Fraunhofer FOKUS

Fraunhofer FOKUS Institute for Open Communication Systems

Content-Aware Encoding & Media Streaming

Daniel Silhavy

Agenda

(1) Video Fundamentals

(2) Video Encoding

(3) Video Quality Assessment

(4) Content Aware Encoding

Video Fundamentals

What is a video?

„A video is a visual multimedia source that combines a **sequence of images (frames)** to form a moving picture.“

Frame 1

Frame 50

Frame 70

Time

Framerate – Frames per second (FPS)

- **Frequency (rate)** at which consecutive images called frames appear on a display.
- Low framerates result in stuttering and influent movement

Framerate – Frames per second (FPS)

Categories of “standard” framerates used in video

Frames per second (FPS)	Value	Origin
23,976	24.000/1001	NTSC
24	24	Film cameras with 24Hz
25	25	PAL/SECAM
29,97	30.000/1001	NTSC
30	30	US TV receivers
48	48	iMAX film
50	50	EU video systems
59,94	60.000 / 1001	US video systems
60	60	US video systems
100	100	UHD/HFR
119,88	120.000 / 1001	UHD/HFR
120	120	UHD/HFR

- Displays can have problems when switching between framerates of different framerate “families” (for instance from 25 FPS to 60 FPS)
- Try to stay within the same framerate family for your video representations, e.g., 25 FPS and 50 FPS.

Resolution and Aspect Ratios, Bitrate

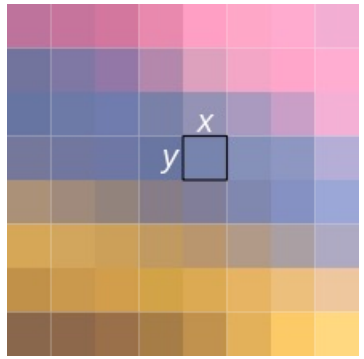
Frame height

Frame width

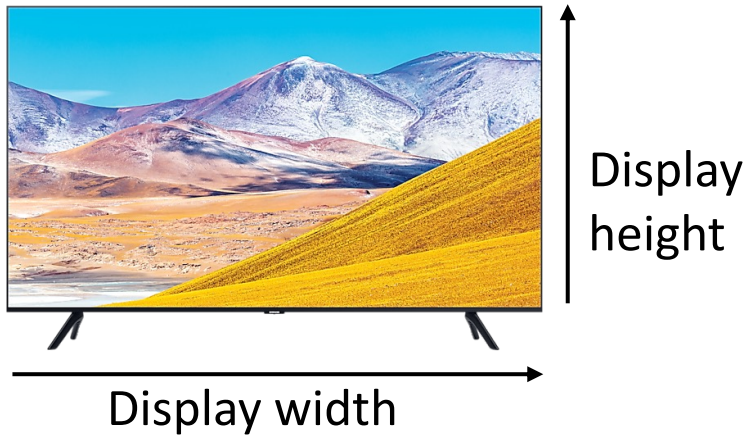
- **Resolution**: width x height
  - HD: 1280 x 720
  - Full HD: 1920 x 1080
  - 4K: 3840 x 2160

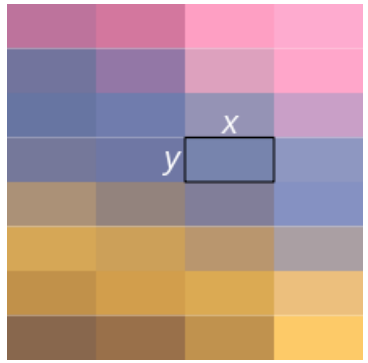
- **Bitrate**: Number of bits that are conveyed or processed in a given unit of time. Usually specified in seconds, e.g bit/s, kbit/s or mbit/s
  - e.g. 720p @ **4200 kbit/s**

Resolution and Aspect Ratio



Pixel aspect ratio 1:1





Pixel aspect ratio 2:1

- **Frame Resolution**:
  - Width x Height (pixels)
- **Frame Aspect Ratio (FAR)**
  - Width / Height
- **Display Aspect Ratio (DAR)**:
  - Display Width / Display Height
- **Sample/Pixel Aspect Ratio (SAR):**
  - Pixel Width / Pixel Height

Resolution and aspect ratios

Width	Height	DAR	SAR
256	144	16:9	1:1
384	218	16:9	1:1
480	270	16:9	1:1
640	360	16:9	1:1
960	540	16:9	1:1
1280	720	16:9	1:1
1600	900	16:9	1:1
1920	1080	16:9	1:1
2560	1440	16:9	1:1
3840	2160	16:9	1:1
7680	4320	16:9	1:1

- **Frame Resolution**:
  - Width x Height (pixels)
- **Frame Aspect Ratio**
  - Width / Height
- **Display Aspect Ratio**:
  - Display Width / Display Height
- **Sample/Pixel Aspect Ratio:**
  - Pixel Width / Pixel Height

Progressive vs. Interlaced

- Each frame consists of lines
- **Progressive**: The video signal is sampled as a series of complete frames
- **Interlaced:** The video is sampled as a series of interlaced field. Half of the data in a frame (one field) is typically sampled, representing half of the information in a complete video frame.

Progressive

Interlaced: Top Field

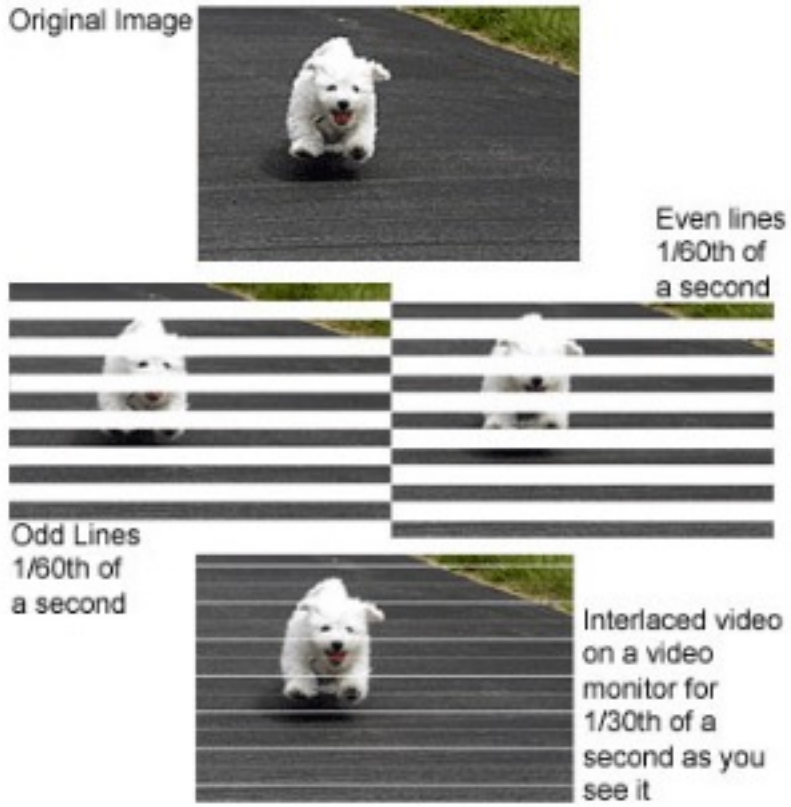
Interlaced: Bottom Field

Progressive vs. Interlaced - Examples

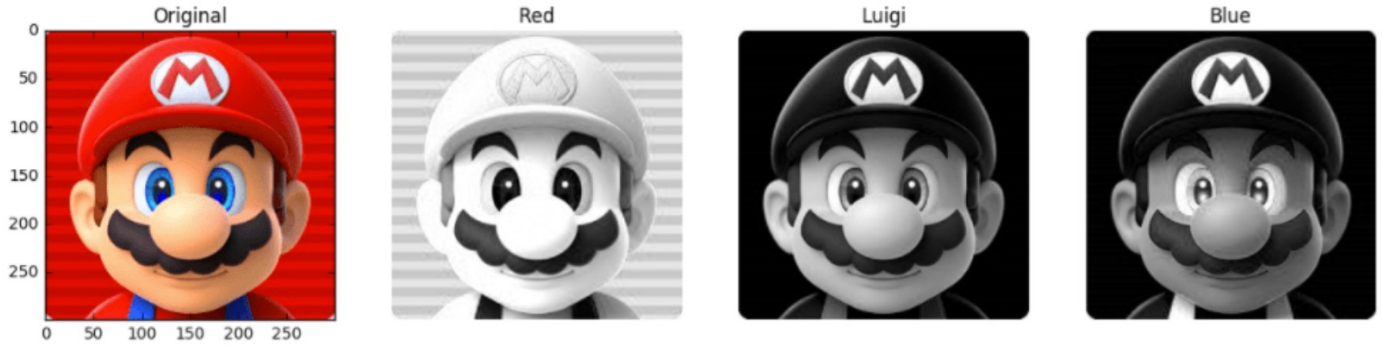
Interlaced video shown in a progressive format.

Deinterlaced video

- **Common Notation**
  - Progressive: 1080**p**
  - Interlaced:1080**i**



RGB Color Space

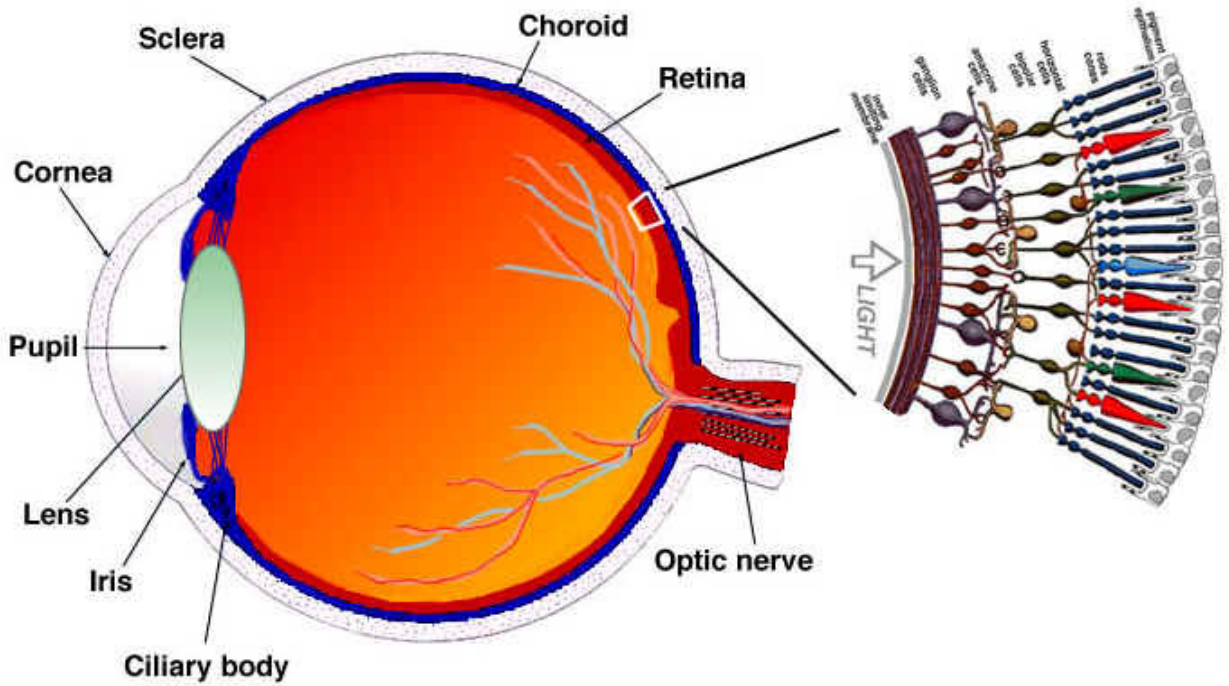


- In the **RGB color** space the relative proportion of Red, Green and Blue is represented as a number.
- Example: 8bits per color plane leads to 24bits per pixel.

