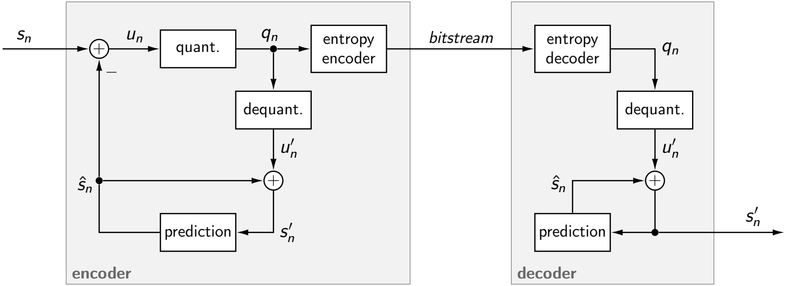
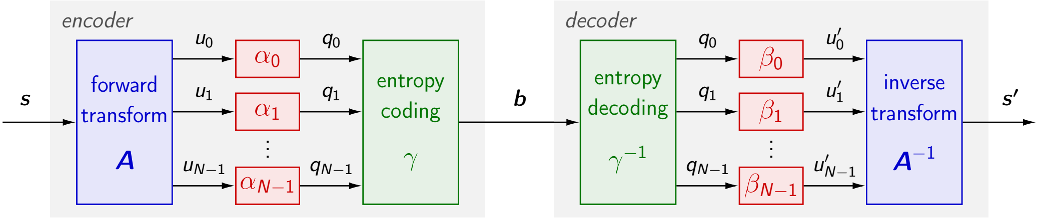
Predictive Quantization



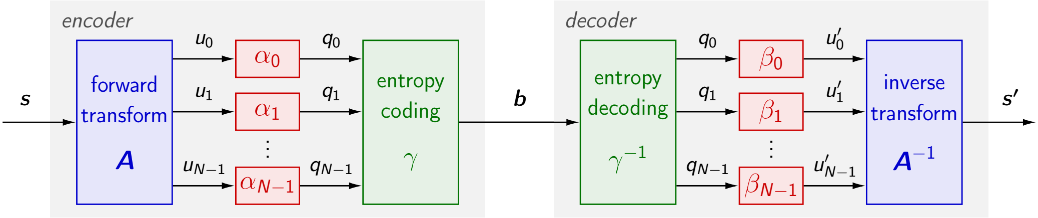
Last Lectures: Transform Coding with Orthogonal Block Transform



- Forward transform: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s}$
- Scalar quantization: $q_k = \alpha_k(u_k)$
- Entropy coding: $\boldsymbol{b} = \gamma(\,\{q_k\}\,)$

- Entropy decoding: $\{q_k\} = \gamma^{-1}(\boldsymbol{b})$
- Dequantization: $u'_k = \beta_k(q_k)$
- Inverse transform: $\boldsymbol{s}' = \boldsymbol{A}^{-1} \cdot \boldsymbol{u}'$

Last Lectures: Transform Coding with Orthogonal Block Transform



- Forward transform: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s}$
- Scalar quantization: $q_k = \alpha_k(u_k)$
- Entropy coding: $\boldsymbol{b} = \gamma(\{q_k\})$

- Entropy decoding: $\{q_k\} = \gamma^{-1}(\boldsymbol{b})$
- Dequantization: $u'_k = \beta_k(q_k)$
- Inverse transform: $\boldsymbol{s'} = \boldsymbol{A}^{-1} \cdot \boldsymbol{u'}$

Orthogonal Block Transforms

- Inverse matrix is equal to transposed matrix: $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$
- Geometric Interpretation: Rotation (and possible reflection) in N-d signal space

➞ Can be interpreted as lattice vector quantizer with orthogonal lattice

- Preservation of MSE distortion: Independent quantization of individual transform coefficients

Last Lectures: High-Rate Approximations

High-Rate Approximation of Operational Distortion-Rate Function

- Optimal bit allocation at high rates: $D_k(R_k) = \text{const}$ (for URQs: $\Delta_k = \text{const}$)

➡ High-rate distortion-rate function for transform coding (with optimal bit allocation)

$$D_{TC}(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \qquad \text{with} \qquad \tilde{\varepsilon}^2 = \Big(\prod_k \varepsilon_k^2\Big)^{\frac{1}{N}}, \quad \tilde{\sigma}^2 = \Big(\prod_k \sigma_k^2\Big)^{\frac{1}{N}}$$

Last Lectures: High-Rate Approximations

High-Rate Approximation of Operational Distortion-Rate Function

- Optimal bit allocation at high rates: $D_k(R_k) = \text{const}$ (for URQs: $\Delta_k = \text{const}$)

➡ High-rate distortion-rate function for transform coding (with optimal bit allocation)

$$D_{TC}(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \qquad \text{with} \qquad \tilde{\varepsilon}^2 = \left(\prod_k \varepsilon_k^2\right)^{\frac{1}{N}}, \quad \tilde{\sigma}^2 = \left(\prod_k \sigma_k^2\right)^{\frac{1}{N}}$$

Goal of Transform Selection

- Minimize distortion $D_{TC}(R)$ for given rate R ➡ Maximize transform coding gain G_T

$$G_T = \frac{D_{SQ}(R)}{D_{TC}(R)} = \frac{\varepsilon_S^2 \cdot \sigma_S^2}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2}$$

- Typically, ignore impact of pdf shapes ➡ **Maximize energy compaction G_{EC} of transform**

$$G_{EC} = \frac{\sigma_S^2}{\tilde{\sigma}^2} = \frac{\frac{1}{N}\sum_{k=0}^{N-1} \sigma_k^2}{\sqrt[N]{\prod_{k=0}^{N-1} \sigma_k^2}}$$

Last Lectures: Transform Selection

Karhunen Loève Transform (KLT)

- Basis vectors (rows of $\boldsymbol{A}$) are unit-norm eigenvectors of the N-th order auto-covariance matrix $\boldsymbol{C}_N$

➡ KLT yields uncorrelated transform coefficients ($\boldsymbol{C}_{UU}$ becomes a diagonal matrix)

➡ KLT achieves maximum possible energy compaction

$$G_{EC}^{(\text{KLT})} = \left| \frac{1}{\sigma_S^2} \, \boldsymbol{C}_N \right|^{-\frac{1}{N}} \qquad \left(\text{note: determinant } |\cdot| = \text{product of eigenvalues} \right)$$

Last Lectures: Transform Selection

Karhunen Loève Transform (KLT)

- Basis vectors (rows of $\boldsymbol{A}$) are unit-norm eigenvectors of the N-th order auto-covariance matrix $\boldsymbol{C}_N$

➡ KLT yields uncorrelated transform coefficients ($\boldsymbol{C}_{UU}$ becomes a diagonal matrix)

➡ KLT achieves maximum possible energy compaction

$$G_{EC}^{(\text{KLT})} = \left| \frac{1}{\sigma_S^2} \boldsymbol{C}_N \right|^{-\frac{1}{N}} \qquad \big(\text{note: determinant } |\cdot| = \text{product of eigenvalues}\big)$$

Discrete Cosine Transform of Type II (DCT-II)

- Represents discrete Fourier transform of mirrored signal (with half-sample symmetry of both sides)
- Signal independent: Basis vectors (rows of $\boldsymbol{A}$) are sampled cosines of different frequencies
- KLT approaches DCT-II for highly correlated sources ($\varrho \rightarrow 1$)
- For typical stationary signals: Only small loss in coding efficiency relative to KLT

➡ **Most widely used transform in image and video coding** (fast implementations possible)

Last Lectures: Coding Efficiency of Transform Coding

High-Rate Approximation for KLT and Gauss-Markov

- High-rate operational distortion-rate function and transform coding gain (for transform size N)

$$D_N(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R} \qquad \text{and} \qquad G_T^N = G_{EC}^N = (1 - \varrho^2)^{\frac{1-N}{N}}$$

➡ Transform gain increases with transform size N, but approaches a limit

Last Lectures: Coding Efficiency of Transform Coding

High-Rate Approximation for KLT and Gauss-Markov

- High-rate operational distortion-rate function and transform coding gain (for transform size N)

$$D_N(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1-\varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R} \qquad \text{and} \qquad G_T^N = G_{EC}^N = (1-\varrho^2)^{\frac{1-N}{N}}$$

➡ Transform gain increases with transform size N, but approaches a limit

Comparison to Rate-Distortion Bound

- Example: Gauss-Markov, KLT, and optimal scalar quantizers (ECSQ)

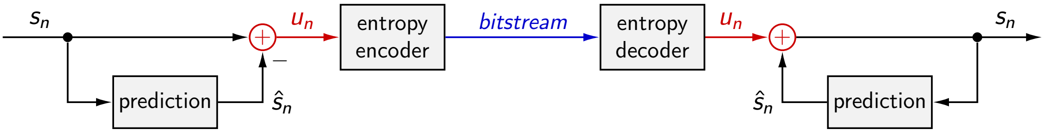
➡ Distortion increase relative to Shannon lower bound

$$\frac{D_N^{(\mathrm{KLT})}(R)}{D_{\mathrm{SLB}}(R)} = \frac{\frac{\pi e}{6} \cdot \sigma_S^2 \cdot (1-\varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}}{\sigma_S^2 \cdot (1-\varrho^2) \cdot 2^{-2R}} = \frac{\pi e}{6} \cdot \left(\frac{1}{1-\varrho^2} \right)^{\frac{1}{N}}$$

➡ Large transform sizes ($N \rightarrow \infty$): Performance gap reduces to space-filling gain (1.53 dB)

- Other sources: Transform coding cannot utilize all dependencies (non-linear dependencies)

Review: Lossless Coding using Prediction



Predictive Lossless Coding

- Predict current sample s_n using a function of preceding samples (referred to as observation set $\boldsymbol{b}_n$)

$$\hat{s}_n = f(s_{n-1}, s_{n-2}, \cdots) = f(\boldsymbol{b}_n)$$

- Entropy coding of prediction error samples

$$u_n = s_n - \hat{s}_n$$

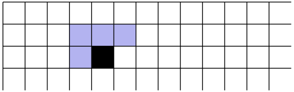
- Decoder reconstructs the original samples according to

$$s_n = \hat{s}_n + u_n$$

examples of observation sets

1D signals:

2D signals:



Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $\boldsymbol{b}_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n \qquad \text{with} \qquad \boldsymbol{a} = \big(a_1, \cdots, a_N\big)^{\mathrm{T}}$$

Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $\boldsymbol{b}_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n \qquad \text{with} \qquad \boldsymbol{a} = \big(a_1, \cdots, a_N\big)^{\mathrm{T}}$$

➡ **Prediction error variance σ_U^2** is given by

$$\sigma_U^2(\boldsymbol{a}) = \sigma_S^2 - 2\boldsymbol{a}^{\mathrm{T}}\boldsymbol{c} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}_B\boldsymbol{a} \qquad \text{with} \qquad \begin{aligned} \boldsymbol{C}_B &= \mathrm{E}\Big\{ \big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)^{\mathrm{T}} \Big\} \\ \boldsymbol{c} &= \mathrm{E}\Big\{ \big(S_n - \mathrm{E}\{S_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big) \Big\} \end{aligned}$$

Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $\boldsymbol{b}_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n \qquad \text{with} \qquad \boldsymbol{a} = \left(a_1, \cdots, a_N\right)^{\mathrm{T}}$$

➡ **Prediction error variance σ_U^2** is given by

$$\sigma_U^2(\boldsymbol{a}) = \sigma_S^2 - 2\boldsymbol{a}^{\mathrm{T}}\boldsymbol{c} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}_B\boldsymbol{a} \qquad \text{with} \qquad \begin{aligned} \boldsymbol{C}_B &= \mathrm{E}\Big\{ \big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)^{\mathrm{T}} \Big\} \\ \boldsymbol{c} &= \mathrm{E}\Big\{ \big(S_n - \mathrm{E}\{S_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big) \Big\} \end{aligned}$$

➡ Prediction error variance is minimized by solution of the **Yule-Walker equations**

$$\boldsymbol{C}_B \cdot \boldsymbol{a}_{\mathrm{opt}} = \boldsymbol{c} \qquad\qquad \text{(linear equation system)}$$

Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $\boldsymbol{b}_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n \qquad \text{with} \qquad \boldsymbol{a} = \big(a_1, \cdots, a_N\big)^{\mathrm{T}}$$

➡ **Prediction error variance** σ_U^2 is given by

$$\sigma_U^2(\boldsymbol{a}) = \sigma_S^2 - 2\boldsymbol{a}^{\mathrm{T}}\boldsymbol{c} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{C}_B\,\boldsymbol{a} \qquad \text{with} \qquad \begin{aligned} \boldsymbol{C}_B &= \mathrm{E}\Big\{ \big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)^{\mathrm{T}} \Big\} \\ \boldsymbol{c} &= \mathrm{E}\Big\{ \big(S_n - \mathrm{E}\{S_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big) \Big\} \end{aligned}$$

➡ Prediction error variance is minimized by solution of the **Yule-Walker equations**

$$\boldsymbol{C}_B \cdot \boldsymbol{a}_{\mathrm{opt}} = \boldsymbol{c} \qquad \text{(linear equation system)}$$

➡ Prediction error variance for optimal linear prediction

$$\sigma_U^2 \;=\; \sigma_S^2 - \boldsymbol{a}_{\mathrm{opt}}^{\mathrm{T}}\boldsymbol{c} \;=\; \sigma_S^2 - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{C}_B^{-1}\boldsymbol{c}$$

Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

$$\hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n$$

Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

$$\hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n$$

➡ Prediction error variance σ_U^2 is not impacted by constant offset a_0

Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

$$\hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n$$

➡ Prediction error variance σ_U^2 is not impacted by constant offset a_0

➡ Mean of prediction error μ_U can be forced to zero by choosing

$$a_0 = \mu_S \left(1 - \sum_{k=1}^{N} a_k \right)$$

Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

$$\hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{b}_n$$

➡ Prediction error variance σ_U^2 is not impacted by constant offset a_0

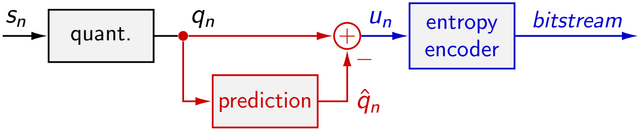
➡ Mean of prediction error μ_U can be forced to zero by choosing

$$a_0 = \mu_S \left(1 - \sum_{k=1}^{N} a_k \right)$$

➡ Affine prediction reduces mean squared prediction error (compared to linear prediction)

$$\varepsilon_U^2 = \mathrm{E}\left\{ \left(S_n - \hat{S}_n \right)^2 \right\} = \sigma_U^2 + \mu_U^2$$

Lossy Coding with Prediction: (1) Prediction of Quantization Indexes



Encoder

- Scalar quantization

$$q_n = \alpha(s_n)$$

- Prediction of quantization indexes

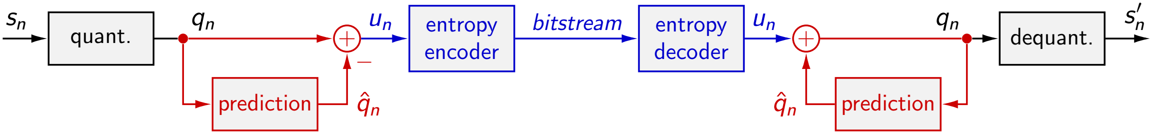
$$\hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots)$$

- Determining prediction residual

$$u_n = q_n - \hat{q}_n$$

- Entropy coding of prediction residual

Lossy Coding with Prediction: (1) Prediction of Quantization Indexes



Encoder

- Scalar quantization

$$q_n = \alpha(s_n)$$

- Prediction of quantization indexes

$$\hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots)$$

- Determining prediction residual

$$u_n = q_n - \hat{q}_n$$

- Entropy coding of prediction residual

Decoder

- Entropy decoding of prediction residual
- Prediction of quantization indexes

$$\hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots)$$

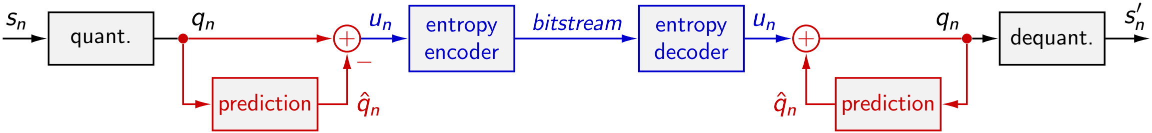
- Reconstruct quantization indexes

$$q_n = u_n + \hat{q}_n$$

- Dequantization (typically: scaling)

$$s'_n = \beta(q_n)$$

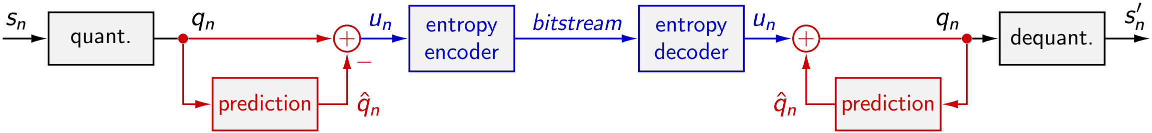
Lossy Coding with Prediction: (1) Prediction of Quantization Indexes



Prediction after Quantization

- Sources with memory: Quantization indexes have statistical dependencies

Lossy Coding with Prediction: (1) Prediction of Quantization Indexes



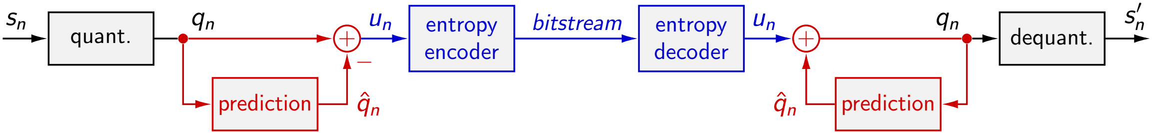
Prediction after Quantization

- Sources with memory: Quantization indexes have statistical dependencies

➡ Prediction can improve lossless coding of quantization indexes

  - More accurate: With prediction, entropy coding can be simplified
  - Require: Predicted value $\hat{q}_n$ must be integer (since u_n must be integer)

Lossy Coding with Prediction: (1) Prediction of Quantization Indexes



Prediction after Quantization

- Sources with memory: Quantization indexes have statistical dependencies

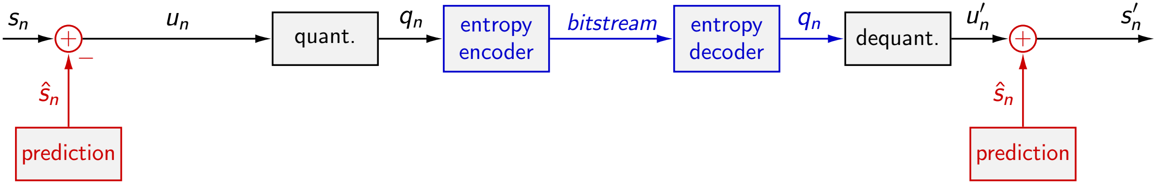
➡ Prediction can improve lossless coding of quantization indexes

  - More accurate: With prediction, entropy coding can be simplified
  - Require: Predicted value $\hat{q}_n$ must be integer (since u_n must be integer)

Extension: Prediction after Transform and Quantization

- Prediction of some transform coefficients (e.g., DC coefficient)
- Examples: JPEG, MPEG-2 Video, H.263, MPEG-4 Visual

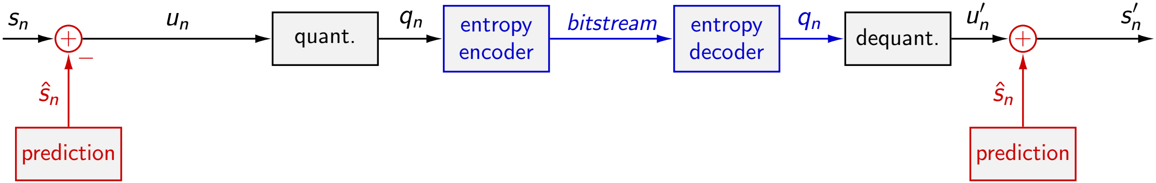
Lossy Coding with Prediction: (2) Quantization of Prediction Error



Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

Lossy Coding with Prediction: (2) Quantization of Prediction Error



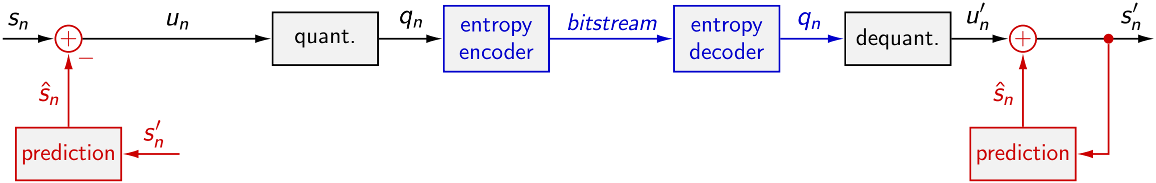
Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}\left(s'_{n-1}, s'_{n-2}, \cdots\right)$$

Lossy Coding with Prediction: (2) Quantization of Prediction Error



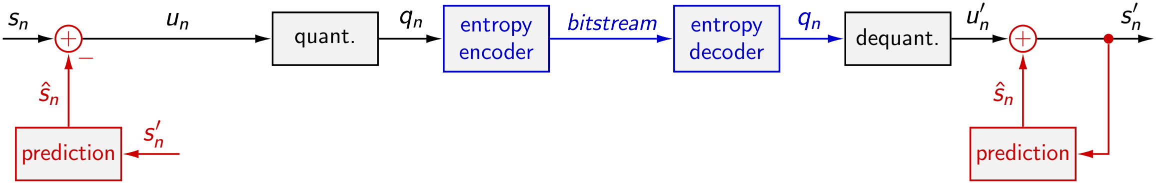
Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

Lossy Coding with Prediction: (2) Quantization of Prediction Error



Prediction Before Quantization

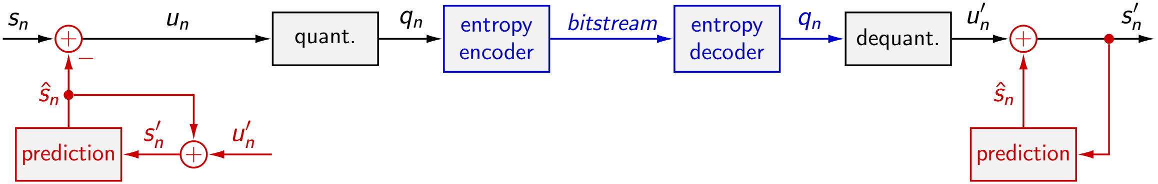
- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

➡ Encoder includes decoder (except for entropy decoding)

Lossy Coding with Prediction: (2) Quantization of Prediction Error



Prediction Before Quantization

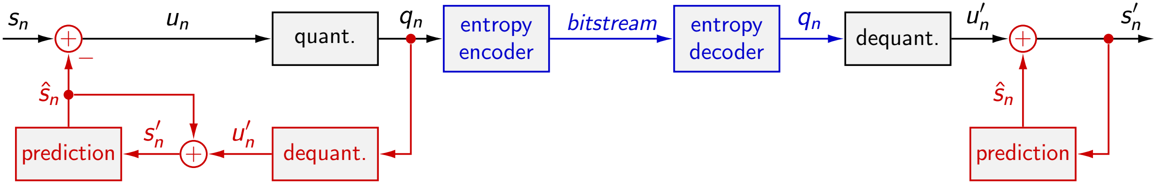
- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

➡ Encoder includes decoder (except for entropy decoding)

Lossy Coding with Prediction: (2) Quantization of Prediction Error



Prediction Before Quantization

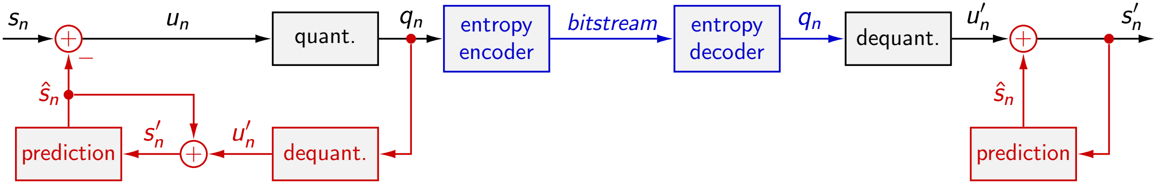
- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

➡ Encoder includes decoder (except for entropy decoding)

Lossy Coding with Prediction: (2) Quantization of Prediction Error



Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

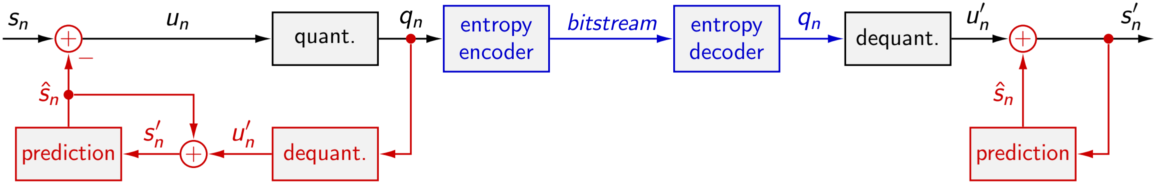
➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

➡ Encoder includes decoder (except for entropy decoding)

- Concept also referred to as **differential pulse code modulation (DPCM)**

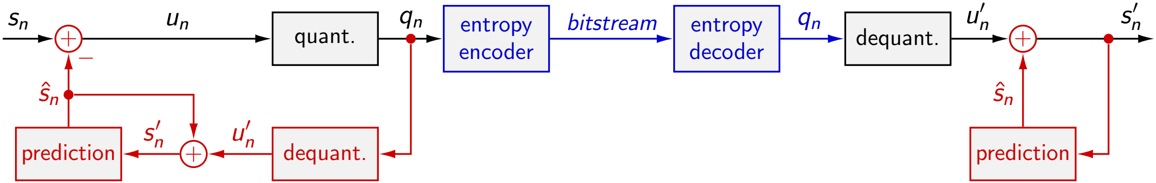
Lossy Coding with Prediction: (2) Quantization of Prediction Error



DPCM Encoder

- Prediction: $\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$
- Quantization: $q_n = \alpha(s_n - \hat{s}_n)$
- Reconstruction: $s'_n = \hat{s}_n + \beta(q_n)$
- Entropy coding of quantization indexes q_n

Lossy Coding with Prediction: (2) Quantization of Prediction Error



DPCM Encoder

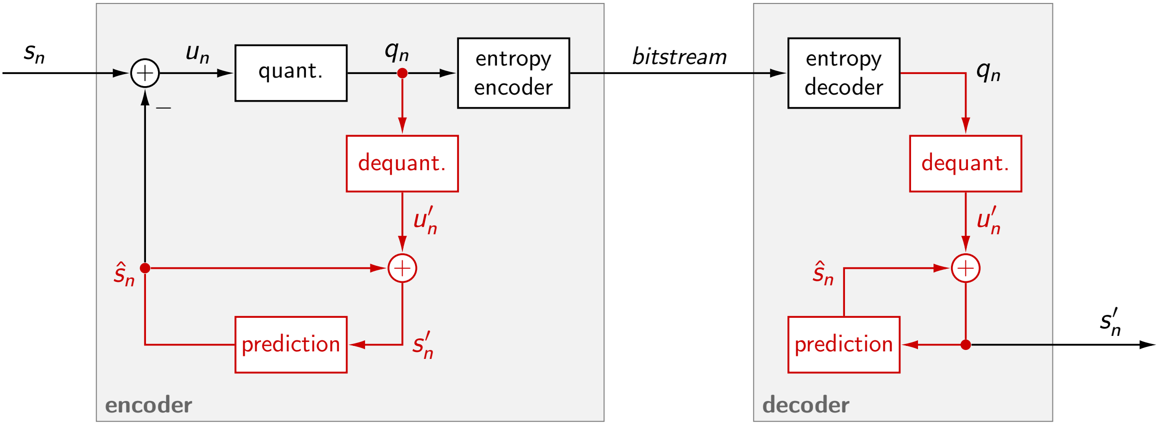
- Prediction: $\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$
- Quantization: $q_n = \alpha(s_n - \hat{s}_n)$
- Reconstruction: $s'_n = \hat{s}_n + \beta(q_n)$
- Entropy coding of quantization indexes q_n

