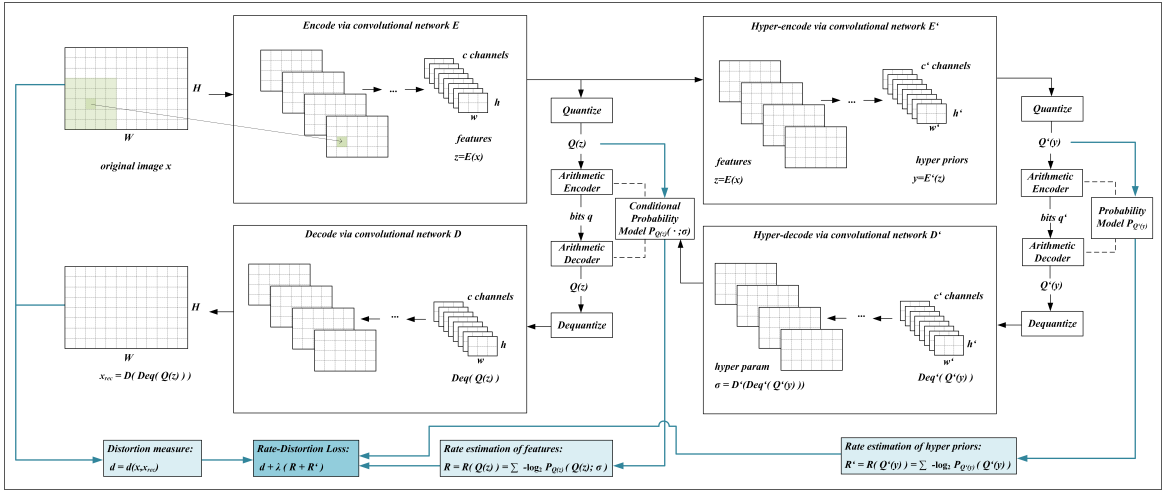
Full images as domain for the data driven transforms:

- Instead of using a block-based approach, consider transforms for full images.
- ➡ By using **convolutions** as basic processing blocks, designed transforms do not depend on the specific image shape.
- Can avoid block-shape-dependency of trained models.
- But: Convolutional layers not directly applicable to small blocks. Effect of boundary padding too large, particularly for deeper networks and downsampling steps.

Encoder and decoder layers:

- Encoder and decoder functions are defined as compositions of multiple encoder and decoder layers.
- Each encoder layer is a convolutional layer, followed by an activation and a downsampling.
- Each decoder layer is an upsampling by zeros, followed by a convolutional layer and an activation.
- ➡ Some similarity to a *filter bank*. But: **Non-linear** activation present.

Flow chart


Encode via convolutional network E
c channels
H
W
h
w
...
original image x
features
z=E(x)
Quantize
Q(z)
Arithmetic Encoder
bits q
Arithmetic Decoder
Q(z)
Dequantize
Conditional Probability Model $P_{Q(z)}(\cdot;\sigma)$
Hyper-encode via convolutional network E'
c' channels
h'
w'
features
z=E(x)
hyper priors
y=E'(z)
Quantize
Q'(y)
Arithmetic Encoder
bits q'
Arithmetic Decoder
Q'(y)
Dequantize
Probability Model $P_{Q'(y)}$
Decode via convolutional network D
c channels
H
W
h
w
x_{rec} = D(Deq(Q(z)))
Deq(Q(z))
Hyper-decode via convolutional network D'
c' channels
h'
w'
hyper param
σ = D'(Deq'(Q'(y)))
Deq'(Q'(y))
Distortion measure:
$d = d(x, x_{rec})$
Rate-Distortion Loss:
$d + \lambda (R + R')$
Rate estimation of features:
$R = R(Q(z)) = \sum -\log_2 P_{Q(z)} (Q(z); \sigma)$
Rate estimation of hyper priors:
$R' = R(Q'(y)) = \sum -\log_2 P_{Q'(y)} (Q'(y))$