Color	HTML / CSS Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
	Red	#FF0000	(255,0,0)
	Lime	#00FF00	(0,255,0)
	Blue	#0000FF	(0,0,255)
	Yellow	#FFFF00	(255,255,0)
	Cyan / Aqua	#00FFFF	(0,255,255)
	Magenta / Fuchsia	#FF00FF	(255,0,255)
	Silver	#C0C0C0	(192,192,192)
	Gray	#808080	(128,128,128)
	Maroon	#800000	(128,0,0)
	Olive	#808000	(128,128,0)
	Green	#008000	(0,128,0)
	Purple	#800080	(128,0,128)
	Teal	#008080	(0,128,128)
	Navy	#000080	(0,0,128)

The Human Visual System

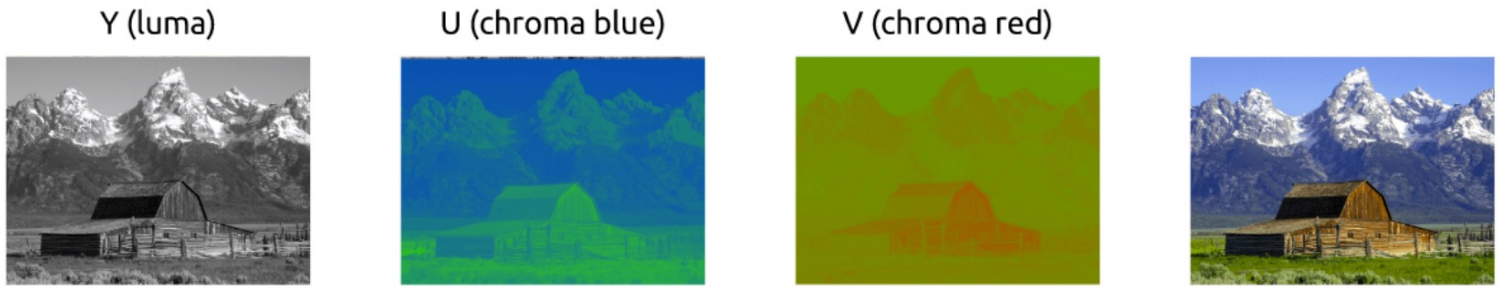
- The rod cells are mostly responsible for brightness while the cone cells are responsible for color
- There are three types of cones, each with different pigment, namely: S-cones (Blue), M-cones (Green) and L-cones (Red).
- Since we have many more rod cells (brightness) than cone cells (color), one can infer that we are more capable of distinguishing dark and light than colors



→ Since we know that we're more sensitive to luma we can try to exploit it

Color Space - YCrCb

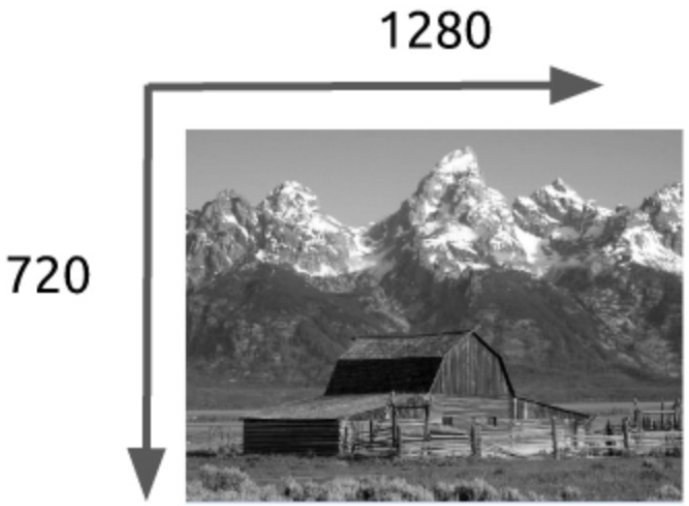
- **Idea**: Separate the luminance from the color information
- Y : Luminance component
- Cr, Cb, Cg: Chroma components
- In YCrCb only the luma (Y) and the red (Cr) and blue (Cb) components are stored.
- Before displaying the image, it is usually necessary to convert back to RGB.



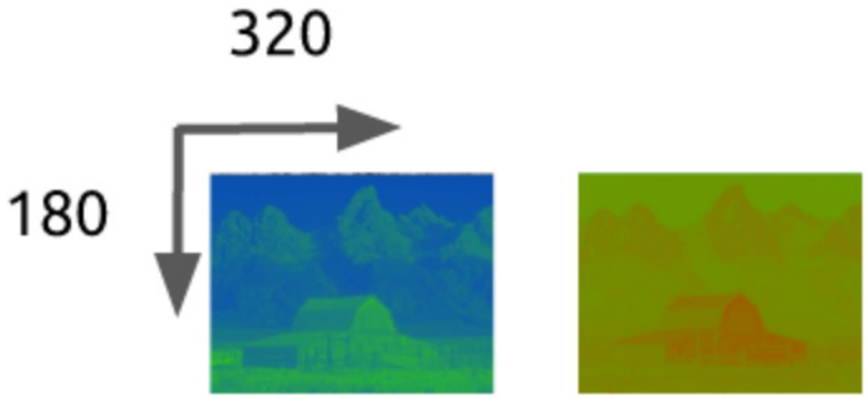
Cr and Cb can be represented with a lower resolution than Y because the HVS is less sensitive to color than luminance. **This reduces the amount of data without having an obvious effect on the visual quality.**

Chroma Subsampling

- With the image represented as luma and chroma components, we can take advantage of the human visual system's greater sensitivity for luma resolution rather than chroma to selectively remove information.
- Chroma subsampling is the technique of encoding images **using less resolution for chroma than for luma**.



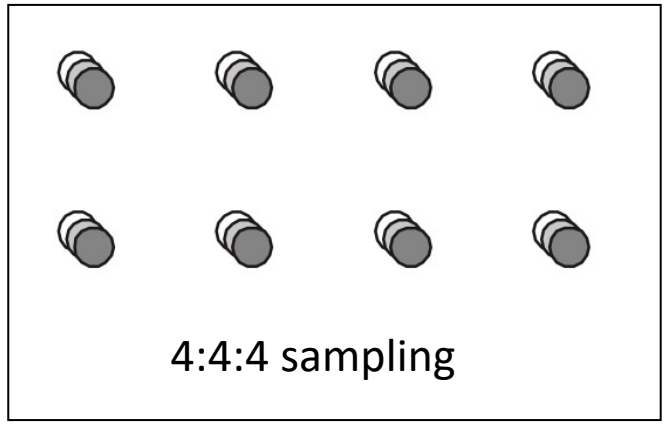
Luma resolution

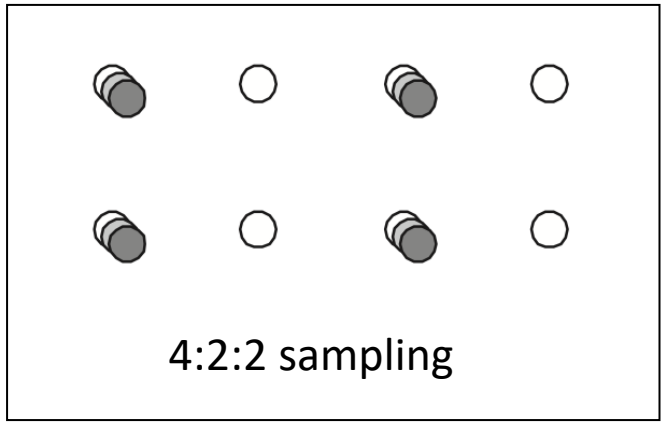


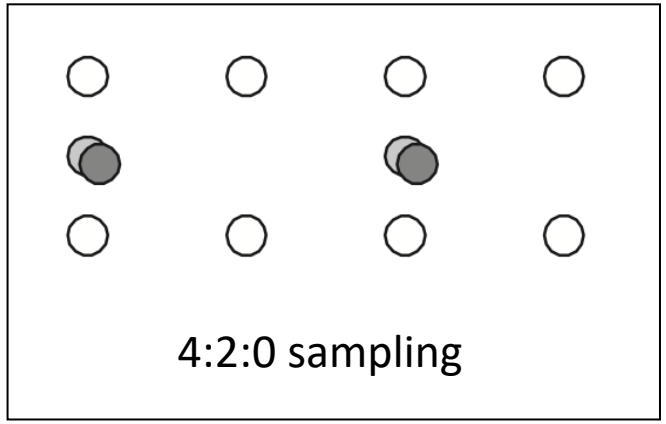
Chroma resolution

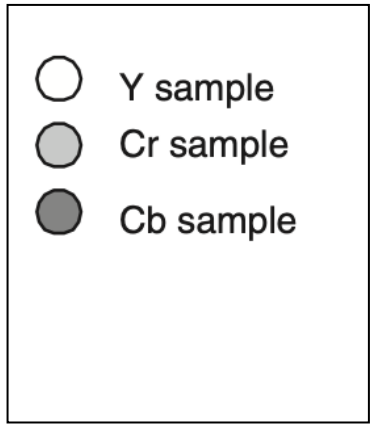
Chroma subsampling formats

- Formats are expressed as a **three-part ratio → a:x:y**
  - a: is the horizontal sampling reference (usually 4)
  - x: is the number of chroma samples in the first row of “a” pixels
  - y: is the number of changes of chroma samples between the first and seconds rows of pixels