DPCM Decoder

- Entropy decoding of quantization indexes q_n
- Prediction: $\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$
- Reconstruction: $s'_n = \hat{s}_n + \beta(q_n)$

DPCM Encoder and Decoder

- Redrawing yields typical DPCM structure
- Note: Encoder contains decoder except entropy decoding



Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer

- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error

Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer

- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error

Joint Design of Predictor and Quantizer

- Choose sufficiently large training set and Lagrange multiplier
- Start: Design predictor for original training data
- Iteratively refine quantizer and predictor
  - ➡ Design quantizer for predictor error signal (using current predictor)
  - ➡ Run DPCM encoding with current predictor and quantizer
  - ➡ Update predictor using obtained reconstructed signal

Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer

- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error

Joint Design of Predictor and Quantizer

- Choose sufficiently large training set and Lagrange multiplier
- Start: Design predictor for original training data
- Iteratively refine quantizer and predictor
  - ➡ Design quantizer for predictor error signal (using current predictor)
  - ➡ Run DPCM encoding with current predictor and quantizer
  - ➡ Update predictor using obtained reconstructed signal

Simplified Design

- Design predictor for original signal (i.e., neglect quantization in predictor design)
- Typically: Works reasonably well

Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step

$$S_n = \varrho \cdot S_{n-1} + Z_n \qquad \text{(dependencies are specified by correlation coefficient } \varrho\text{)}$$

- Innovation process Z_n is zero-mean Gaussian iid

Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step

$$S_n = \varrho \cdot S_{n-1} + Z_n \qquad \text{(dependencies are specified by correlation coefficient } \varrho\text{)}$$

- Innovation process Z_n is zero-mean Gaussian iid

DPCM Coding of Gauss-Markov Source

- Consider optimal linear predictor for original source (i.e., one-tap filter $a = \varrho$)

$$\hat{s}_n = \varrho \cdot s'_{n-1} = \varrho \cdot (s_{n-1} - x_{n-1}) \qquad \text{with quantization error} \qquad x_n = u_n - u'_n = s_n - s'_n$$

Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step

$$S_n = \varrho \cdot S_{n-1} + Z_n \qquad \text{(dependencies are specified by correlation coefficient } \varrho\text{)}$$

- Innovation process Z_n is zero-mean Gaussian iid

DPCM Coding of Gauss-Markov Source

- Consider optimal linear predictor for original source (i.e., one-tap filter $a = \varrho$)

$$\hat{s}_n = \varrho \cdot s'_{n-1} = \varrho \cdot (s_{n-1} - x_{n-1}) \qquad \text{with quantization error} \qquad x_n = u_n - u'_n = s_n - s'_n$$

➞ Resulting prediction error

$$\begin{aligned} u_n &= s_n - \hat{s}_n \\ &= s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\sigma_U^2 = \mathrm{E}\{\, U_n^2 \,\}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\sigma_U^2 = \mathrm{E}\{ U_n^2 \}$$

$$= \mathrm{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned} \sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\ &= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\ &= \mathrm{E}\{\, S_n^2 \,\} \end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\}
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\}
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\})
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}) \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad\qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}) \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}) \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}) \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} + 2\varrho \mathrm{E}\{\, Z_n E_{n-1} \,\}
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}\text{)} \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} + 2\varrho \mathrm{E}\{\, Z_n E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho \mathrm{E}\{\, Z_n E_{n-1} \,\}
\end{aligned}$$

DPCM for Gauss-Markov: Prediction Error Variance

➡ **Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

$$
\begin{aligned}
\sigma_U^2 &= \mathrm{E}\{\, U_n^2 \,\} \\
&= \mathrm{E}\{\, (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \,\} \\
&= \mathrm{E}\{\, S_n^2 \,\} + \varrho^2 \cdot \mathrm{E}\{\, S_{n-1}^2 \,\} - 2\varrho \cdot \mathrm{E}\{\, S_n S_{n-1} \,\} + \varrho^2 \cdot \mathrm{E}\{\, X_{n-1}^2 \,\} \\
&\qquad + 2\varrho \cdot \mathrm{E}\{\, S_n X_{n-1} \,\} - 2\varrho^2 \cdot \mathrm{E}\{\, S_{n-1} X_{n-1} \,\} \\
&= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \qquad \text{(note: distortion } D = \mathrm{E}\{\, (S_n - S'_n)^2 \,\}) \\
&\qquad + 2\varrho \mathrm{E}\{\, (\varrho S_{n-1} + Z_n) E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} - 2\varrho^2 \mathrm{E}\{\, S_{n-1} E_{n-1} \,\} + 2\varrho \mathrm{E}\{\, Z_n E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho \mathrm{E}\{\, Z_n E_{n-1} \,\} \\
&= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D
\end{aligned}
$$

DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

➡ Prediction error variance σ_U^2 depends on distortion D

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 D$$

DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

➡ Prediction error variance σ_U^2 depends on distortion D

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 D$$

➡ Distortion D is caused by scalar quantization of prediction error u_n

$$D(R) = \sigma_U^2 \cdot g(R)$$

where $g(R)$ is the distortion-rate function for a unit-variance Gaussian

DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

➡ Prediction error variance σ_U^2 depends on distortion D

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 D$$

➡ Distortion D is caused by scalar quantization of prediction error u_n

$$D(R) = \sigma_U^2 \cdot g(R)$$

where $g(R)$ is the distortion-rate function for a unit-variance Gaussian

➡ **Rate-dependent prediction error variance**

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 \cdot \sigma_U^2 \cdot g(R)$$

DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

➡ Prediction error variance σ_U^2 depends on distortion D

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 D$$

➡ Distortion D is caused by scalar quantization of prediction error u_n

$$D(R) = \sigma_U^2 \cdot g(R)$$

where $g(R)$ is the distortion-rate function for a unit-variance Gaussian

➡ **Rate-dependent prediction error variance**

$$\sigma_U^2 \;=\; \sigma_S^2\,(1-\varrho^2) + \varrho^2 \cdot \sigma_U^2 \cdot g(R)$$

$$\Rightarrow \quad \boxed{\sigma_U^2 \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)}}$$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R)$$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R) \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)} \cdot g(R)$$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R) \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)} \cdot g(R)$$

High-Rate Approximation

- High rates R: $\quad g(R) = \varepsilon^2 \cdot 2^{-2R} \quad$ and $\quad g(R) \ll 1$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R) \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)} \cdot g(R)$$

High-Rate Approximation

- High rates R: $\quad g(R) = \varepsilon^2 \cdot 2^{-2R} \quad$ and $\quad g(R) \ll 1$

➡ Operational distortion-rate function at high rates

$$D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1-\varrho^2) \cdot 2^{-2R}$$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R)$$

High-Rate Approximation

- High rates R: $g(R) = \varepsilon^2 \cdot 2^{-2R}$ and $g(R) \ll 1$

➡ Operational distortion-rate function at high rates

$$D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2) \cdot 2^{-2R}$$

➡ Same formula as for transform coding with large KLT $(N \to \infty)$

DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R) \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)} \cdot g(R)$$

High-Rate Approximation

- High rates R: $\quad g(R) = \varepsilon^2 \cdot 2^{-2R} \quad$ and $\quad g(R) \ll 1$

➡ Operational distortion-rate function at high rates

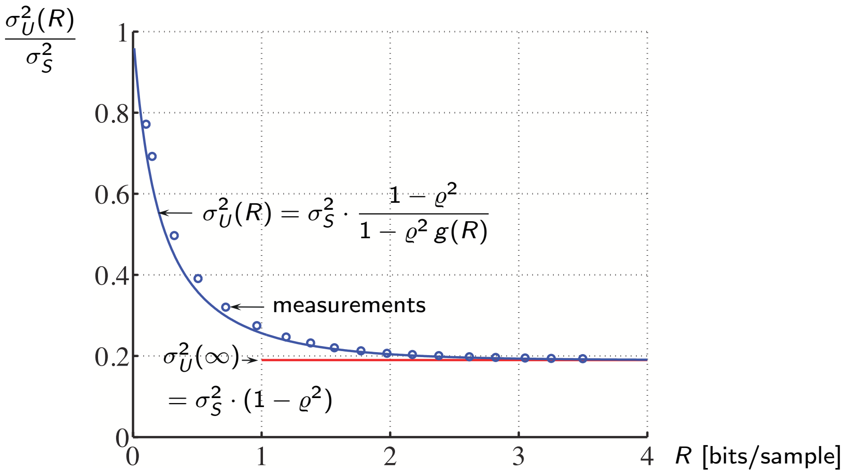
$$D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1-\varrho^2) \cdot 2^{-2R}$$

➡ Same formula as for transform coding with large KLT $\quad (N \rightarrow \infty)$

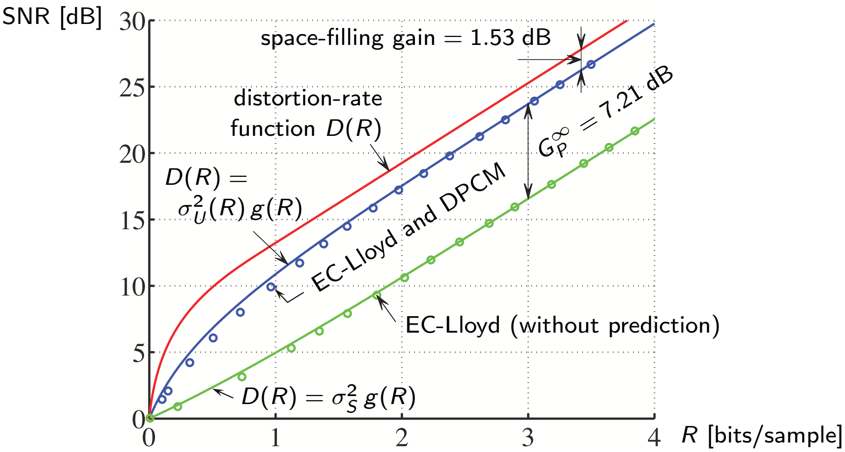
➡ Gap to rate-distortion bound represents space-filling gain of vector quantization

DPCM for Gauss-Markov: Experimental Results for $\varrho = 0.9$

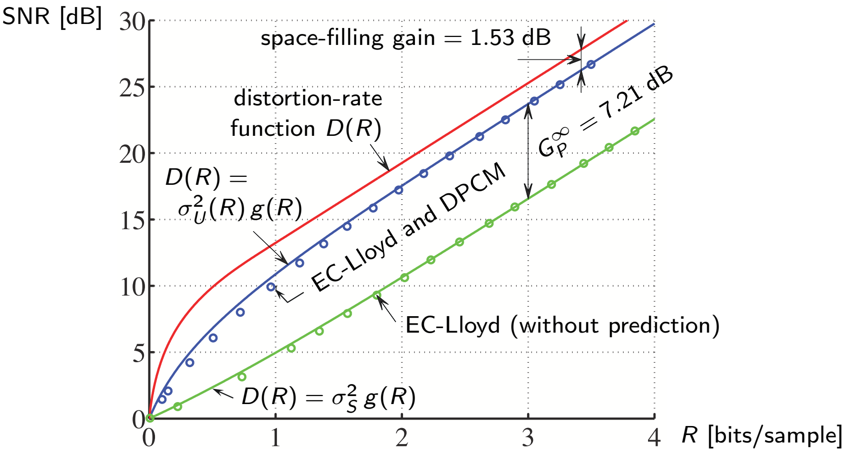
➡ Prediction error variance σ_U^2 depends on bit rate



DPCM for Gauss-Markov: Experimental Results for $\varrho = 0.9$

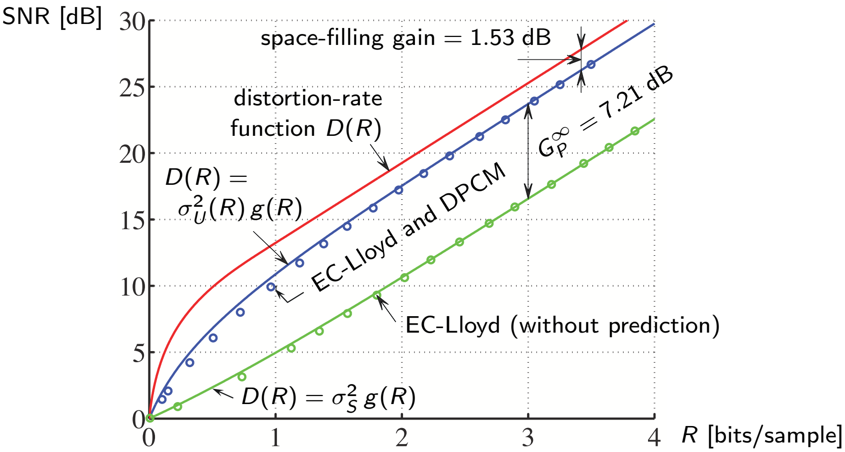


DPCM for Gauss-Markov: Experimental Results for $\varrho = 0.9$



➡ High rates: Shape and memory gain are achievable

DPCM for Gauss-Markov: Experimental Results for $\varrho = 0.9$



- High rates: Shape and memory gain are achievable
- Low rates: Worse than transform coding