Encoder and Decoder

- Encoder applies **Convolutional neural network** (CNN) E to original image to obtain features z.
- **Features:** Union of downsampled image planes z_k,

$$z = \bigsqcup_{k=1}^{c} z_k .$$

  - Each z_k should represent a different class of typical characteristics present in natural images.
  - Typical image z should have sparse representation in terms of the z_k.
- Quantized version of feature are transmitted in the bit stream.
- Decoder applies **Deconvolutional neural network** D to reconstructed features to obtain reconstructed image.
- Parameters of encoder and decoder are trained to minimize the rate-distortion loss at a given operational point parametrized by λ.

Structure of encoder and decoder layers

- A **convolutional layer** computes the output $y \in \mathbb{R}^{M \times N \times C_{i+1}}$ from the input $x \in \mathbb{R}^{M \times N \times C_i}$ as

$$y^{i+1}(m,n,k) = \sum_{l=1}^{C_i} (A^i_{k,l} * x^i)(m,n) + b^i_k, \quad 1 \leq m \leq M,\ 1 \leq n \leq N,\ 1 \leq k \leq C_{i+1}.$$

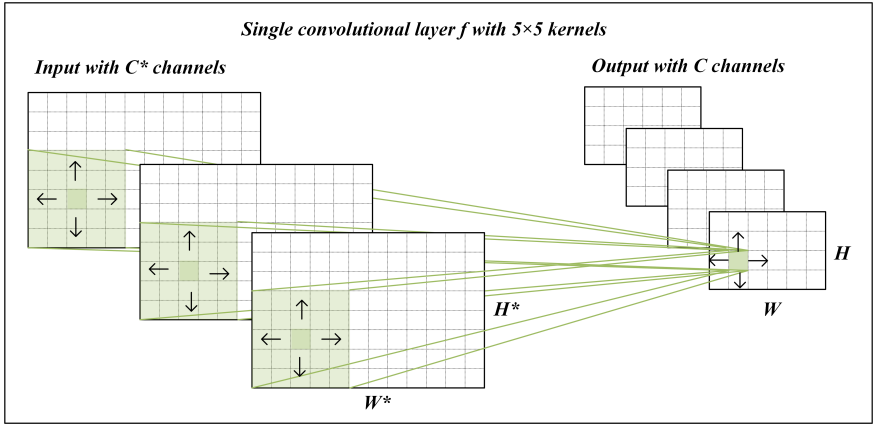
  where $*$ denotes ordinary convolution.

- The **activation** computes the output $v^{i+1} \in \mathbb{R}^{M \times N \times C_{i+1}}$ from the input $u^{i+1} \in \mathbb{R}^{M \times N \times C_{i+1}}$ as

$$v^{i+1}(m,n,k) = \frac{u^{i+1}(m,n,k)}{\left(\beta^{i+1}_k + \sum_{l=1}^{C_{i+1}} \gamma^{i+1}_{k,l} \cdot u(m,n,k)^2\right)^{1/2}}, \quad 1 \leq m \leq M,\ 1 \leq n \leq N,\ 1 \leq k \leq C_{i+1}.$$

- The kernels $A^i_{k,l} \in \mathbb{R}^{p \times p \times C_{i+1} \times C_i}$ and the $b^i_k, \beta^i_k, \gamma^i_{k,l} \in \mathbb{R}$ are trained on a large set of data.

Example: 5×5 convolution with downsampling



- For each target channel, there are C^* many convolution kernels and one bias
- The convolution requires to pad the input at the boundary by either reflecting or zero-padding

Entropy coding of quantized features

Probability estimation:

- Assume **continuous, parametrized probability distribution** $P(z|\sigma)$ of the features z.
- For each image, transmit separate parameters σ, called **hyper parameters**, in bit stream.
- Use probability distribution $P(z|\sigma)$ for entropy coding of the quantized features.
- **Arithmetic coding :** Can code the features with bit rate approximately equal to entropy $H(z|\sigma)$.