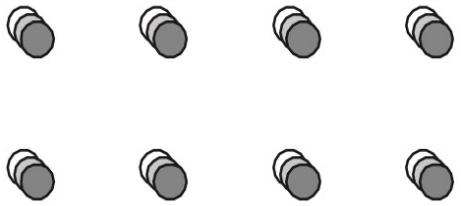






Chroma subsampling – RGB vs. 4:4:4 vs. 4:2:2 vs. 4:2:0

RGB model

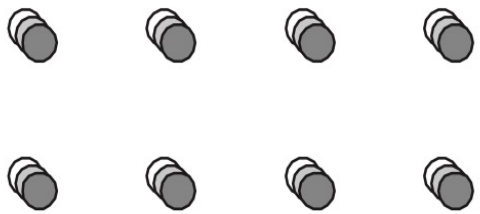


- Each pixel is represented with:
  - R: 8 bits
  - G: 8 bits
  - B: 8 bits

→ 12 samples per row

→ 24 bits per pixel

4:4:4 sampling

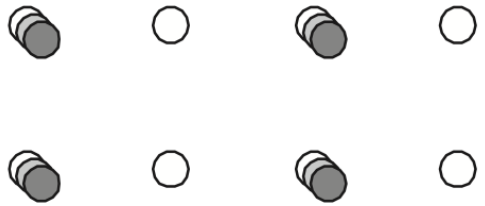


- Each pixel is represented with:
  - Y: 8 bits
  - Cr: 8 bits
  - Cb: 8 bits

→ 12 samples per row

→ 24 bits per pixel

4:2:2 sampling

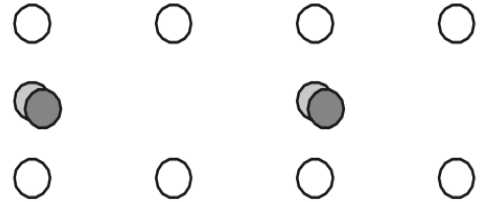


- Each pixel is represented with:
  - Y: 8 bits
  - Cr: 4 bits
  - Cb: 4 bits

→ 8 samples per row

→ 16 bits per pixel

4:2:0 sampling

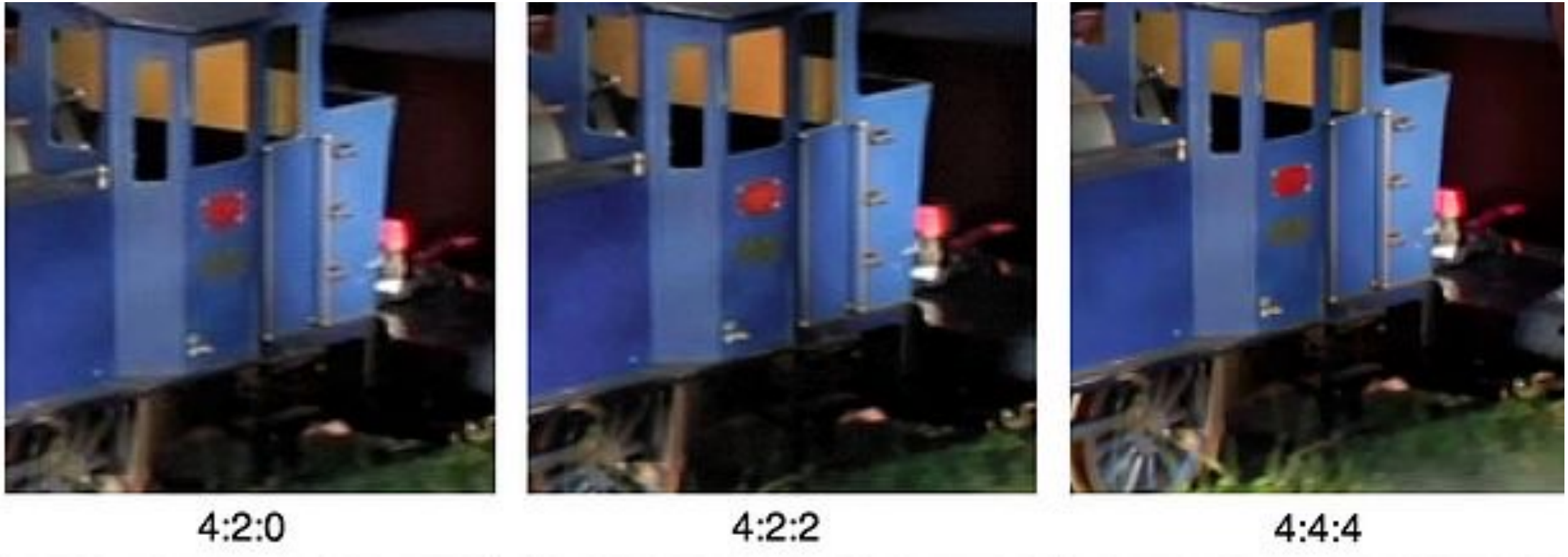


- Each pixel is represented with:
  - Y: 8 bits
  - Cr: 2 bits
  - Cb: 2 bits

→ 6 samples per row

→ 12 bits per pixel

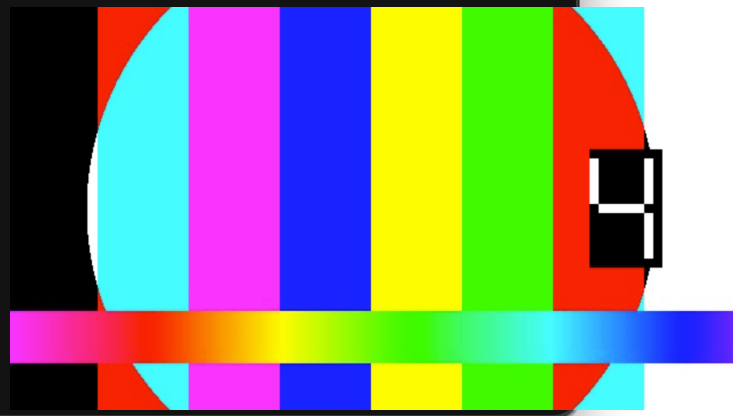
Chroma subsampling - Examples



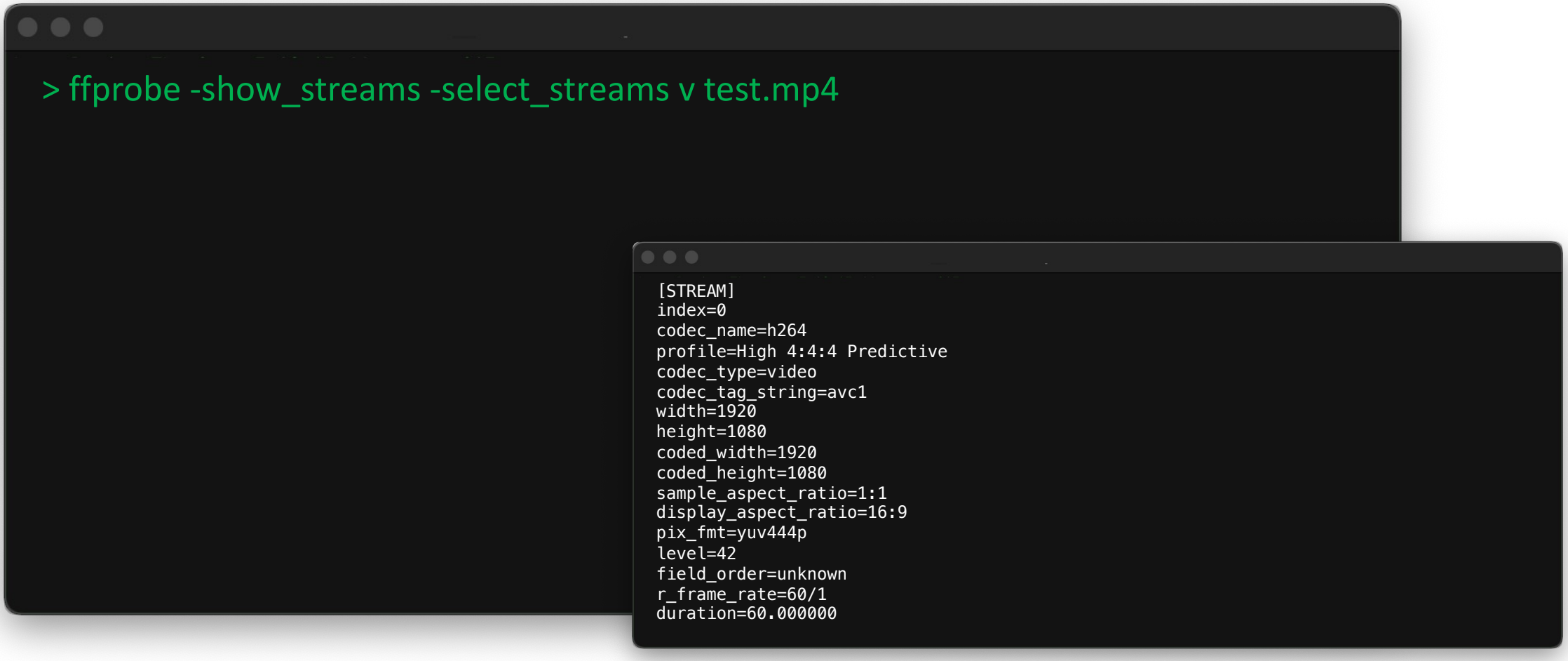
Video Hands on

Generate a test video

```
> ffmpeg -f lavfi -i testsrc=duration=30:size=1920x1080:rate=60 \
-vf "drawtext=text=%{n}:fontsize=72:r=60:x=(w-tw)/2: y=h-
(2*lh):fontcolor=white:box=1:boxcolor=0x00000099" test.mp4
```

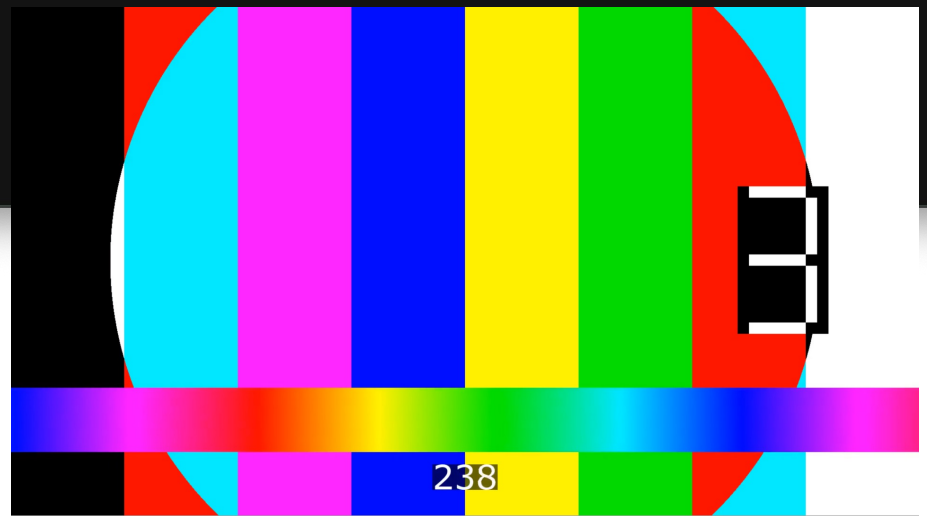


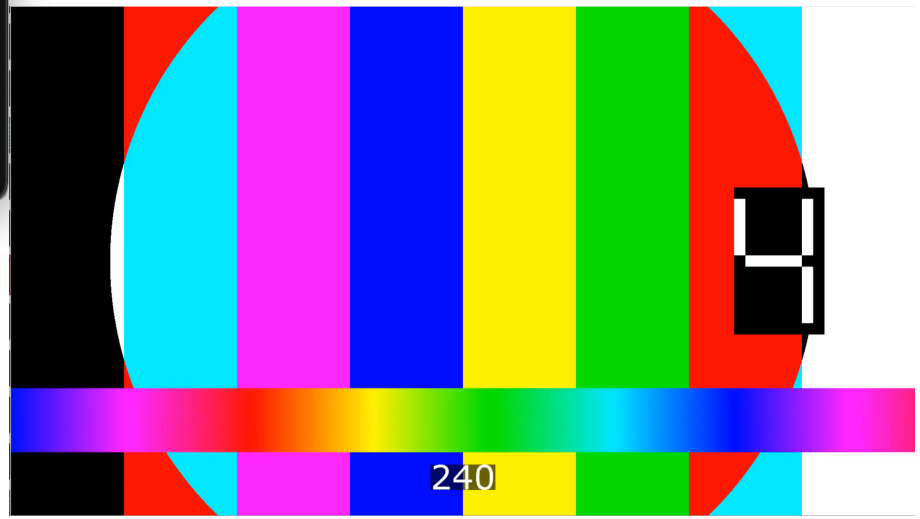
Inspect the video metadata



Changing the Framerate

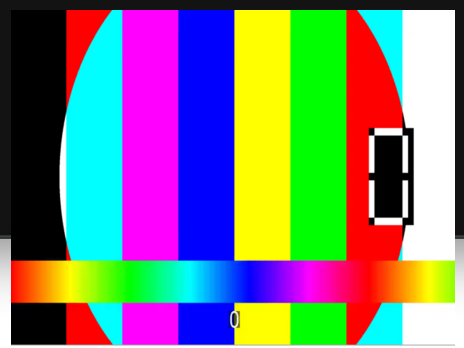
```
# Convert to 30 FPS
ffmpeg -i test.mp4 -filter:v fps=30 test_30fps.mp4

# Convert back to 60 FPS
ffmpeg -i test_30fps.mp4 -filter:v fps=60 test_60fps.mp4
```

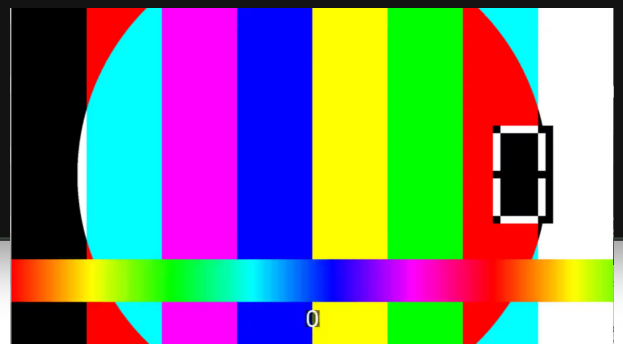




Changing the resolution

```
# 16:9
 > ffmpeg -i test.mp4 -vf scale=1280:720 ../hands_on_1/test_720.mp4

# 4:3
> ffmpeg -i test.mp4 -vf scale=640:480 ../hands_on_1/test_480.mp4

# 4:3 with pixel aspect ratio = 1:1
> ffmpeg -i test.mp4 -vf scale=640:480,setsar=1:1 ../hands_on_1/test_480_b.mp4
```





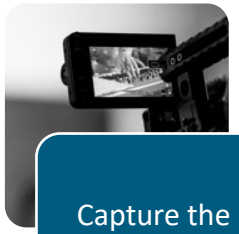
Dealing with chroma and colors

```
# Subsample to 420
> ffmpeg -i test.mp4 -c:v libx264 -vf format=yuv420p
../hands_on_1/test_420sub.mp4

# Create color histogram
ffmpeg -i caminandes_20_sec.mp4 -vf
"split=2[a][b],[b]histogram,format=yuv420p[hh],[a][hh]overlay“
../hands_on_1/test_420sub_histogram.mp4
```

Video Encoding

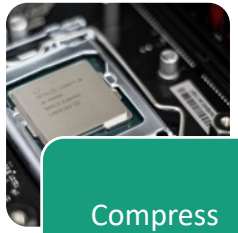
Overview - Media Streaming Chain

Media Input



Capture the media input (video, audio)

Encoding



Compress media using codecs

Packaging

Create media segments and manifest files

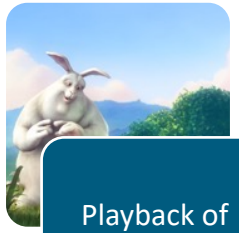
DRM Protection

Protect the content against piracy

Hosting

Host media segments and the manifest files on a CDN

Playback



Playback of the content on the client side

Monitoring

Monitor streaming performance and detect errors and failures

Why do we need video codecs?

Without encoding

$$width * height * fps * bits\ per\ pixel$$

320 * 240 * 30 * 24 = 55.3Mbit/s

With encoding

H.264: 0.235Mbit/s

Compression ratio: 235x

Introduction

- The encoder / decoder pair is often referred to **Codec** (en**CO**der / **DEC**oder)
- Data compression is achieved by removing redundancy
  - Lossless compression ratio “only” around 3-5 times
- Video **codecs are lossy** which means the decoded sequence differs from the original source
- Video coding methods exploit **temporal and spatial redundancy** to achieve compression

The basic steps

Video Source

- "Raw" video data

Prediction Model

- Intra-prediction
- Inter-prediction
- Outputs a residual frame

Spatial Model

- Discrete Cosinus Transformation (DCT)
- Quantization
- Outputs a set of quantized transform coefficients

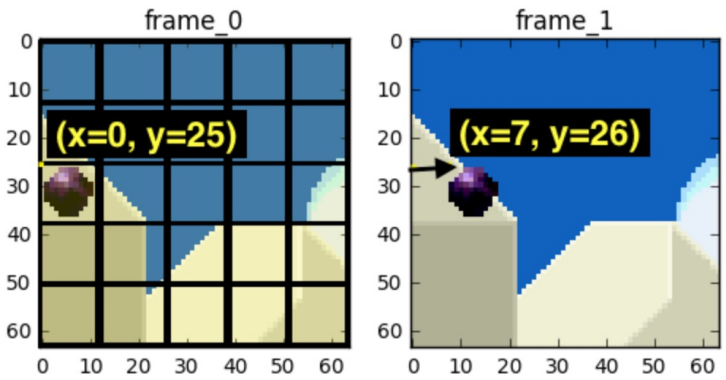
Entropy Encoder

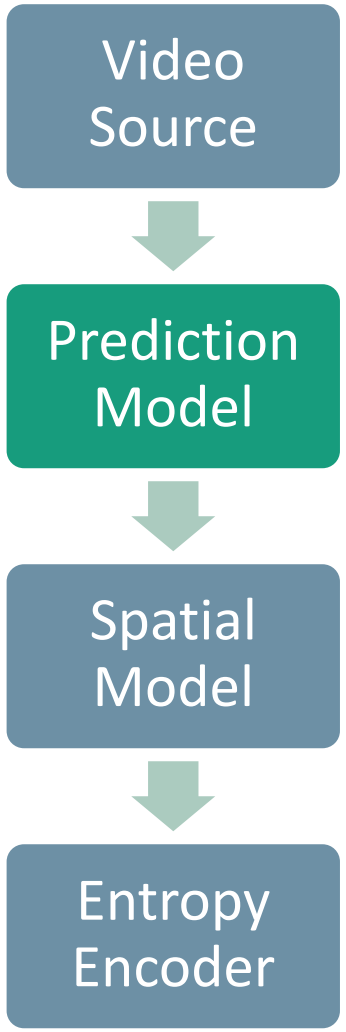
- Removes statistical redundancy
- Outputs a compressed bitstream

We are only providing a high-level overview on the next slides.