Adaptive DPCM

Audio signals, images, videos

- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding

Adaptive DPCM

Audio signals, images, videos

- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding

Backward adaptation

- Simultaneously estimate predictor at encoder and decoder side
- Estimation has to be based on reconstructed samples
- No additional rate, but accuracy, complexity, error resilience issues

Adaptive DPCM

Audio signals, images, videos

- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding

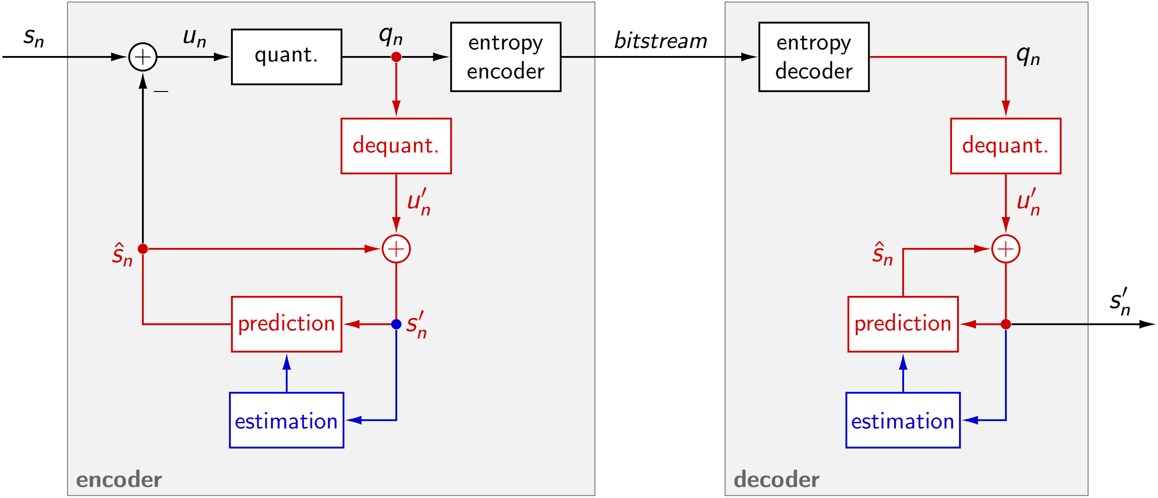
Backward adaptation

- Simultaneously estimate predictor at encoder and decoder side
- Estimation has to be based on reconstructed samples
- No additional rate, but accuracy, complexity, error resilience issues

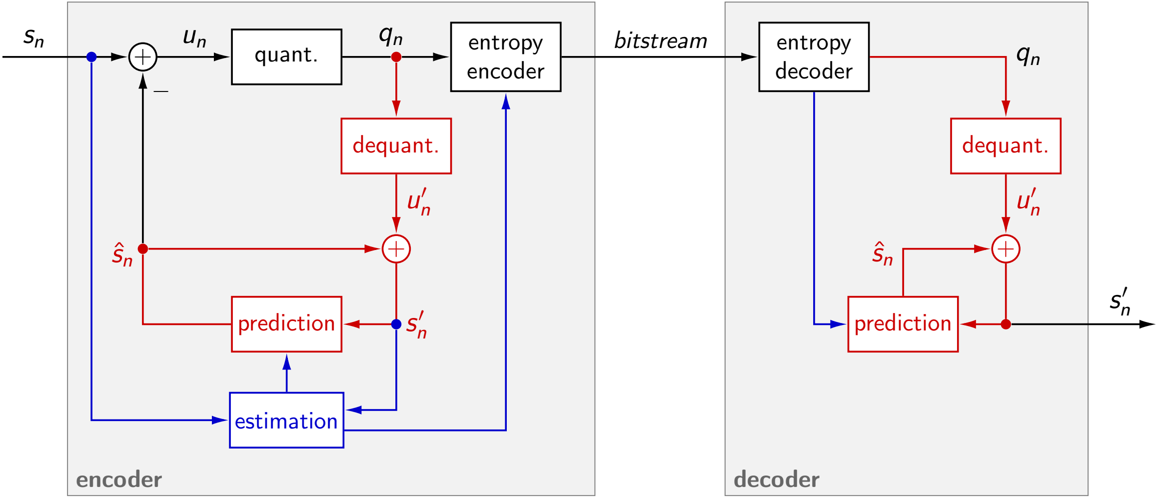
Forward adaptation

- Analyze signal at encoder side
- Send predictor as part of the bitstream (requires additional bit rate)
- Simple method: Switched predictors

Backward-Adaptive DPCM



Forward-Adaptive DPCM



Lossy Source Coding: DPCM or Transform Coding ?

Differential Pulse Coding Modulation (DPCM)

- Worse coding efficiency than transform coding at low rates (interesting opertation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

Lossy Source Coding: DPCM or Transform Coding ?

Differential Pulse Coding Modulation (DPCM)

- Worse coding efficiency than transform coding at low rates (interesting opertation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

Transform Coding

- Better coding efficiency than DPCM at low rates (interesting operation points)
- Better subjective quality than DPCM: Can better control perceived quality (frequency spectrum)
- Cannot utilize statistical dependencies between blocks (only inside blocks)

Lossy Source Coding: DPCM or Transform Coding ?

Differential Pulse Coding Modulation (DPCM)

- Worse coding efficiency than transform coding at low rates (interesting opertation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

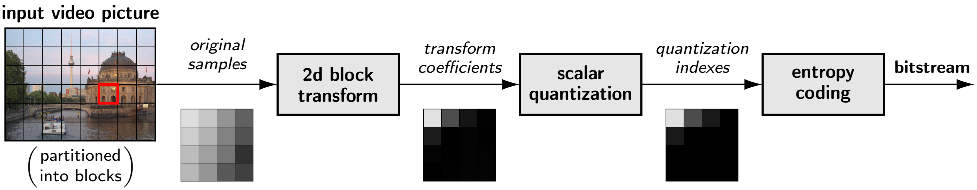
Transform Coding

- Better coding efficiency than DPCM at low rates (interesting operation points)
- Better subjective quality than DPCM: Can better control perceived quality (frequency spectrum)
- Cannot utilize statistical dependencies between blocks (only inside blocks)

Combination of Transform Coding and Prediction

- Improve coding efficiency of transform coding by prediction between transform blocks
- Two concepts:
  1. Prediction after quantization: Only for certain transform coefficients
  2. Prediction before transform: DPCM with transform coding as quantizer

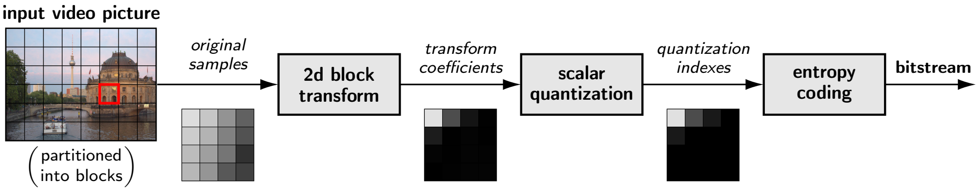
Image Coding Example: JPEG



Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables

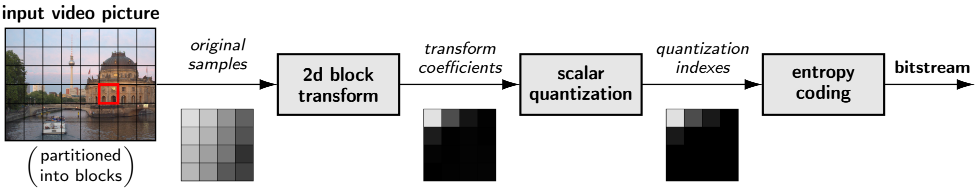
Image Coding Example: JPEG



Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   a. Prediction of quantization index for DC coefficient (mean of block): $\Delta q^{n}_{(0,0)} = q^{n}_{(0,0)} - q^{n-1}_{(0,0)}$

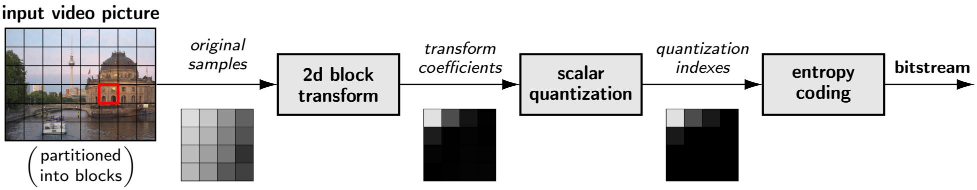
Image Coding Example: JPEG



Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   a. Prediction of quantization index for DC coefficient (mean of block): $\Delta q^n_{(0,0)} = q^n_{(0,0)} - q^{n-1}_{(0,0)}$
   b. Huffman table for prediction error $\Delta q^n_{(0,0)}$ for DC coefficient

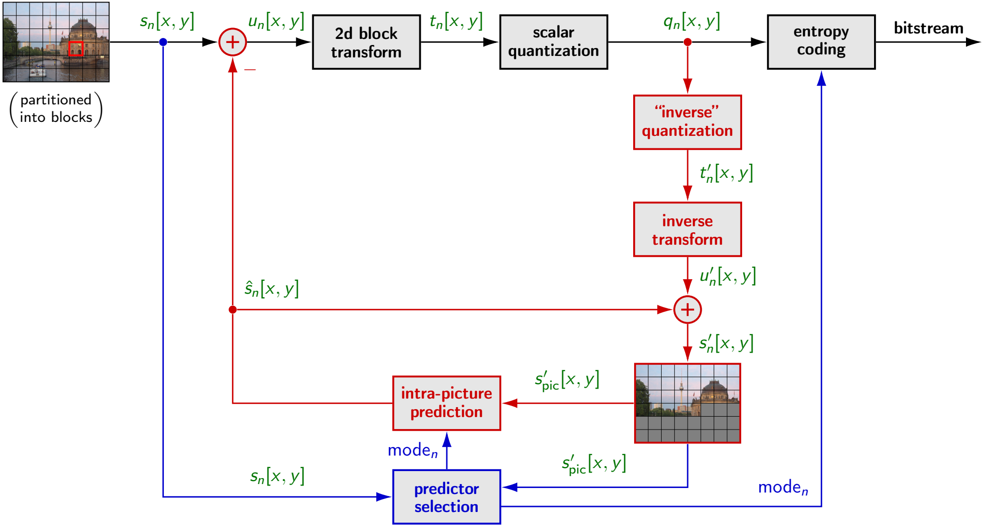
Image Coding Example: JPEG



Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   - a) Prediction of quantization index for DC coefficient (mean of block): $\Delta q^n_{(0,0)} = q^n_{(0,0)} - q^{n-1}_{(0,0)}$
   - b) Huffman table for prediction error $\Delta q^n_{(0,0)}$ for DC coefficient
   - c) Run-level coding of remaining quantization indexes (without prediction)

Modern Image Coding: Forward-Adaptive DPCM Structure



Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

Intra-Picture Prediction

- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction

Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

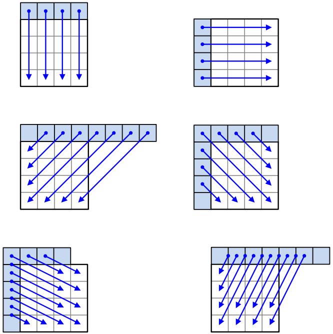
Intra-Picture Prediction

- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction

Examples

- H.264 | AVC: 9 prediction modes

intra prediction modes in H.264 | AVC



Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

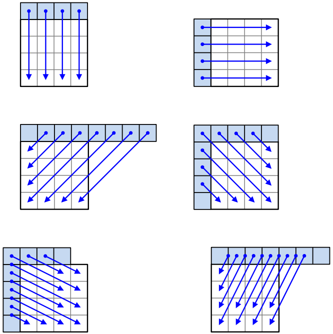
Intra-Picture Prediction

- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction

Examples

- H.264 | AVC: 9 prediction modes
- H.265 | HEVC: 35 prediction modes

intra prediction modes in H.264 | AVC



Successive Pictures in Video Sequences

successive picture are typically very similar

(exception: scene cuts)

Successive Pictures in Video Sequences

successive picture are typically very similar

(exception: scene cuts)

Important: Utilize large amount of dependencies between video picture

➡ **Basic Idea**: Predict current picture from already coded previous picture

Simple Variant: Frame Difference Coding

current original picture s_n

Simple Variant: Frame Difference Coding

current original picture s_n

- Partition current picture s_n into rectangular blocks

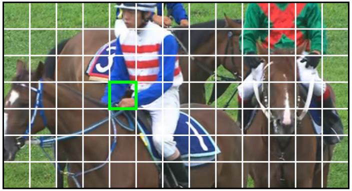
Simple Variant: Frame Difference Coding

current original picture s_n

- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n

Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}

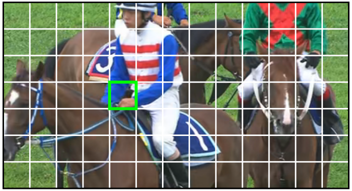


current original picture s_n

- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n

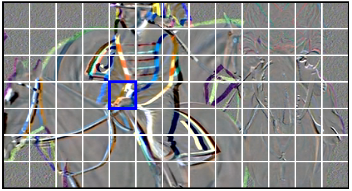
Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}



current original picture s_n

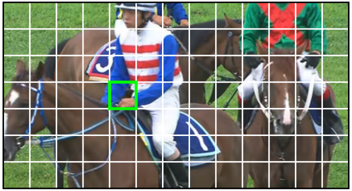
prediction error u_n (FD)



- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x, y]$

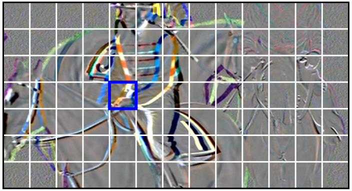
Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}



current original picture s_n

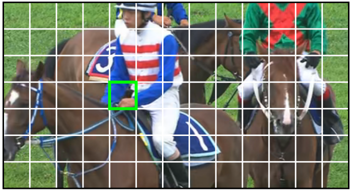
prediction error u_n (FD)



- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x,y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$

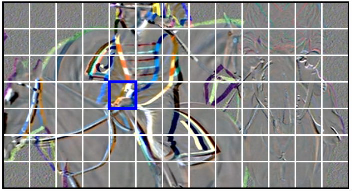
Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}



current original picture s_n

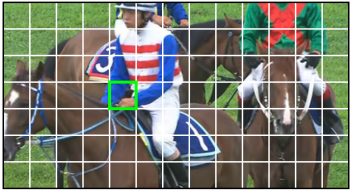
prediction error u_n (FD)



- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x,y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$

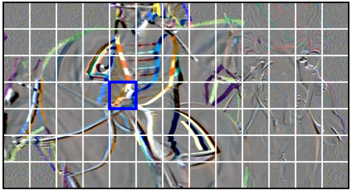
Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}



current original picture s_n

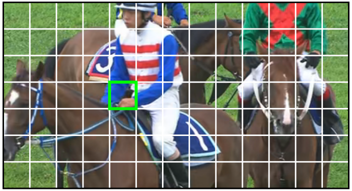
prediction error u_n (FD)



- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x,y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$
  4. Reconstruction: $s'_n[x,y] = s'_{n-1}[x,y] + u'_n[x,y]$

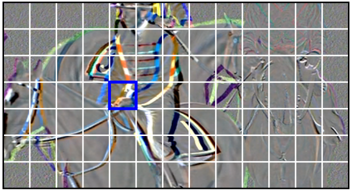
Simple Variant: Frame Difference Coding

rec. previous picture s'_{n-1}



current original picture s_n

prediction error u_n (FD)



- Partition current picture s_n into rectangular blocks
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x,y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$
  4. Reconstruction: $s'_n[x,y] = s'_{n-1}[x,y] + u'_n[x,y]$

➡ **Problem: Ineffective for moving regions**

Improvement: Motion-Compensated Prediction

current original picture s_n

Improvement: Motion-Compensated Prediction

current original picture s_n

- Partition current picture s_n into rectangular blocks

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

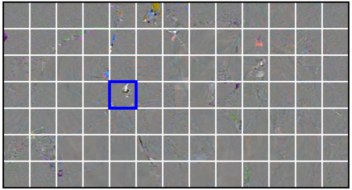
current original picture s_n

- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}
- Block-wise coding of current picture s_n

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n



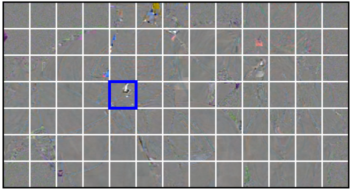
- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x+m_x, y+m_y]$

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

prediction error u_n (MCP)



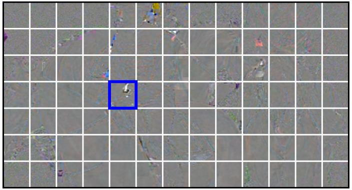
- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x+m_x, y+m_y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

prediction error u_n (MCP)



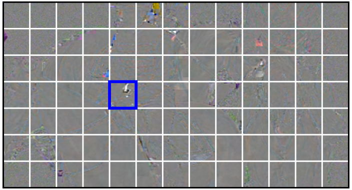
- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x+m_x, y+m_y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$
  3. Transmit in bitstream: Motion vector (m_x, m_y) and quantization indexes $\{q_k\}$

Improvement: Motion-Compensated Prediction

rec. previous picture s'_{n-1}

current original picture s_n

prediction error u_n (MCP)



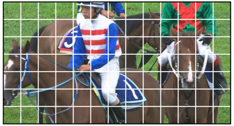
- Partition current picture s_n into rectangular blocks
- Estimate motion vectors (m_x, m_y) of blocks in current picture relative to previous picture s'_{n-1}
- Block-wise coding of current picture s_n
  1. Get prediction error: $u_n[x,y] = s_n[x,y] - s'_{n-1}[x+m_x, y+m_y]$
  2. Transform coding: $u_n[x,y] \mapsto u'_n[x,y]$
  3. Transmit in bitstream: Motion vector (m_x, m_y) and quantization indexes $\{q_k\}$
  4. Reconstruction: $s'_n[x,y] = s'_{n-1}[x+m_x, y+m_y] + u'_n[x,y]$

Hybrid Video Coding: Encoder

first input picture s_0

Hybrid Video Coding: Encoder

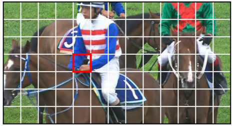
first input picture s_0



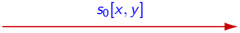
(*partitioned into blocks*)

Hybrid Video Coding: Encoder

first input picture s_0

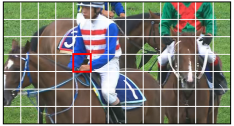


(*partitioned into blocks*)



Hybrid Video Coding: Encoder

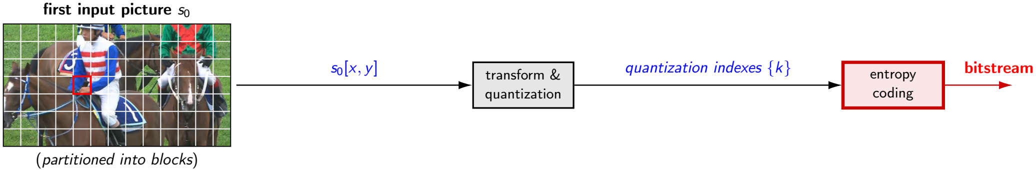
first input picture s_0



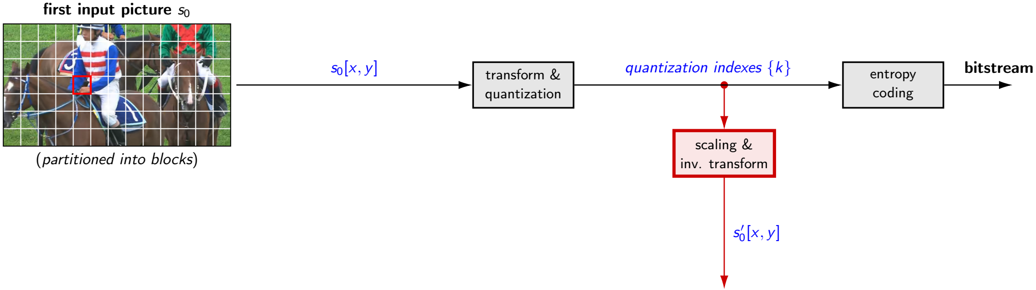
(*partitioned into blocks*)

$s_0[x, y]$ → transform & quantization → *quantization indexes* $\{k\}$

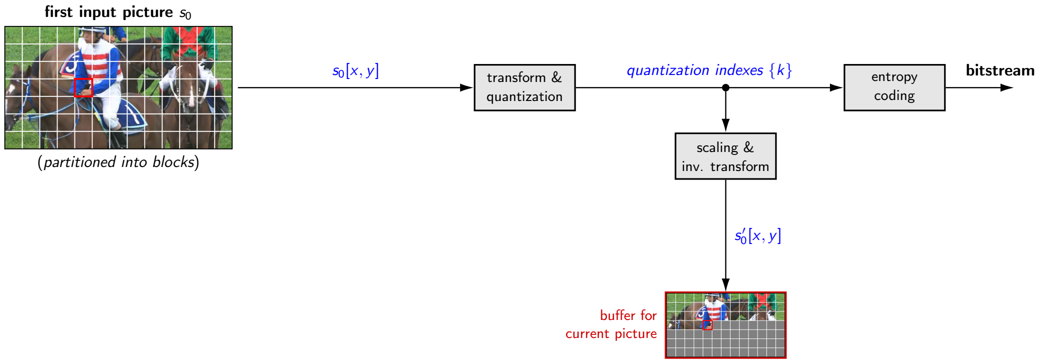
Hybrid Video Coding: Encoder



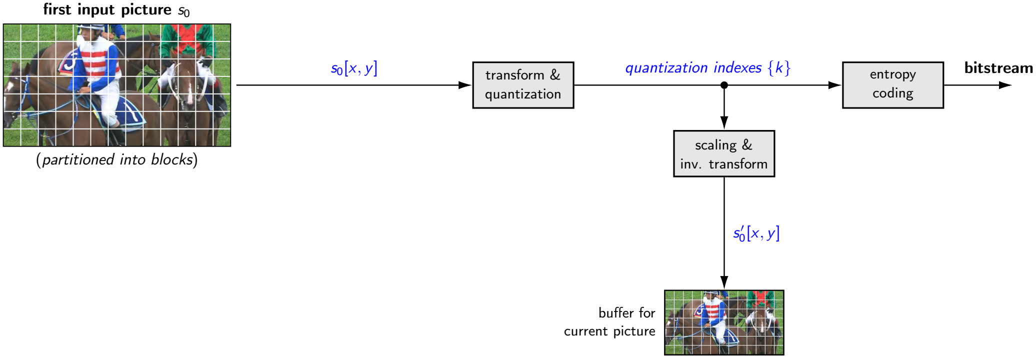
Hybrid Video Coding: Encoder



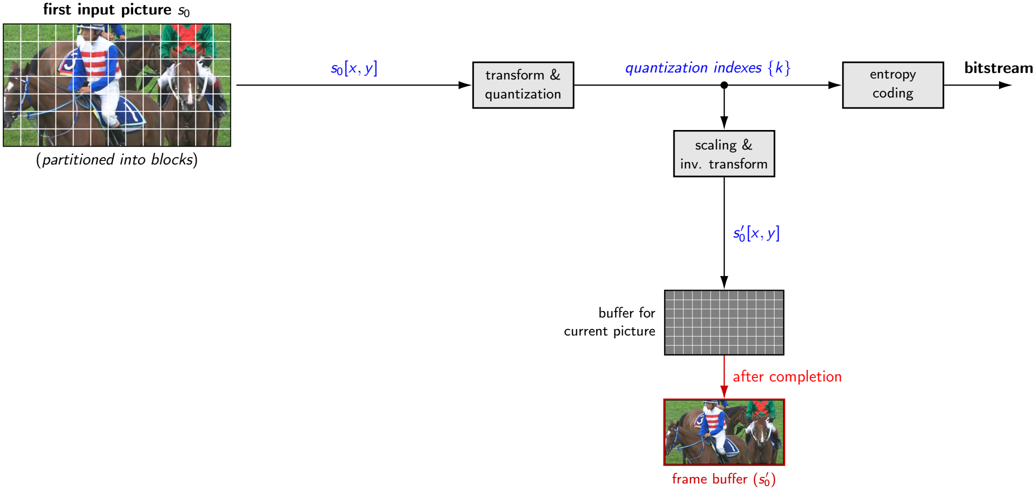
Hybrid Video Coding: Encoder



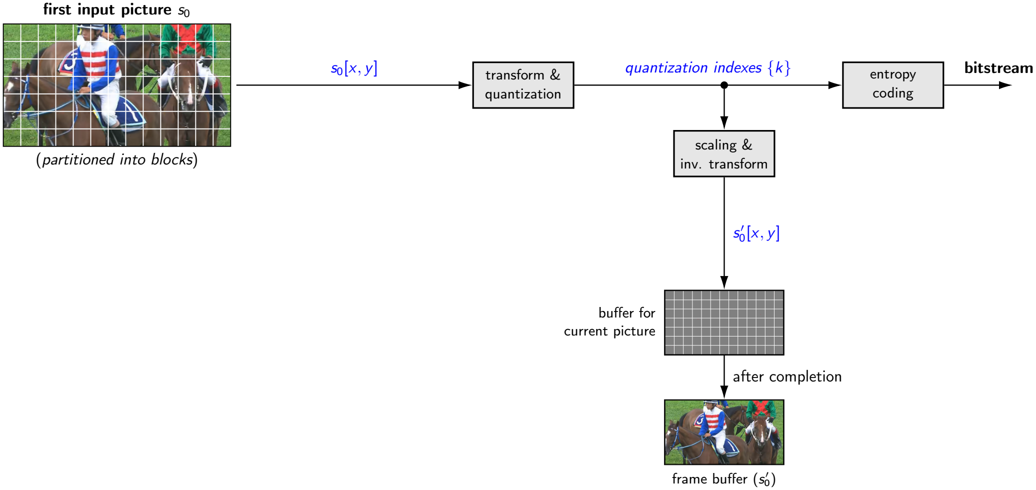
Hybrid Video Coding: Encoder



Hybrid Video Coding: Encoder



Hybrid Video Coding: Encoder



Hybrid Video Coding: Encoder

current input picture s_n

frame buffer (s'_{n-1})

Hybrid Video Coding: Encoder

current input picture s_n

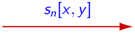
(*partitioned into blocks*)

frame buffer (s'_{n-1})

Hybrid Video Coding: Encoder

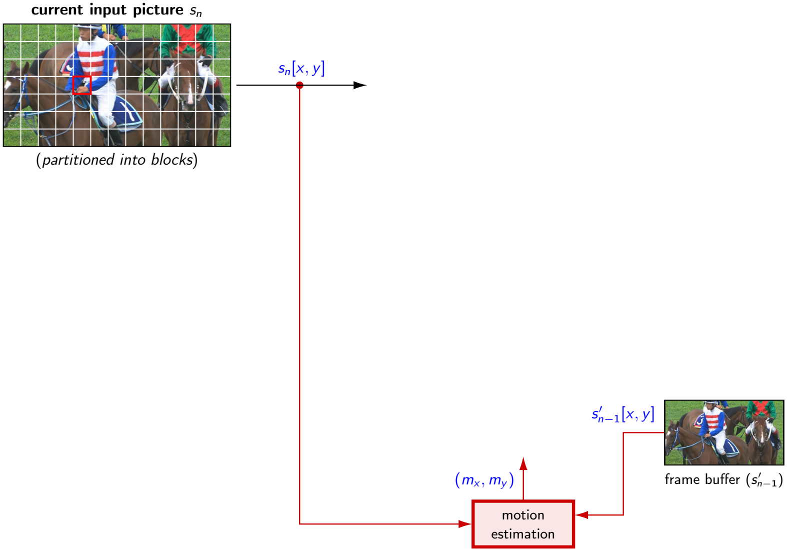
current input picture s_n

(*partitioned into blocks*)