Transmission of hyper parameters:

- Use second convolutional neural network to transmit hyper parameters.
- Encoder: Transform hyper parameters to **hyper priors** via a CNN.
- Hyper priors are quantized and transmitted in bit stream.
- Fixed probability model is used for entropy-coding of hyper priors.
- Decoder: Generate hyper parameters from reconstructed hyper priors by deconvolutional neural network.

Example: Independent Gaussian-distributed features

- Assume that that feature for different feature positions and feature channels are independent.
- For a given position (i, j) in the k-th channel, assume that 1-d feature variable $z := z_{i,j,k}$ is **Gaussian distributed** wit mean $\mu := \mu_{i,j,k}$ and variance $\sigma := \sigma_{i,j,k}$, i.e.

$$z \sim \mathcal{N}(\sigma_{i,j,k}, \mu_{i,j,k}).$$

- Mean μ and variance σ are transmitted in bit stream.
- Apply uniform scalar quantization to z with reconstruction points $z_l := \mu + l \cdot \Delta$, with Δ fixed stepsize.
- Code quantization indices l using the discrete pdf

$$p(l) := p(z_l) := \frac{1}{\sqrt{2\pi\sigma^2}} \int_{z_l - \Delta/2}^{z_l + \Delta/2} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) dx.$$

Extension: **Autoregressive models**. Assume that probability of feature $z_{i,j,k}$ also depends on previously coded features $z_{i-m,j-n,k}$; mean and variance depend on hyperpriors and on $z_{i-1,j,k}, z_{i,j-1,k}, \ldots$.

Quantization and encoder control for end-to-end based compression

- **Quantization** is much more complicated than in classical transform coding.
- Reasons: Nonlinearity and non-orthogonality. Difficult to understand how quantization error in feature-domain relates to error in sample domain.

Observations and findings of Schäfer et al., 2021:

- If model is trained for a single Lagrangian parameter on RD-curve, unquantized value $E(z)$ may not be a distortion minimum. This makes quantization problem very hard.
- Training a single auto-decoder for all Lagrangian parameters, in particular for high reconstruction qualities, can mitigate this phenomenon.
- The *bit-allocation between feature planes* of the initial encoder network $E(z)$ seems to be optimal, i.e. at equal slope.
- The distortion in the feature- and sample domain can be approximately related by a 4-th order polynomial. Coefficients depend on feature-plane.
- Using above polynomial, a fast **RDOQ - algorithm** can be designed **for end-to-end based image compression** that yields significant rate savings.

Data-driven in-loop filters

In-loop filtering: Introduction and classical approaches

Basic task of in-loop filters:

- Try to correct or mitigate **compression artefacts** by applying a filter.
- In-loop filters are placed after the reconstruction.
- In-loop: Frame with loop-filter applied is used as a reference frame for motion compensated prediction of future frames.

Example of in-loop filters:

- The initial output frames of hybrid, block-based video codecs may reveal artificial discontinuities at block-boundaries of the selected partitioning of a frame, called **blocking artefacts**.

➡ **Deblocking filter:** Correct these artefacts by applying signal adaptive filters across block boundaries. Huge benefits, in particular subjectively.

HEVC output frame before (left) and after (right) deblocking. Norkin et al., 2012.

In-loop filtering: Classical approaches

- **Gibbs phenomenon**: Fourier series of piecewise smooth function converges everywhere to the average of left-and right-side, *but:* Overshooting and undershooting by about 9 per-cent from side of maximum and minimum at discontinuity ➡**ringing artefacts** for images and videos.

➡ **Sample adaptive offset (SAO)** filters in HEVC: Attenuate ringing artefacts by local offset addition, offset signaled in bit-stream.