The Prediction Model

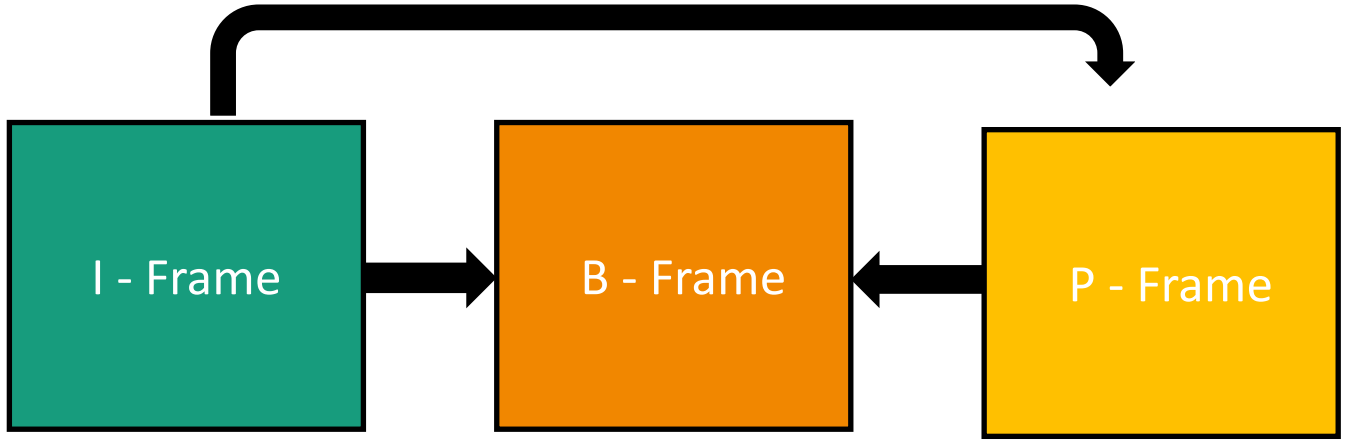
- **Goal**: Reduce redundancy by forming a prediction of the data and subtracting this from the current data
- Divide a frame into macroblocks
- Prediction can be formed from previously coded frames (**temporal prediction**) or image samples in the same frame (**spatial prediction**)

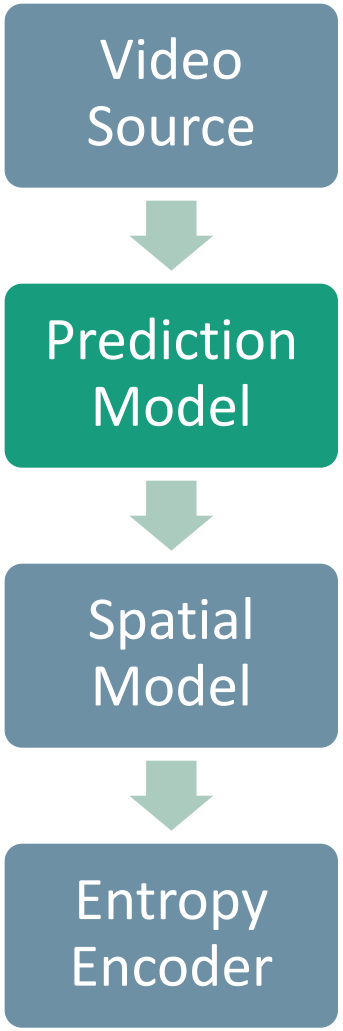




Frame Types

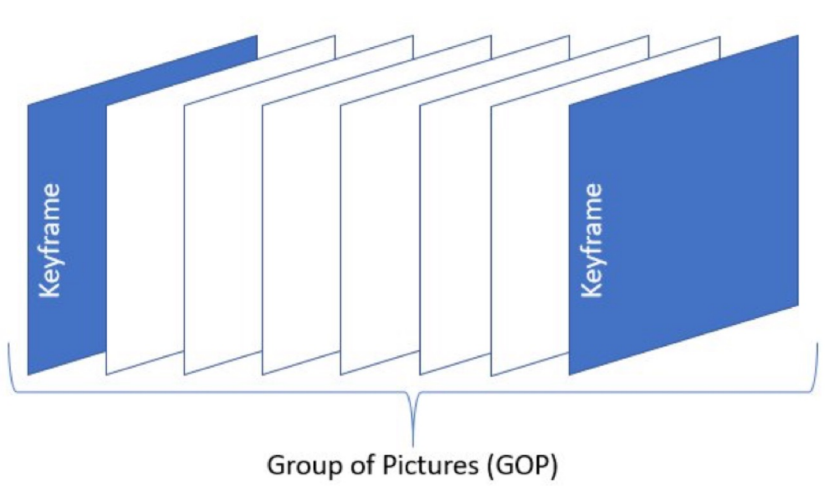
- **I-Frame** (Intraframe, Keyframe) : Self-contained frame. It doesn't rely on anything to be rendered; an I-frame looks like a static photo.
- **P-Frame** (Predictive): Rendered using information of a previous frame.
- **B-Frame** (Bi-Predictive): Rendered using information of a previous frame and a future frame

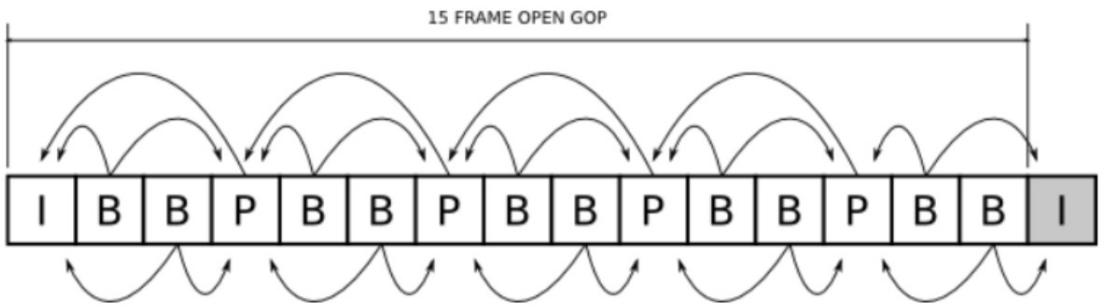


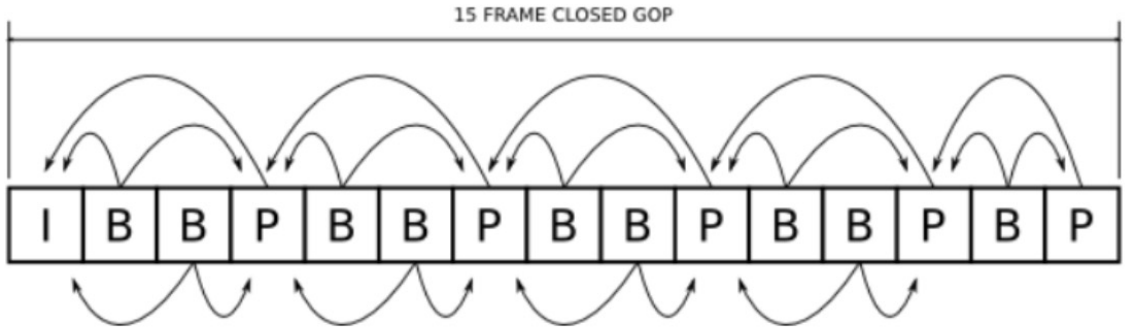


Group of Pictures (GoP)

- **Group of Pictures (GoP),** structure often referred to as M, N.
- **M**: distance between two I or P frames
- **N**: Distance between two I frames (GoP size)
- Open GoP: frames can refer to frames in other GoPs
- Closed GoP: frames can only refer to frames within the GoP.
  - This is used for ABR streaming.







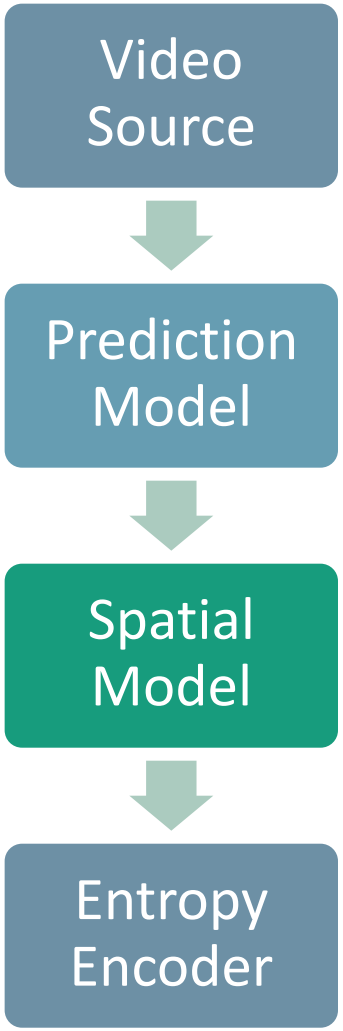
Transform Coding

- Convert image or motion-compensated residual data into another domain, the transform domain
- The output data is arranged in decreasing order of importance
- Example:
  - **Discrete Cosine Transform (DCT)**
  - the top-left corner of the output 2D DCT is called the DC coefficient. It is the most important output of the DCT and contains a lot of information about the original image.
  - the rest of the coefficients contain the image's finer details

	1	2	3	4	5	6	7	8
1	255	255	255	255	255	255	255	255
2	255	255	255	255	255	255	255	255
3	255	255	255	255	255	255	255	255
4	255	255	255	255	255	255	255	255
5	255	255	255	255	255	255	255	255
6	255	255	255	255	255	255	255	255
7	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255

DCT

	1	2	3	4	5	6	7	8
1	2040	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

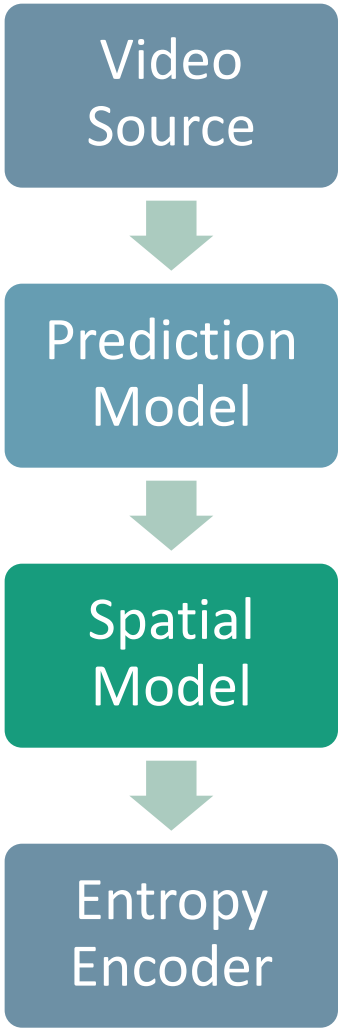


Quantization

- Quantization is used to reduce the precision of image data after applying a transform such as DCT
- Maps a signal with a range of values X to a quantized signal with a reduced range of values Y
- Example: JPEG matrix: https://github.com/google/guetzli/blob/master/guetzli/jpeg_data.h#L48

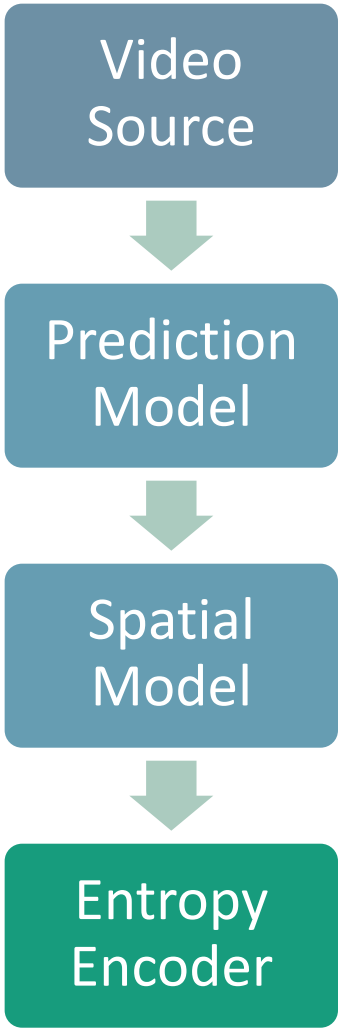
	Y			
X	QP = 1	QP = 2	QP = 3	QP = 5
-4	-4	-4	-3	-5
-1	-1	0	0	0
0	0	0	0	0
2	2	2	3	0
5	5	4	6	5
10	10	10	9	10