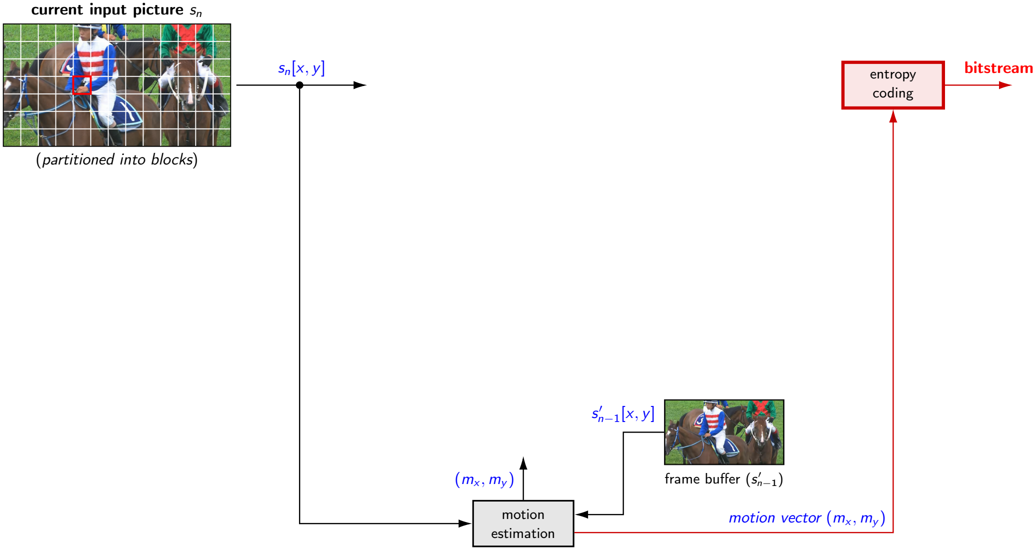


frame buffer (s'_{n-1})