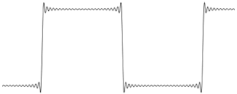


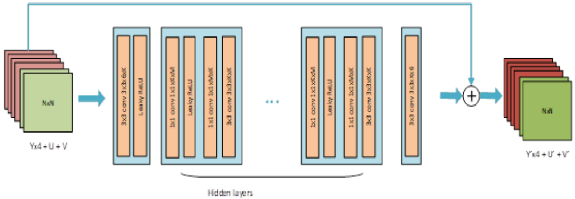
Illustration of the Gibbs phenomenon.

- **Local noise or over-smoothing of texture**: Can be mitigated by the **Adaptive Loop Filter ALF** in VVC. Basic functioning:
  - 2x2 subblocks of initial output frame are classified by gradient intensity and direcionality.
  - For each class, a class-dependent linear filter and a non-linear clipping operation may be applied. Parameters of filter and clipping may be signaled in bit-stream.
  - Cross component filtering from luma to chroma is also supported (CCALF).

In-loop filtering: Data-driven approaches

Basic question: Are there other compression artefacts that one can mitigate efficiently?

- Generalization of linear in-loop filters: Apply **convolutional neural networks** to reconstructed frames that decrease the deviation of reconstructed frames from original frames.
- Deviation typically measured as mean squared error or as SSIM, measure that correlates more to subjective judgment.
- **Data-driven approach**: Parameters of the convolutional neural networks trained on large datasets.



Example of a CNN-based in-loop filter, JVET-U0104, Chen et al., 2020. The filter is applied to enhance 128×128 blocks. For 4:2:0-format, the Y-component is interleaved to 4 planes of shape 64×64. The number of hidden layers is equal to 12.

Current activity on CNN-based in-loop filters in JVET

Large post-VVC activity in JVET. Current status:

- It seems possible to obtain **significant objective coding gains over VVC** by applying CNN-based in-loop filters.
- Some approaches manage to completely replace all other in-loop filters (de-blocking, SAO, ALF).
- Since CNN-based in-loop filters are placed after the decoding of a frame, less strict complexity requirements than e.g. for entropy decoding or intra-prediction. Parallel processing supported by a GPU might be possible in the future.

Problems:

- Huge computational complexity. Orders of magnitudes higher than classical in-loop filters. Even for GPU-support, the complexity of current approaches too high for the near future.
- It sometimes seems to be difficult to explain *which compression artefacts* are corrected *in which way* by the CNN-based in-loop filters.

➡ **Better understanding of the technology seems to be required**. This could also help to reduce the complexity.

Further approaches to data-driven methods.

Other data driven approaches to video compression

- Learn parameters of neural network that estimates the probabilities of transform coefficient levels in HEVC intra-prediction residuals. Ma et al., 2020.
- Motion-vector coding and end-to-end approach to inter-prediction. Use auto-encoder-architecture similar as in image coding to code motion-vectors and residual transform coefficients. Rippel et al., 2019.
- In above approach, motion vectors may have an extra *scale-space-dimension* that parametrizes a blurring of the prediction signal. Agustsson et al., 2020,
- New approaches to data-driven intra-prediction and transform coding. Brand et al. 2019, Dumas et al., 2020, Meyer, Schneider et al., 2019-2021.
- Also many other approaches.

Informal conclusions

- Data-driven methods seem to be *natural* for video-compression since distributions of underlying sources for video coding can essentially only be approximated by data-driven methods.
- Methods recently developed in the field of artificial intelligence as well as increasing computational capabilities are very helpful for designing new compression tools.
- Reducing the complexity of the designed tools and understanding them better seem to be interesting and difficult problems.
- The problem of complexity already occurs in the linear case. Non-linear structures might be useful but may also be more difficult to understand.

➜A lot of interesting problems and opportunities for future research on video compression, both in the hybrid, block-based, and in the end-to-end framework.