- Example: Scalar quantization: Rounding to the nearest integer



Entropy Coding

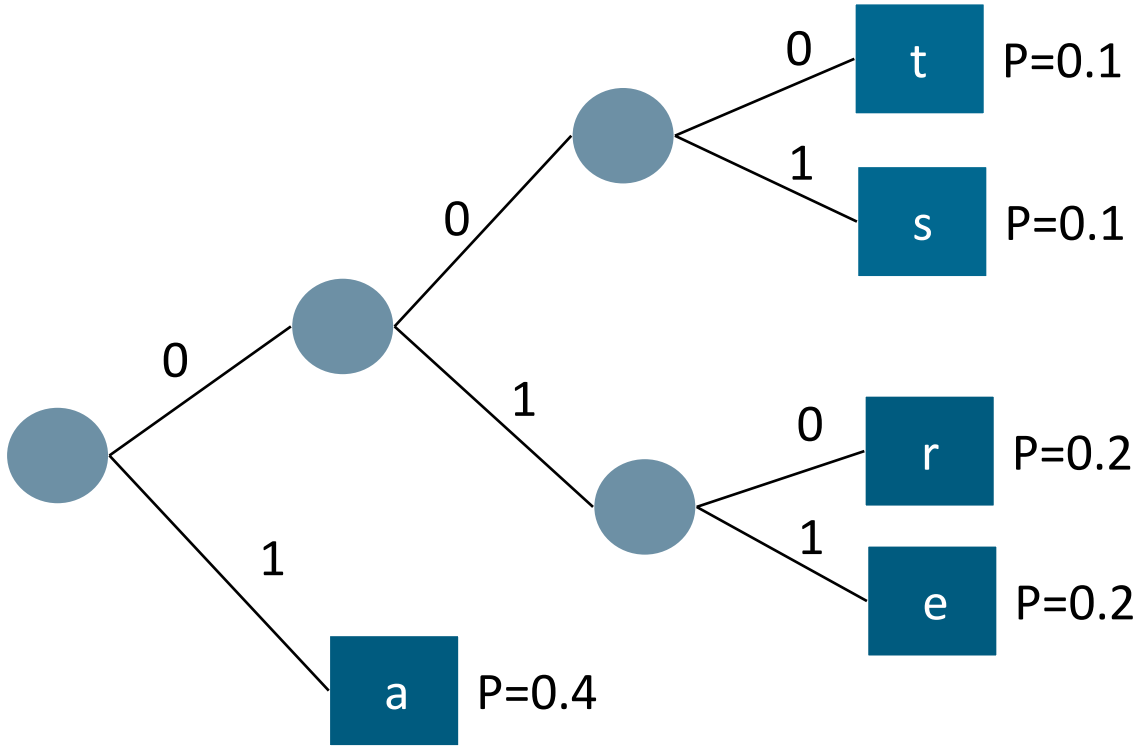
- The entropy encoder converts a series of symbols representing elements of the video sequence into a compressed bitstream
- Applies compression in a “**lossless way**”, similar to zipping a file
- **Variable length coding (VLC)**: Maps input symbols to variable length code (VLCs)
  - Huffman coding
  - CAVLC: Context adaptive variable-length coding: Codes are chosen depending on the statistics of recently-coded coefficients
- **Arithmetic coding (AC)**
  - CABAC: Context adaptive binary arithmetic coding: Method of AC in which the probability models are updated based on previous coding statistics
  - Transmit sequence of data symbols via fractional number
- Arithmetic coding can out-perform Huffman coding



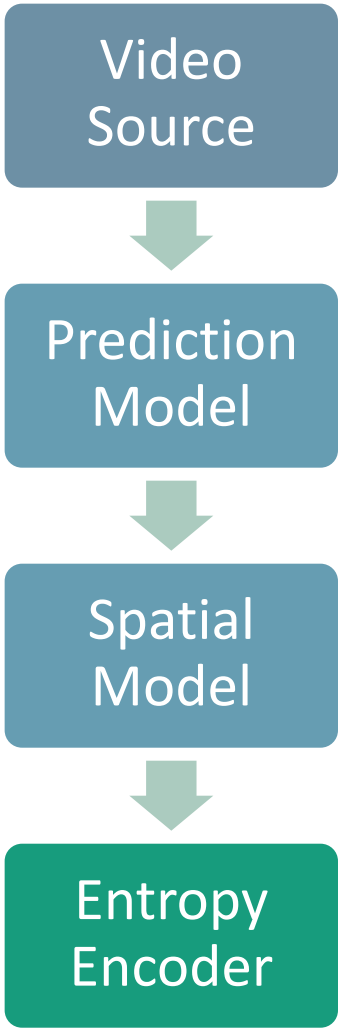
Example - Huffman Coding

Symbol	Probability
a	0.4
e	0.2
r	0.2
t	0.1
s	0.1

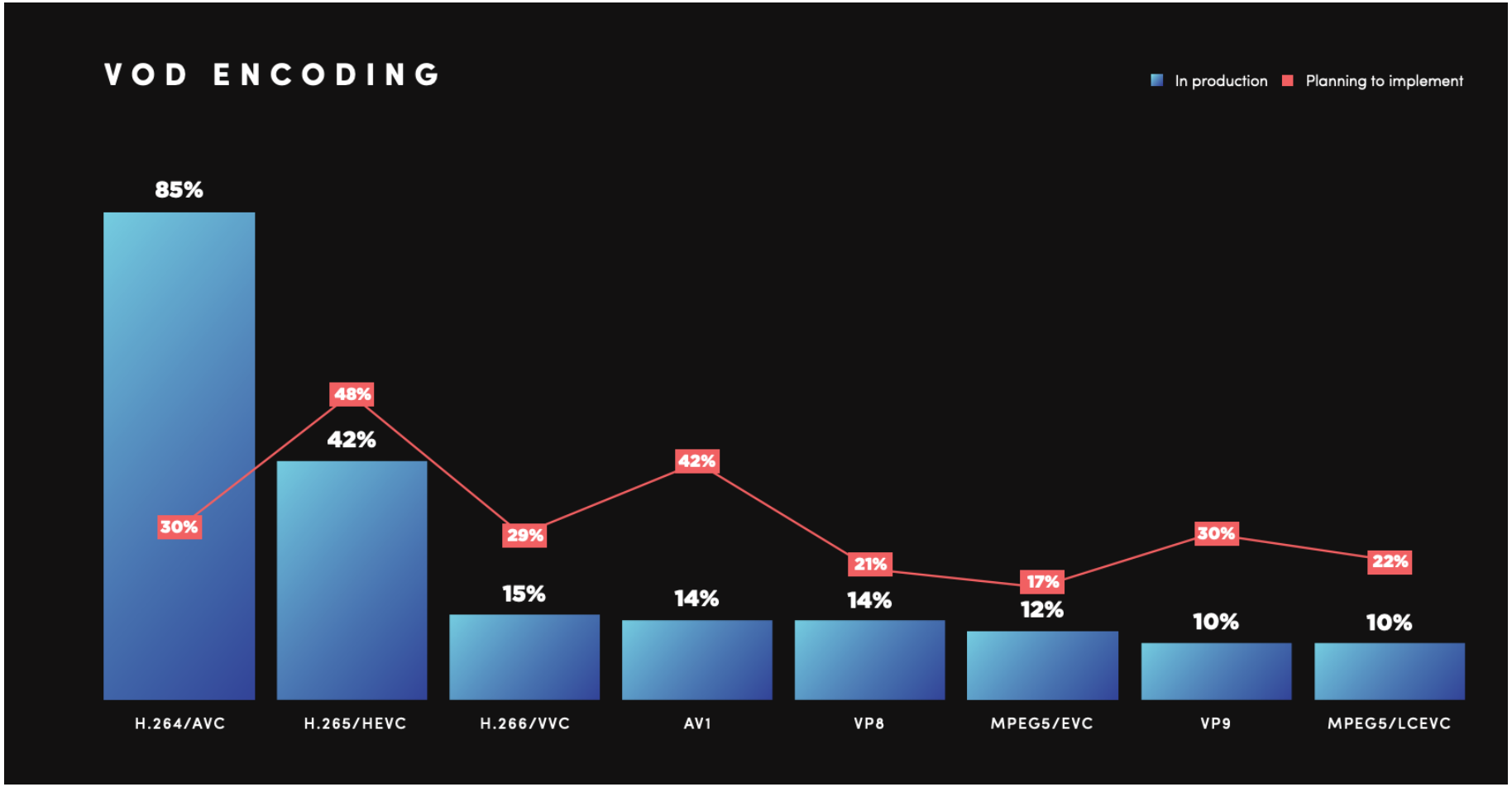
1. Order the list of data in increasing order of probability
2. Combine the two lowest probability items into a node
3. Re-order remaining items and repeat step 2
4. Finish when there is a single root node that contains all other nodes



- “eat”: 0111000
- “tea”: 0000111



Which video codecs are used today?



Video Quality Assessment

Mean Opinion Score (MOS) – Subjective rating

- Subjective rating of video quality: Viewer sits in front of the screen and provides a rating.
- ITU-T has defined several ways of referring to a MOS in Recommendation P.800.1
- MOS : The encoded videos are rated without having a reference to the source
- DMOS: Difference between “reference” and “processed” Mean Opinion Score

Mean Opinion Score (MOS)	Description
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

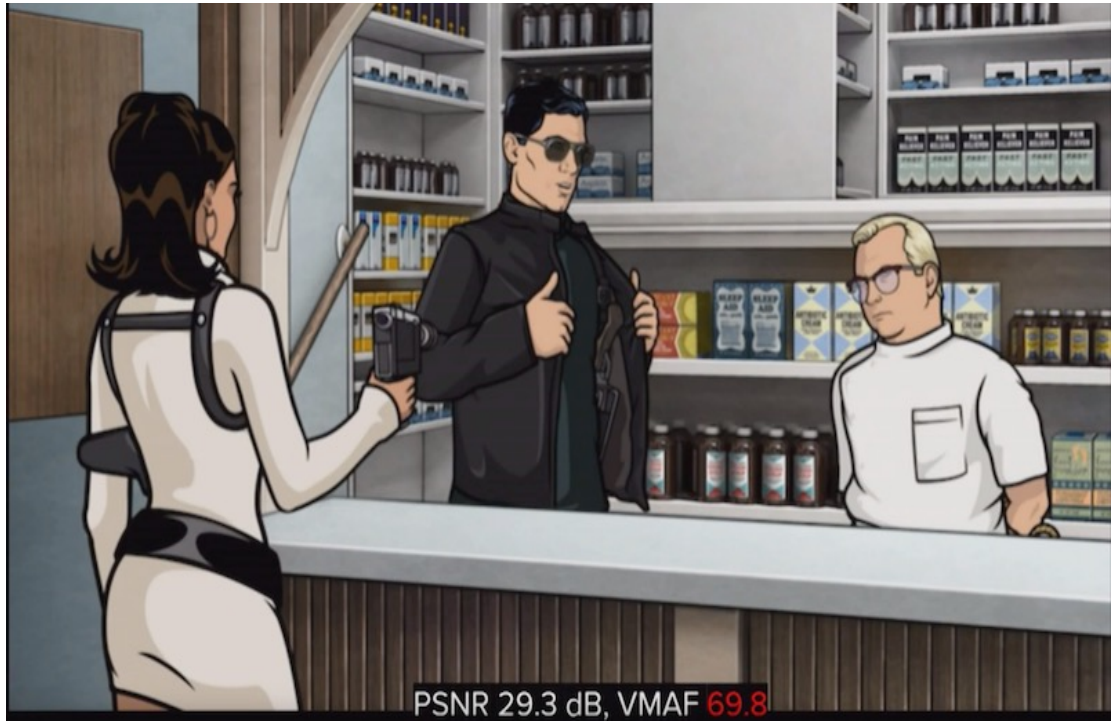
Differential Mean Opinion Score (DMOS)	Description
0 – 0.6	Very Satisfied
0.7 – 1-0	Most Users Satisfied
1.1 – 2.0	Some Users Satisfied
2.1-3.0	Many Users Dissatisfied
3.1-4.0	Most Users Dissatisfied

Objective quality metrics – VMAF and PSNR

- Subjective video testing does not scale. Automated quality assessment is required.
- PSNR and VMAF are full reference metrics. They require the source **and** the encoded video.
- **Peak Signal to Noise Ratio (PSNR)**
  - Based on Mean Squared Error, should be between 35db to 45db
    Low computation time but no direct correlation to the subjective/perceived quality of a video
- **Video Multi-Method Assessment Fusion (VMAF)**
  - Predicts subjective quality by combining multiple elementary metrics
  - Underlying ML Model is trained and tested on „real“ subjective quality measurements
  - Different models for different viewing conditions (mobile vs. 1080 vs. 4k)
  - A VMAF value of 93 is indistinguishable from original or with noticeable but not annoying distortion.