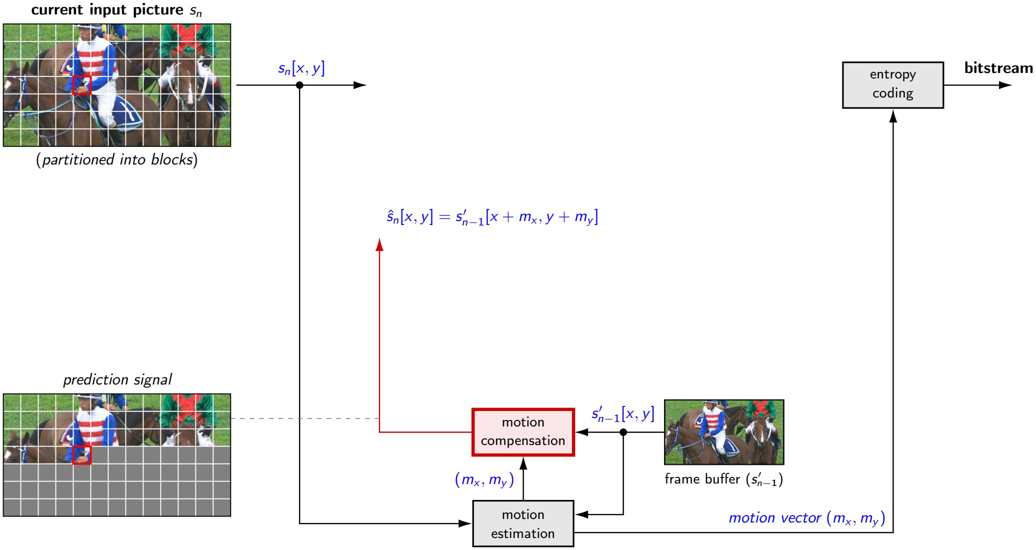
Hybrid Video Coding: Encoder



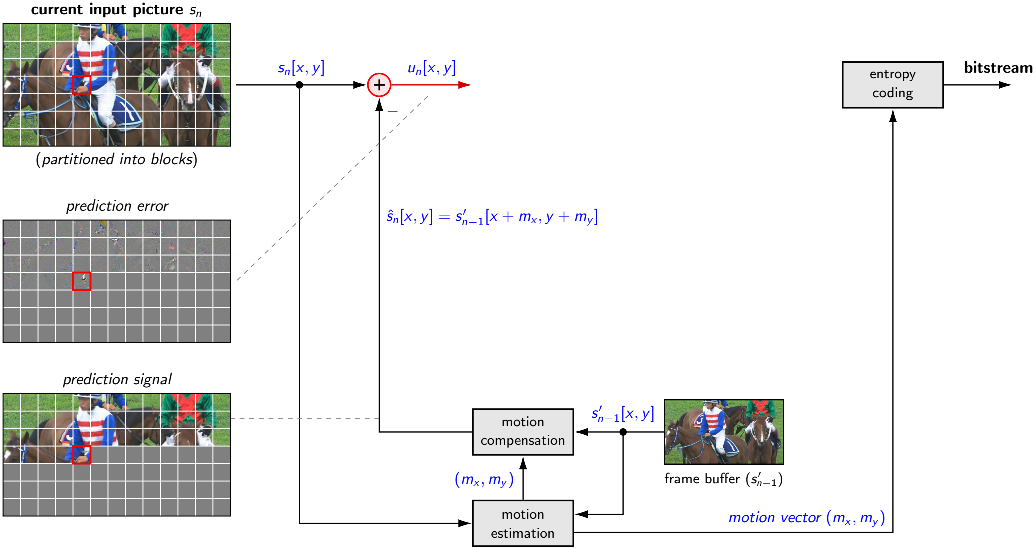
Hybrid Video Coding: Encoder



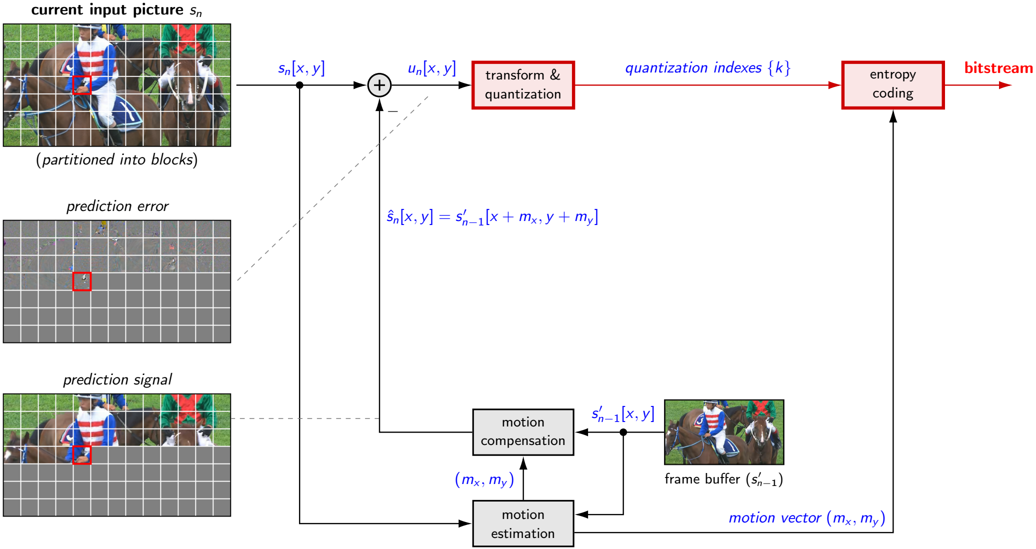
Hybrid Video Coding: Encoder



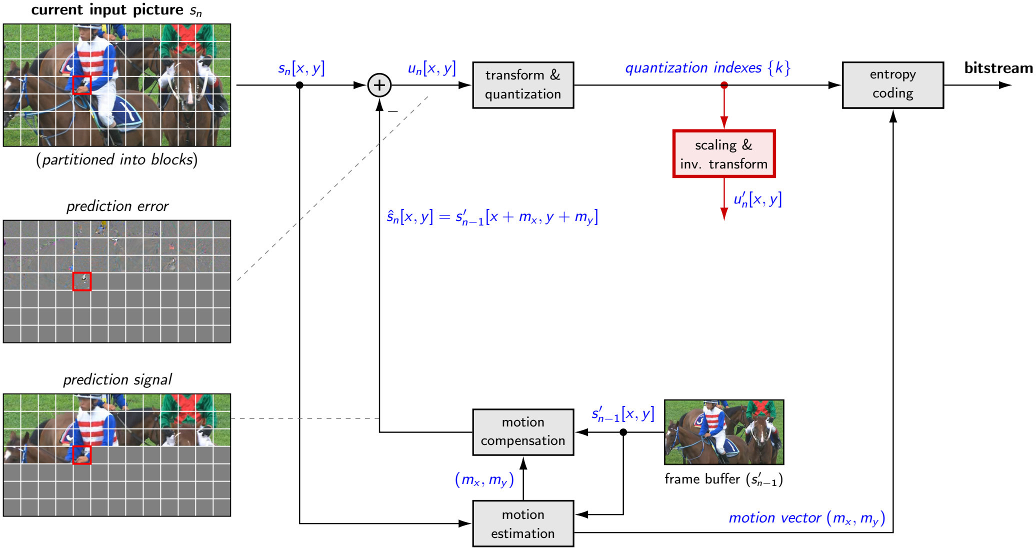
Hybrid Video Coding: Encoder



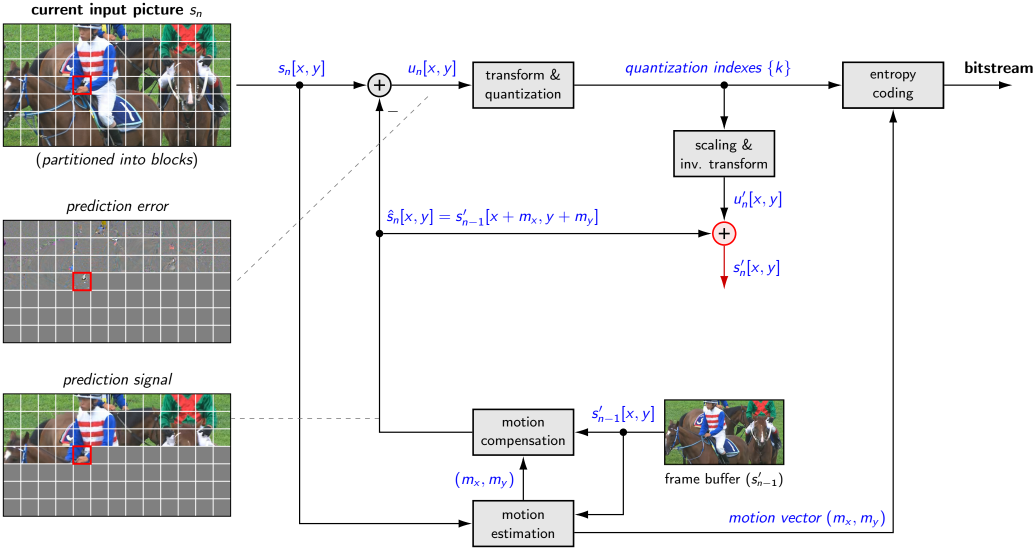
Hybrid Video Coding: Encoder



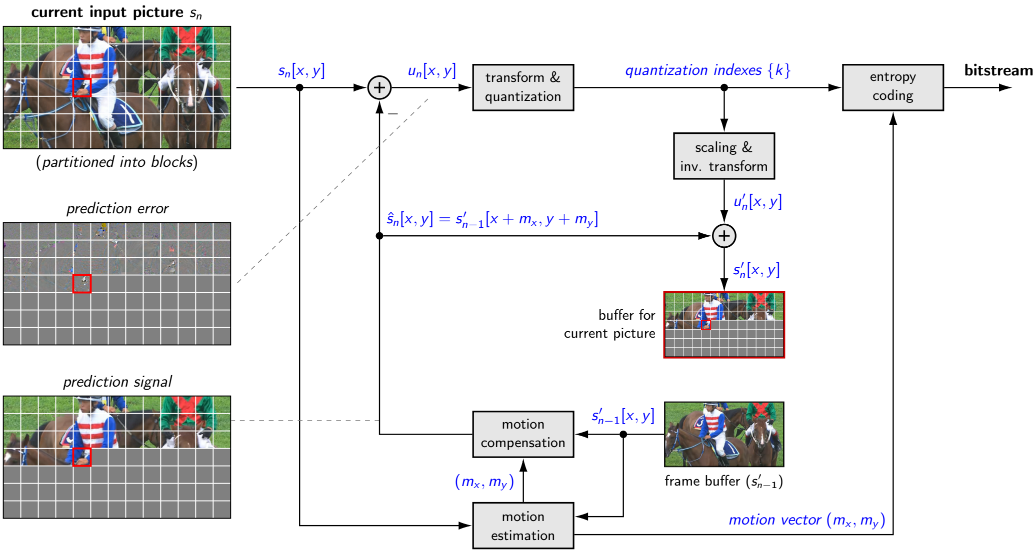
Hybrid Video Coding: Encoder



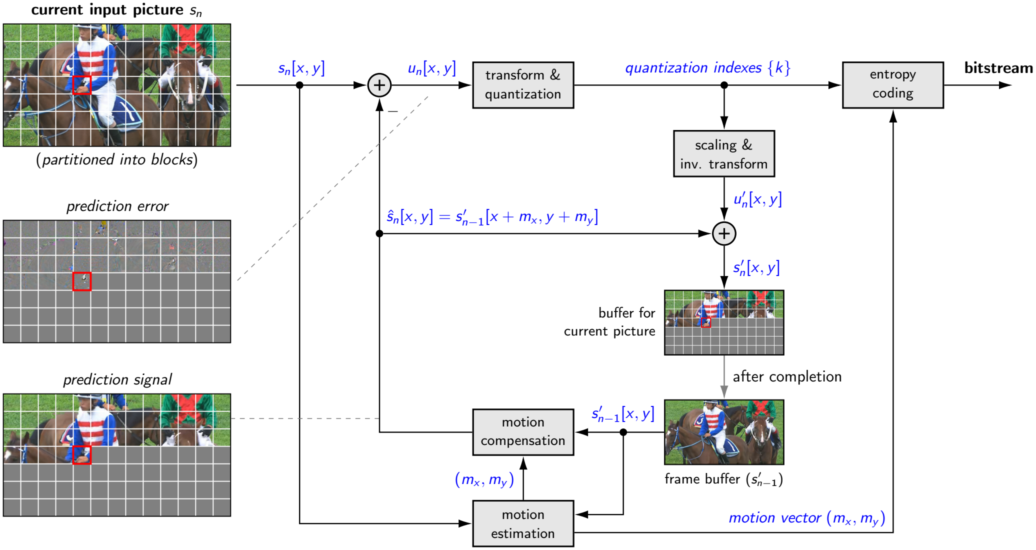
Hybrid Video Coding: Encoder



Hybrid Video Coding: Encoder



Hybrid Video Coding: Encoder



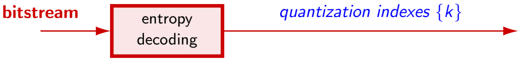
Hybrid Video Coding: Decoder

decoding of first picture

bitstream

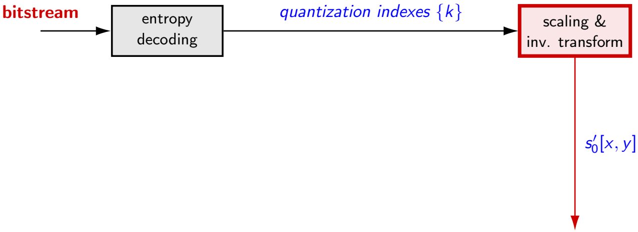
Hybrid Video Coding: Decoder

decoding of first picture



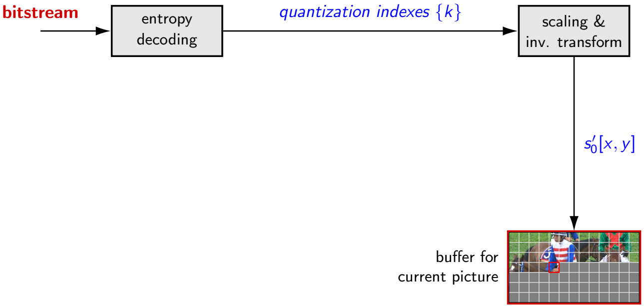
Hybrid Video Coding: Decoder

decoding of first picture



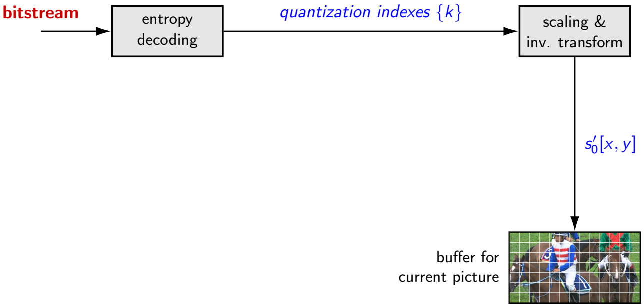
Hybrid Video Coding: Decoder

decoding of first picture



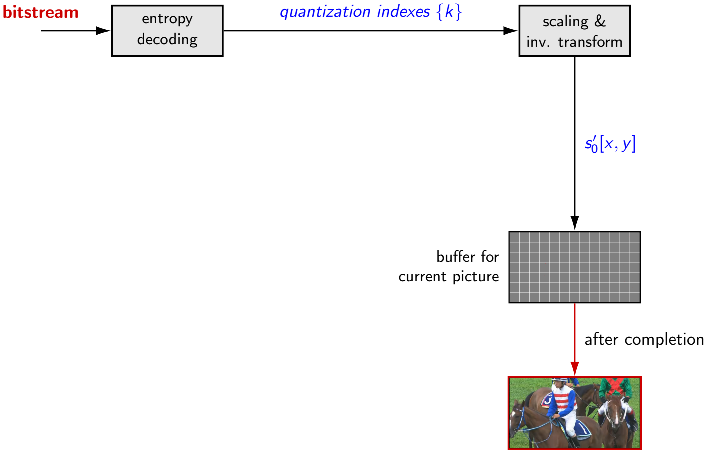
Hybrid Video Coding: Decoder

decoding of first picture



Hybrid Video Coding: Decoder

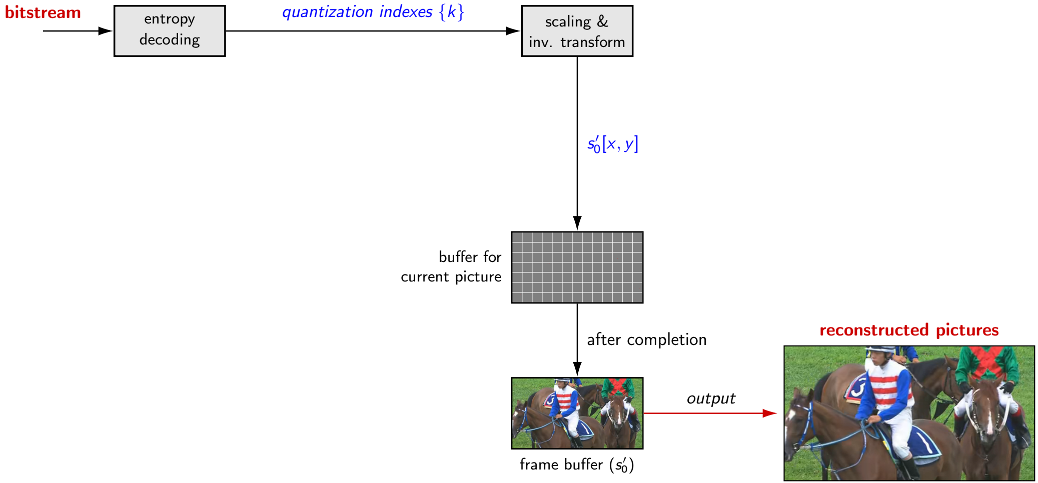
decoding of first picture



frame buffer (s'_0)

Hybrid Video Coding: Decoder

decoding of first picture



Hybrid Video Coding: Decoder

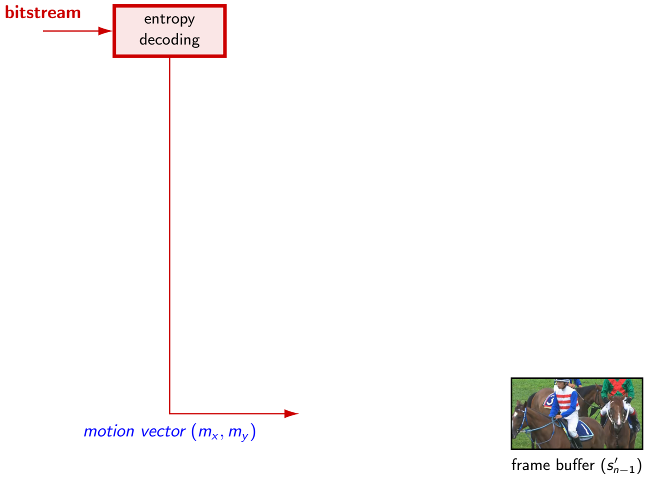
decoding of *n*-th picture

bitstream

frame buffer (s'_{n-1})

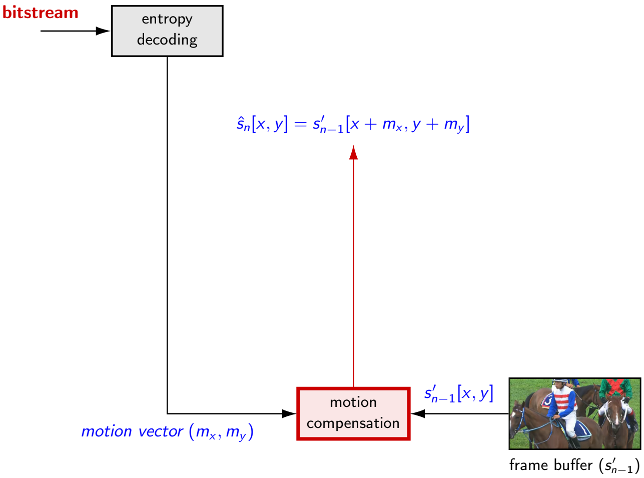
Hybrid Video Coding: Decoder

decoding of n-th picture



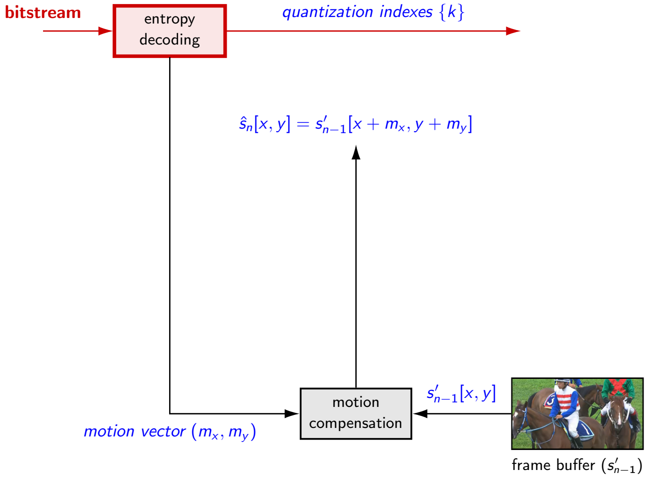
Hybrid Video Coding: Decoder

decoding of n-th picture



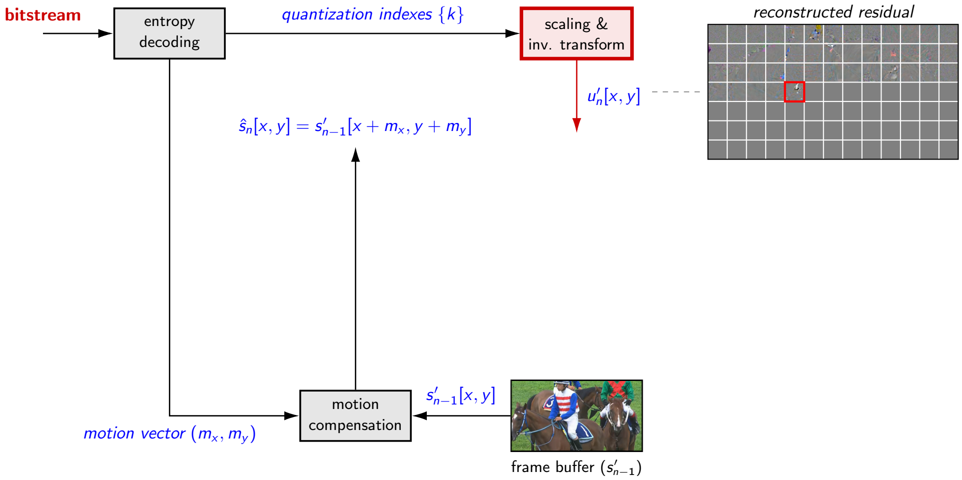
Hybrid Video Coding: Decoder

decoding of n-th picture



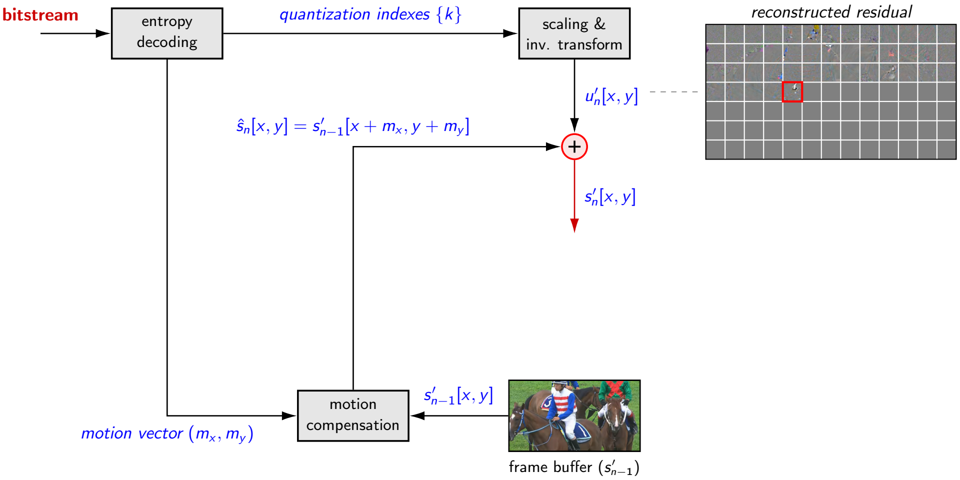
Hybrid Video Coding: Decoder

decoding of n-th picture



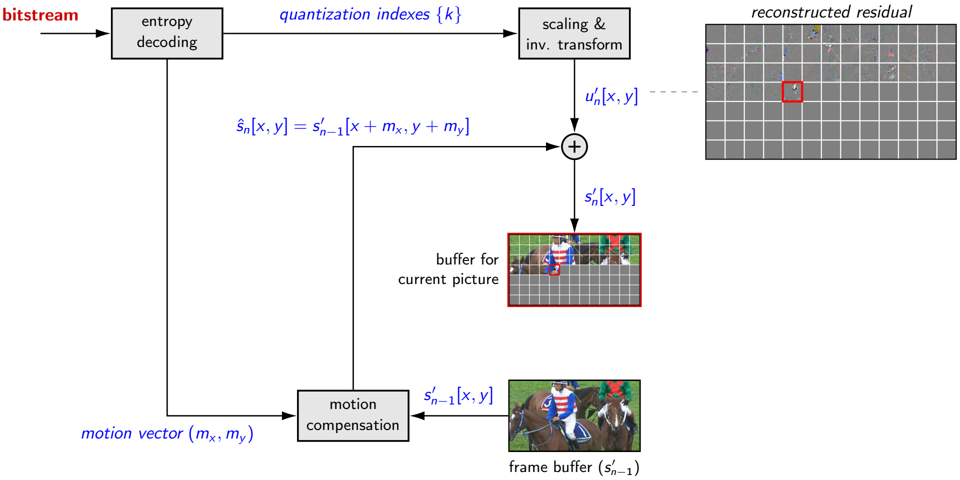
Hybrid Video Coding: Decoder

decoding of n-th picture



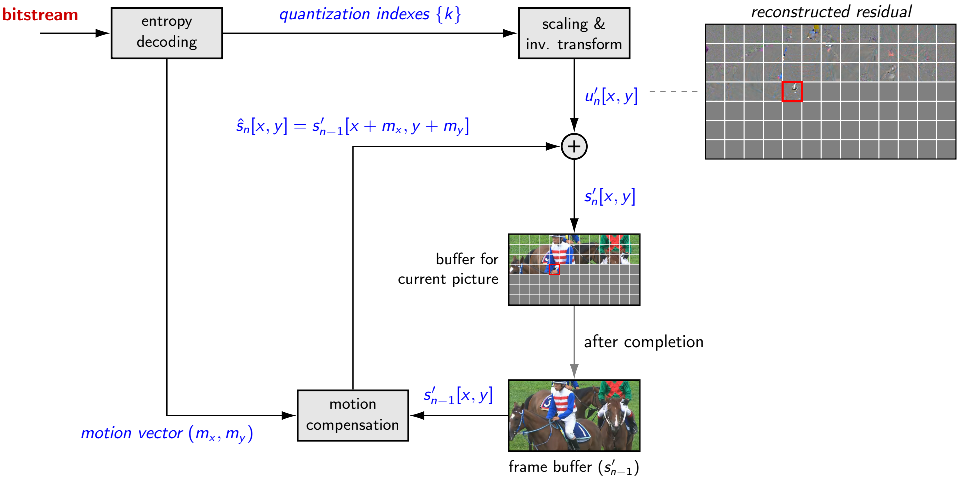
Hybrid Video Coding: Decoder

decoding of *n*-th picture



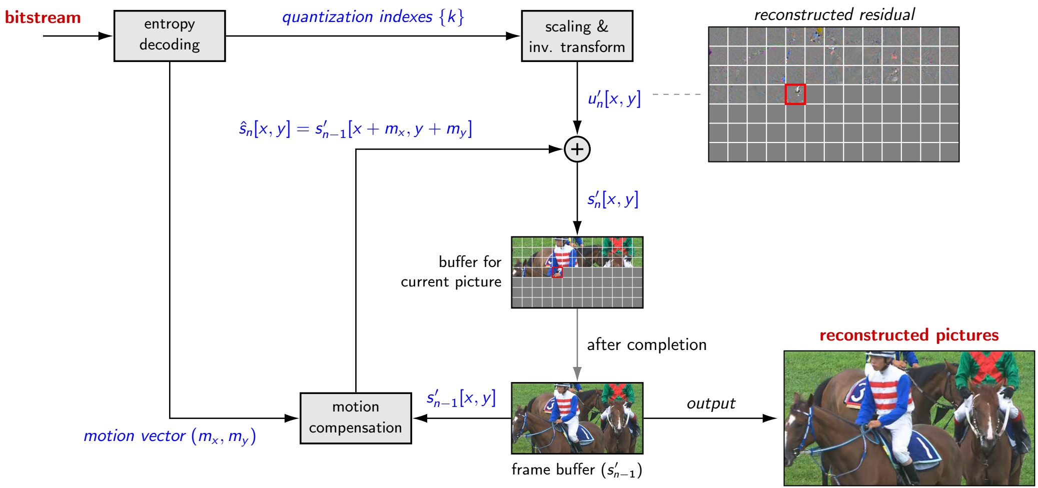
Hybrid Video Coding: Decoder

decoding of n-th picture

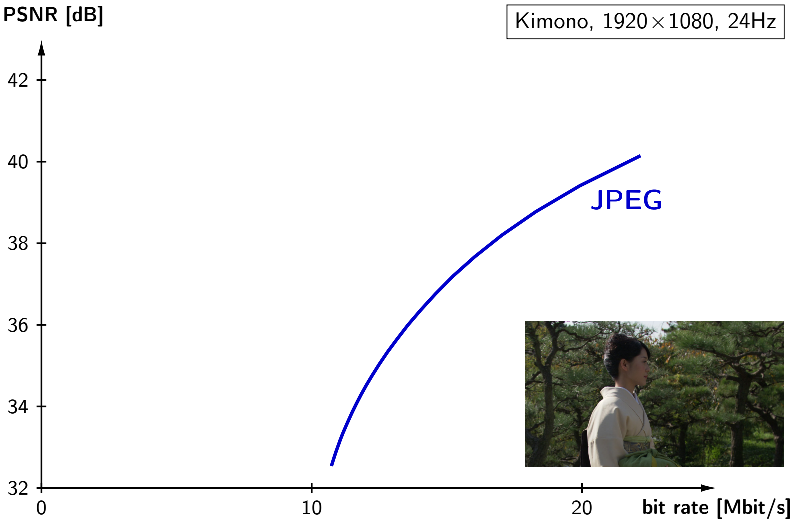


Hybrid Video Coding: Decoder

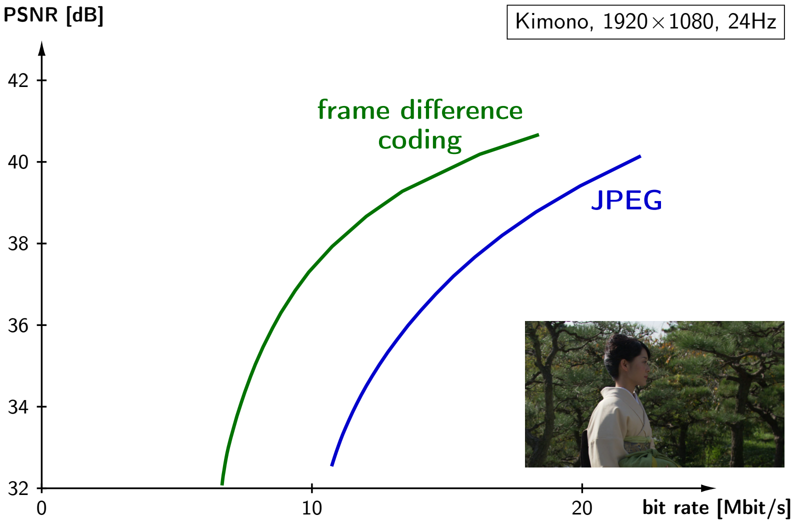
decoding of n-th picture



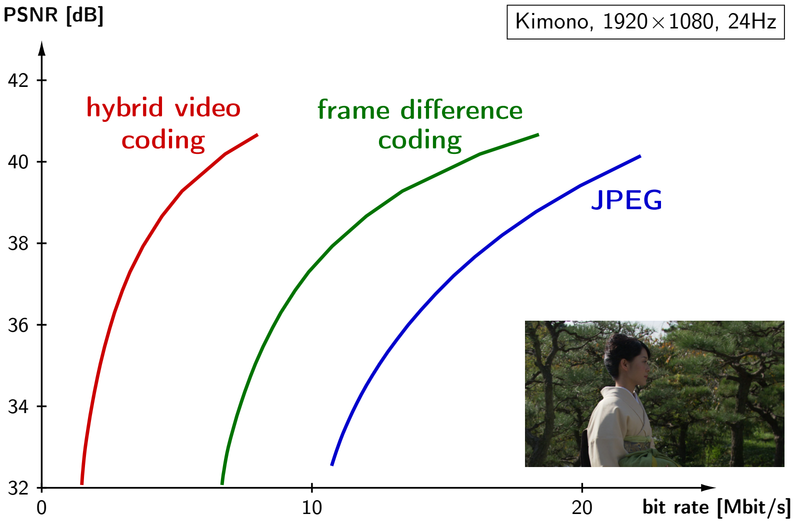
Efficiency of Hybrid Video Coding



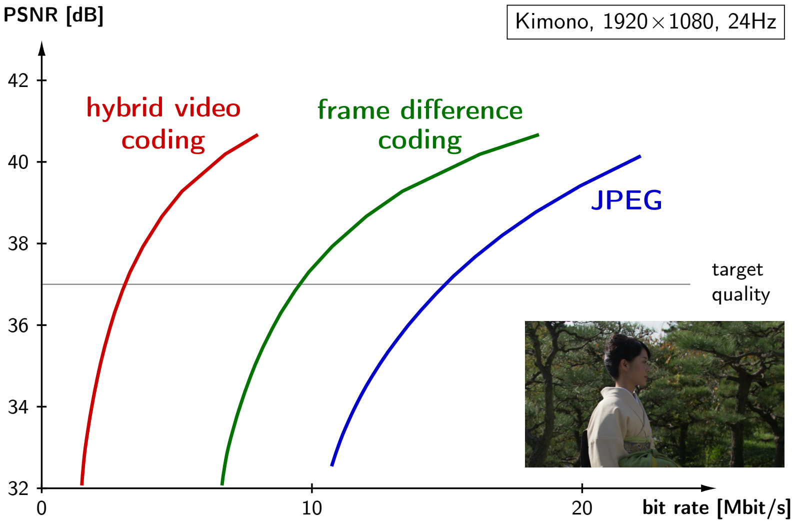
Efficiency of Hybrid Video Coding



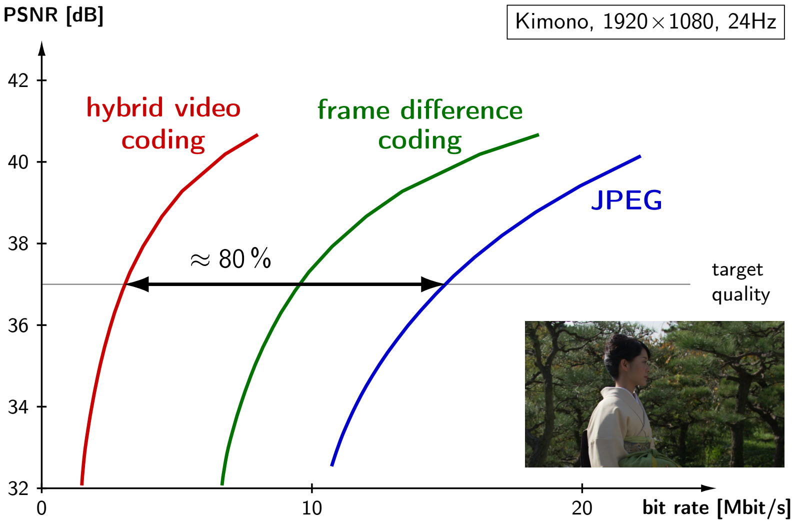
Efficiency of Hybrid Video Coding



Efficiency of Hybrid Video Coding



Efficiency of Hybrid Video Coding



Summary of Lecture

Prediction after Quantization

- Prediction of quantization indexes can improve lossless coding
- Combination with transform coding: Only useful for certain transform coefficients

Prediction before Quantization: Differential Pulse Code Modulation (DPCM)

- Reduce variance (or statistical dependencies) before quantization
- Have to use reconstructed samples for prediction in order to avoid error acculumation
- Quantization impacts quality of prediction (worse at low bit rates)
- Combination with transform coding: Prediction of complete blocks of samples

Image and Video Coding

- Combination of block-based prediction and transform coding
- Prediction of blocks of samples: Intra-picture prediction or motion-compensated prediction
- Transform coding of prediction error blocks

Exercise 1: Compare Your Lossy Codec to JPEG

Evaluate the Coding Efficiency of JPEG

- Choose a PPM image from our data base
- Encoding: Convert image to JPEG using different quality parameters (e.g., using ImageMagick)
- Decoding: Convert the JPEG file back to PPM format (e.g., using ImageMagick)
- Measure the RGB-PSNR between original and reconstructed image (tool available in KVV)
- Measure the bit rate (in bits per sample) based on size of the JPEG file

Evaluate the Coding Efficiency of your Codec

- Encode and decode the PPM image (same as for JPEG) with varying quantization step sizes
- Measure the bit rate of the compressed file and the RGB-PSNR of the reconstructed image

Compare Coding Efficiency of your Codec with that of JPEG

- Plot the RGB-PSNR over the bit rate for both your codec and JPEG (for multiple operation points)
- Compare your codec and JPEG by plotting the PSNR-rate curves into one diagram

Exercise 2: Lossy Image Compression Challenge

Improve your codec for lossy coding of PPM images

- Use any implementation of last weeks exercise as basis (see KVV)
- Try different simple techniques discussed in lectures and exercises

The following might be worth trying

- Use YCoCg format for actual coding (see implementations for lossless coding)
- Add prediction between transform blocks:
  - ➡ Prediction of quantization index for DC coefficient (as in JPEG); or
  - ➡ Subtract mean of surrounding reconstructed samples before transform (should be better), add that mean after reconstruction (dequantization + inverse transform) of prediction error
- Improve entropy coding of quantization indexes:
  - ➡ Adaptive arithmetic coding (see implementations for lossless coding)
  - ➡ Adaptive binary arithmetic coding (see implementations for lossless coding)