Video Quality Assessment

VMAF vs. PSNR



MOS in relation to video quality metrics

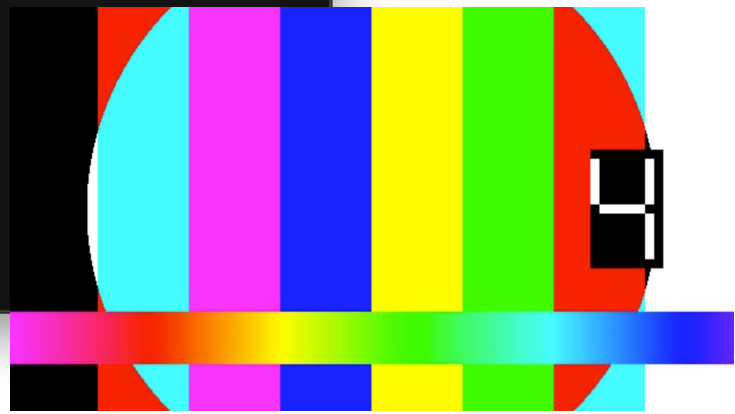
MOS Value	VMAF Value	PSNR value in dB
4 - 5	80-100	35 - 45
3 - 4	60 -80	30 - 35
2 - 3	40 - 60	25 - 30
1 - 2	0 - 40	15 - 25

Just Noticeable Difference (JND):

- The amount (e.g. number of VMAF points) the quality of the video has to change, for there to be a noticeable difference to the viewer
- For VMAF: JND between 2 and 6 points

PSNR calculation

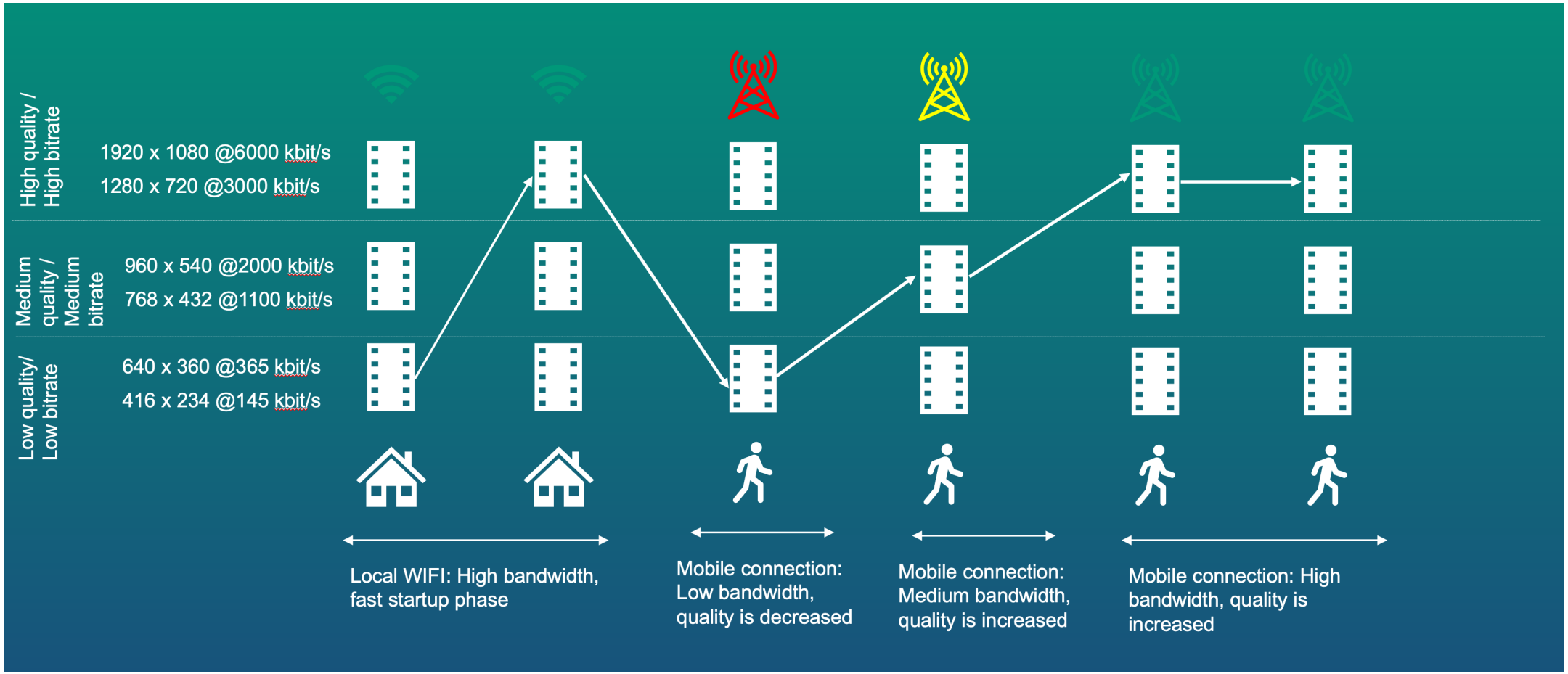
```
> ffmpeg -i ../hands_on_2/crf-23.mp4 -i test.mp4 -filter_complex "psnr" -f null /dev/null

[Parsed_psnr_0 @ 0x7fd55df0f2c0] PSNR y:55.767187 u:53.161004 v:54.088446 average:54.209598 min:51.748051 max:57.578735

> ffmpeg -i ../hands_on_2/crf-51.mp4 -i test.mp4 -filter_complex "psnr" -f null /dev/null

[Parsed_psnr_0 @ 0x7fc248e06500] PSNR y:32.907429 u:36.745032 v:35.442697 average:34.731624 min:33.441237 max:36.154966
```



- The two test videos were encoded with different CRF values. A higher CRF value results in larger quantization values and a lower bitrate/quality

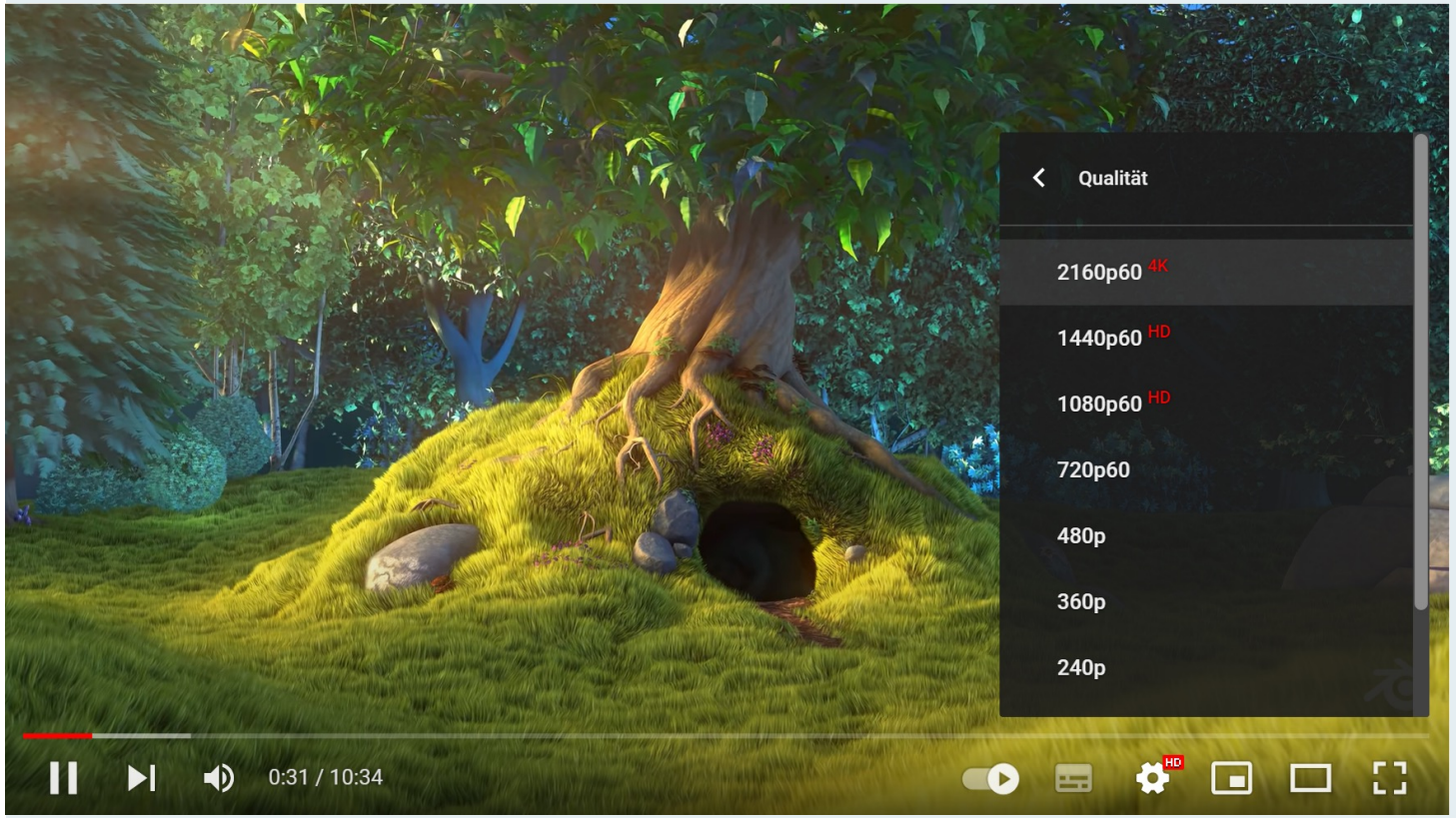
Content Aware Encoding

Background: Adaptive Bitrate Streaming



Background: Video Representations

Example for adaptive bitrate streaming on YouTube



Adaptive Streaming im Youtube Player; Videoquelle: Big Buck Bunny 60fps 4K - Official Blender Foundation Short Film: https://www.youtube.com/watch?v=aqz-KE-bpKQ

Standard Encoding Ladder (Netflix)

#	Bitrate (kbit/s)	Resolution
1	235	320x240
2	375	384x288
3	560	512x384
4	750	640x480
5	1750	720x480
6	2350	1280x720
7	3000	1280x720
8	4300	1920x1080
9	5800	1920x1080

Content Aware Basics - Motivation

Low complexity

High redundancy

Animation

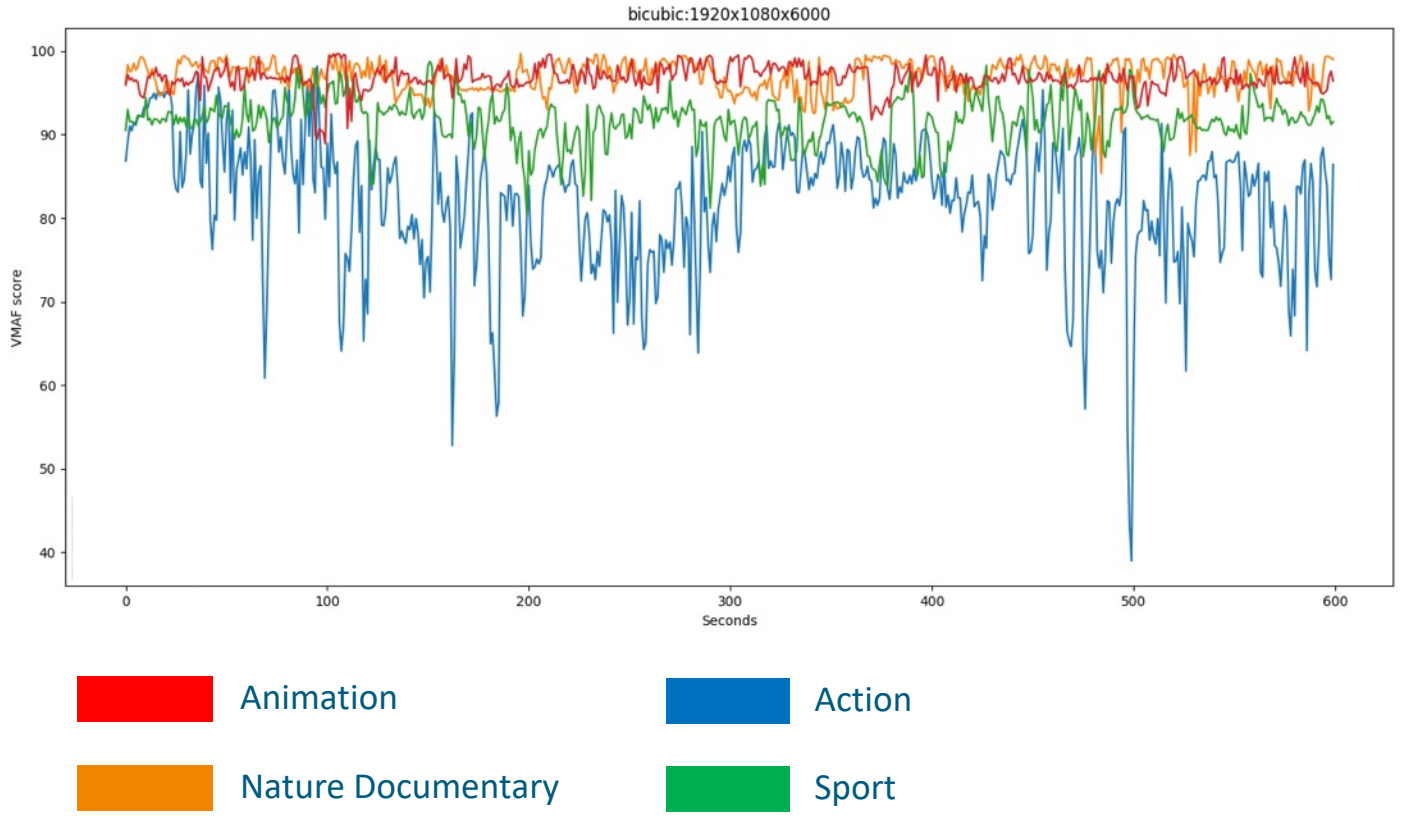
Nature Documentary

High complexity

Low redundancy

Action

Sport



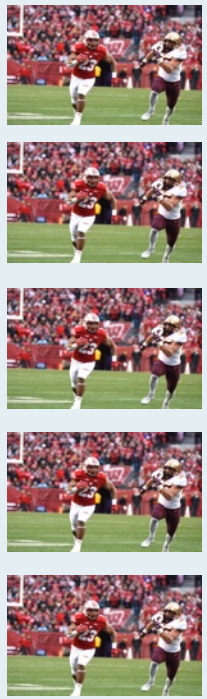
➢ **Different types of content require different encoding settings to achieve same quality**

A Solution: Per-Title Encoding

Time consuming and computationally heavy!

Source Video

240p
200kb/s

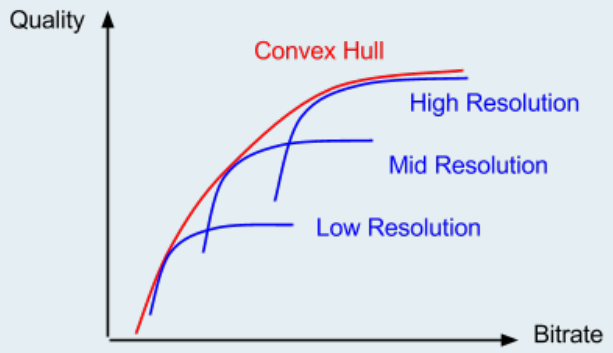


1080p
8000kb/s

Create Test Encodes
with different encoding settings

Low Quality

High Quality



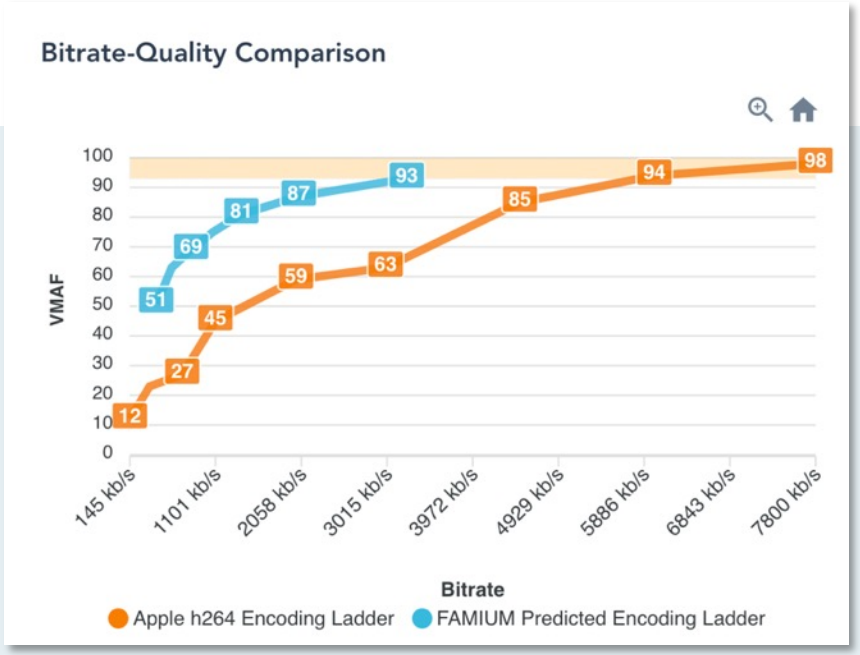
Calculate Quality Metrics
e.g. VMAF, SSIM

Resolution	Bitrate
540p	600
720p	1200
1080p	2300

Choose optimal settings
e.g. Bitrate & Resolution pairs

DeepEncode – AI-powered Content Aware Encoding

- **DeepEncode**: Content-aware (Per-Title & Per Scene) encoding solution for live and VoD video streaming, leveraging artificial intelligence and computer vision methods
- No testencodes needed! – DeepEncode uses trained ML Models to predict optimized encoding settings for any given video, based on the complexity of the video
- **Benefits**: bitrate and storage savings of up to **47%** while maintaining at least the same video quality



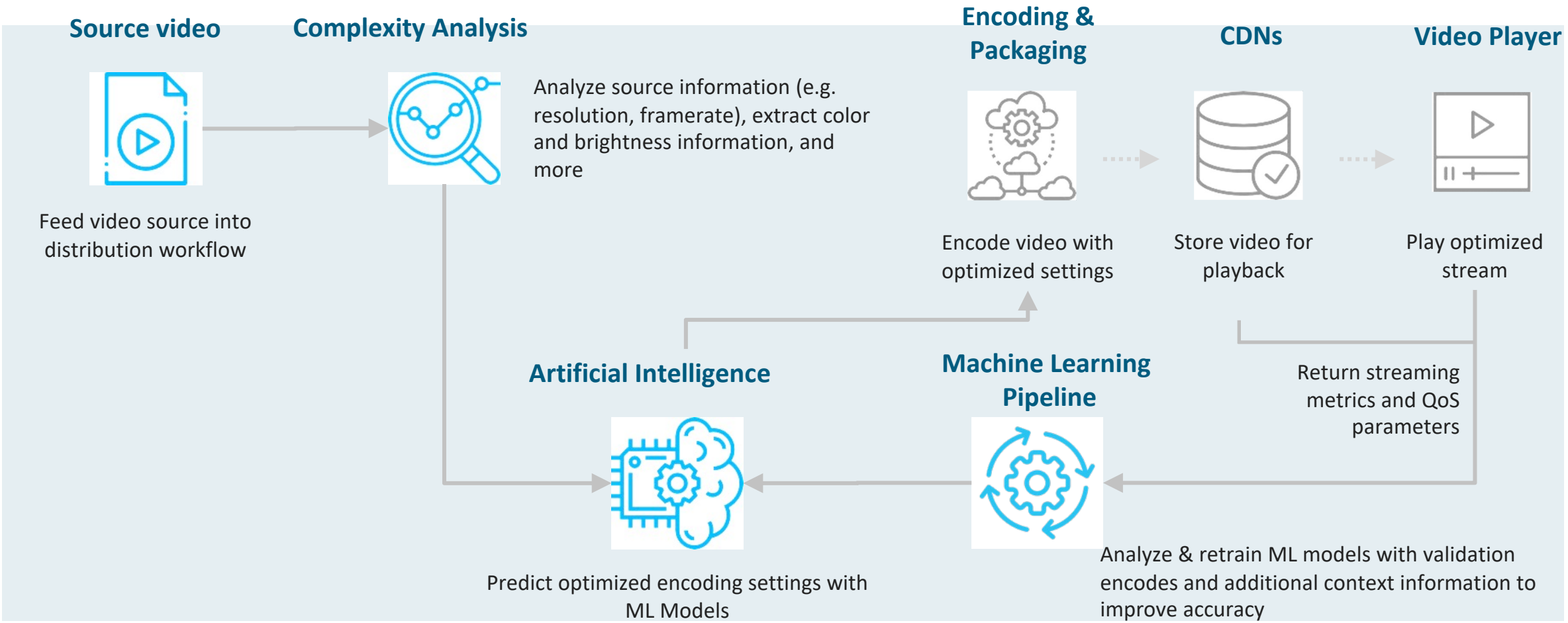
Comparison

Bitrate Ladder	Average Bitrate (kb/s)		Average VMAF		Storage Size (estimated)	
Default Encoding	1050		58		22.30 MB	
DeepEncode	-390 kb/s (-37%)	660	+4 (+7%)	62	-8.27 MB (-37%)	14.03 MB

⚠ Reference ladder contains resolutions greater than those of the source video. These representations are not used for comparison calculations.

Fraunhofer FOKUS

DeepEncode – Per-Title Workflow

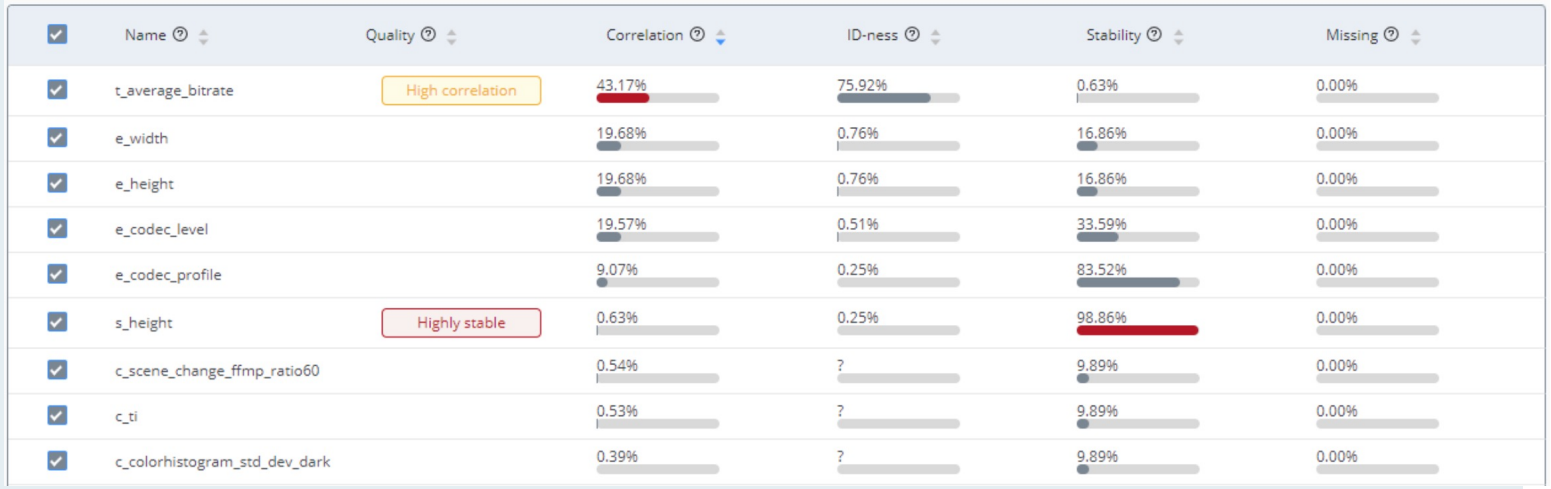


ML Background: Model development

- **About 800 raw original videos** as training, test and validation data set (free available content)
- Every video was encoded about **60 times** (6-9 different resolutions, bitrates, etc)

  → **about 48.000 test-encodes for first models**

- Up to **64 significant features** after preprocessing data with DeepEncode
  - Video-Metadata
  - Encoding-settings (Bitrates, Codec..)
  - Complexity Analysis
  - Quality metrics



Name	Quality	Correlation	ID-ness	Stability	Missing
t_average_bitrate	High correlation	43.17%	75.92%	0.63%	0.00%
e_width		19.68%	0.76%	16.86%	0.00%
e_height		19.68%	0.76%	16.86%	0.00%
e_codec_level		19.57%	0.51%	33.59%	0.00%
e_codec_profile		9.07%	0.25%	83.52%	0.00%
s_height	Highly stable	0.63%	0.25%	98.86%	0.00%
c_scene_change_ffmp_ratio60		0.54%	?	9.89%	0.00%
c_ti		0.53%	?	9.89%	0.00%
c_colorhistogram_std_dev_dark		0.39%	?	9.89%	0.00%

ML Background: Model development

- Baseline: **statistical model**

$$vmaf(bitrate) = \log_2(bitrate) * 7$$

→ **RMSE von 11,74** (error value; maps to VMAF-scale 0-100)

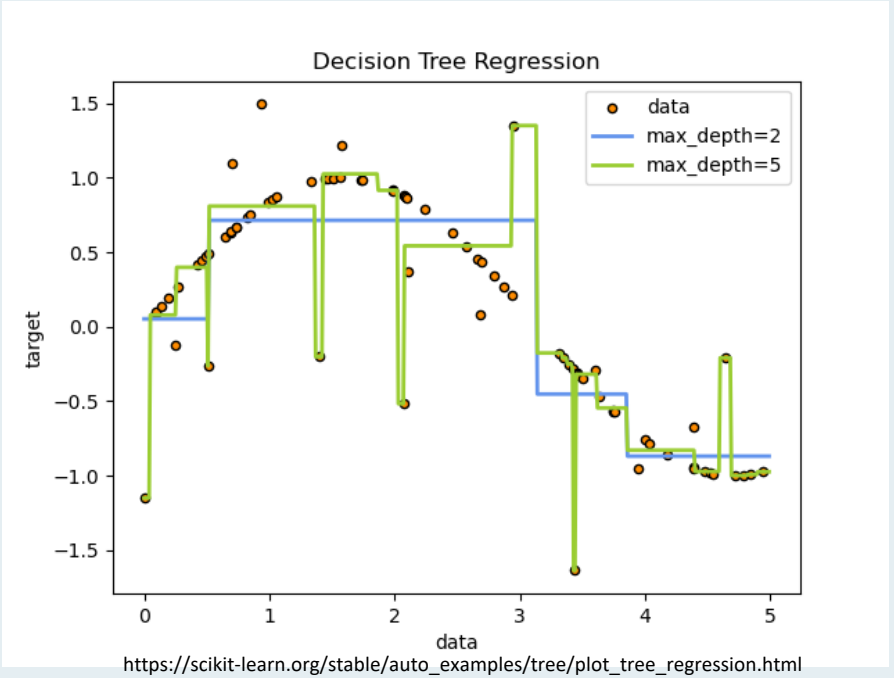
- **RMSE von 1,43 via Boosted Decision Tree Regression**

→ **But: very slow feature processing** (50% of video playtime)

- After **Feature-reduction**

→ **RMSE 3,22**

→Faster video processing (only ca. 10% of video playtime)



https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Silhavy, Daniel; Krauss, Christopher; Chen, Anita; Nguyen, Anh-Tu; Müller, Christoph; Arbanowski, Stefan; Steglich, Stephan; Bassbouss, Louay. Machine Learning for Per-Title Encoding.
The Proceedings of the 2022 NAB Broadcast Engineering and Information Technology (BEIT) Conference, Las Vegas, April 2022.

ML Model Evaluation & Results

- **Lab-Tests with 80 previously unknown videos (test set with 10-fold cross-validation)**

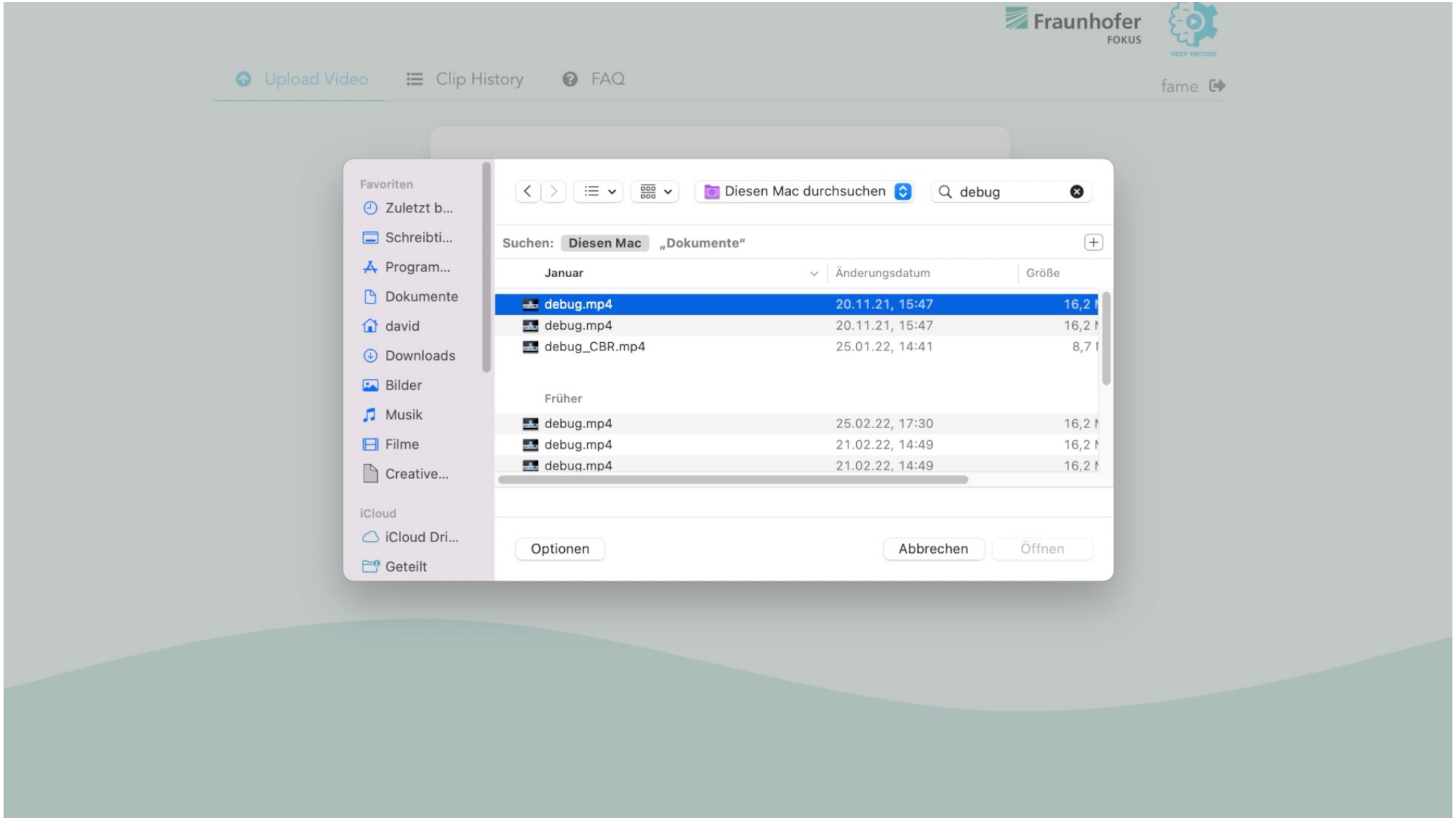
	Storage in MB	Avg. Bitrate in Kbit/s	Avg. VMAF
Standard Encoding	816.2	2136	84.1
Deep Encode Modell	346.5	1360	86.4
Difference	-469.7 | **57.6%**	-776 | **36,3%**	+2.3 | **2,7%**

arte

Saves costs in bitrate and storage while maintaining quality!

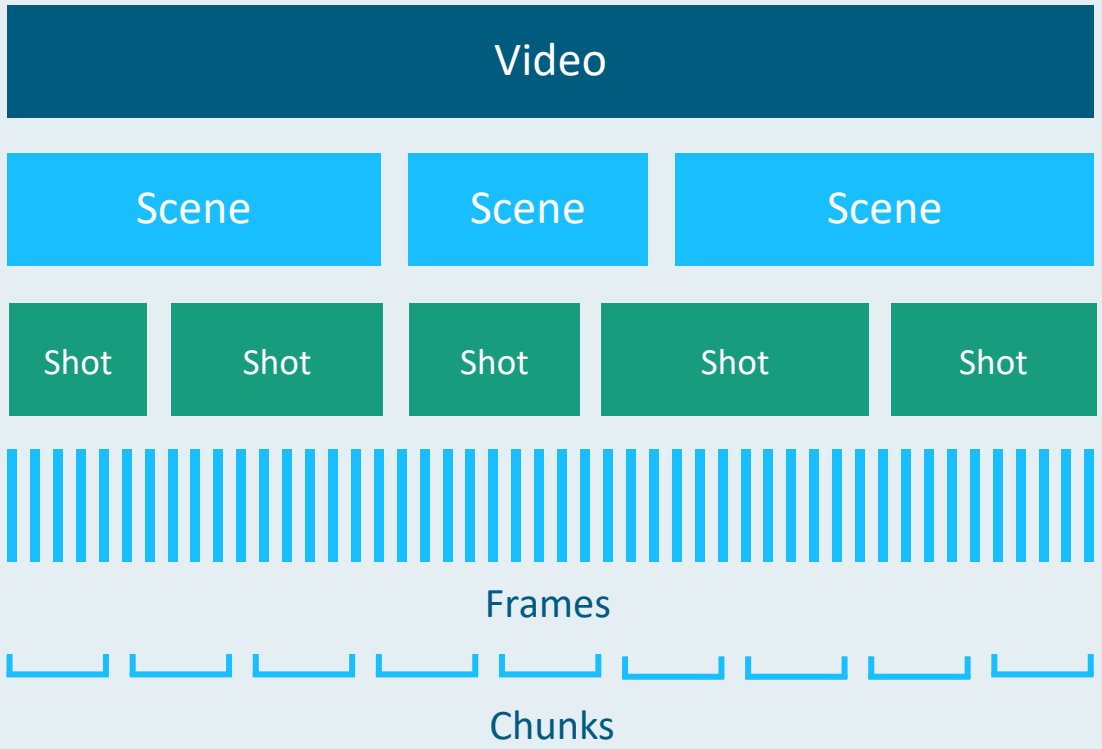
- **Next Steps:**
  - **Context-aware** adaptations
  - Reducing computation times
  - **Live-Streaming**

DEMO – DeepEncode - Per-Title Encoding



Fraunhofer FOKUS

Content Aware – Many Roads lead to Rome

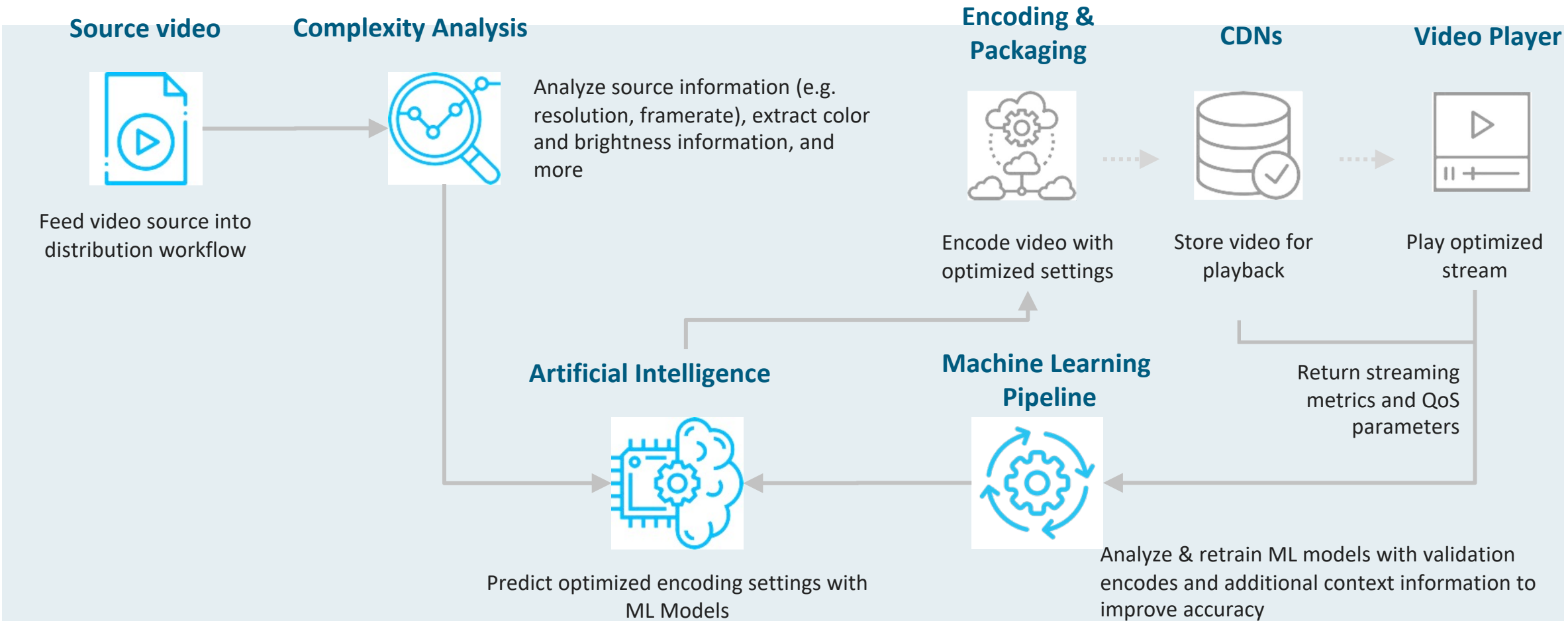


Per-Title Encoding

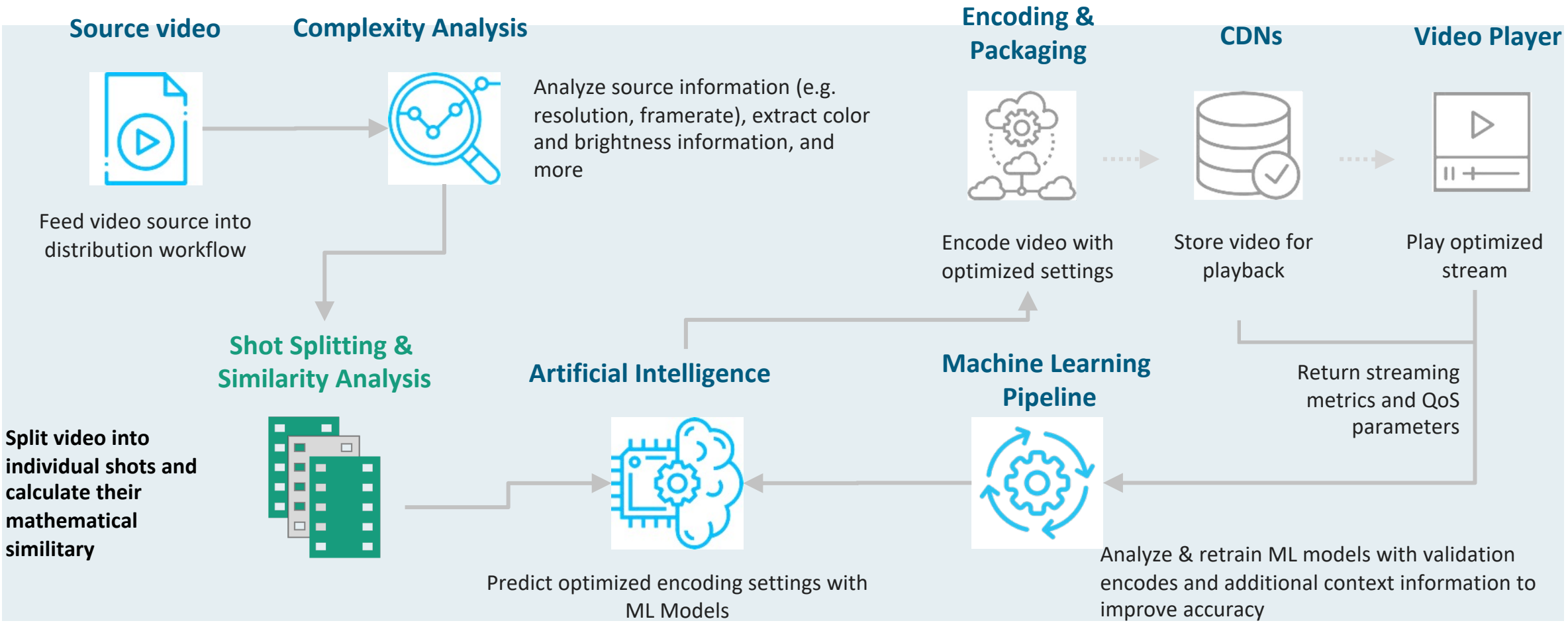
Per-Scene Encoding

Shot-Based Encoding

DeepEncode – Per-Title Workflow

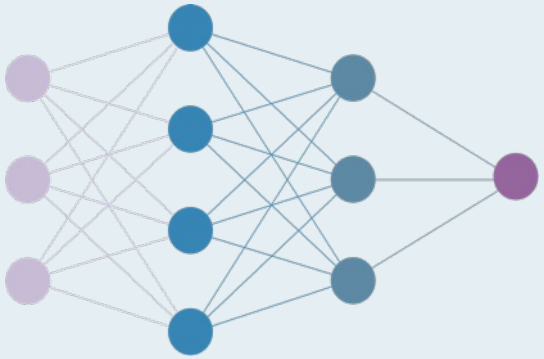


DeepEncode – Per-Shot & Similarity Workflow



Similarity Analysis & Shot-Encoding Workflow

Deep Neural Network

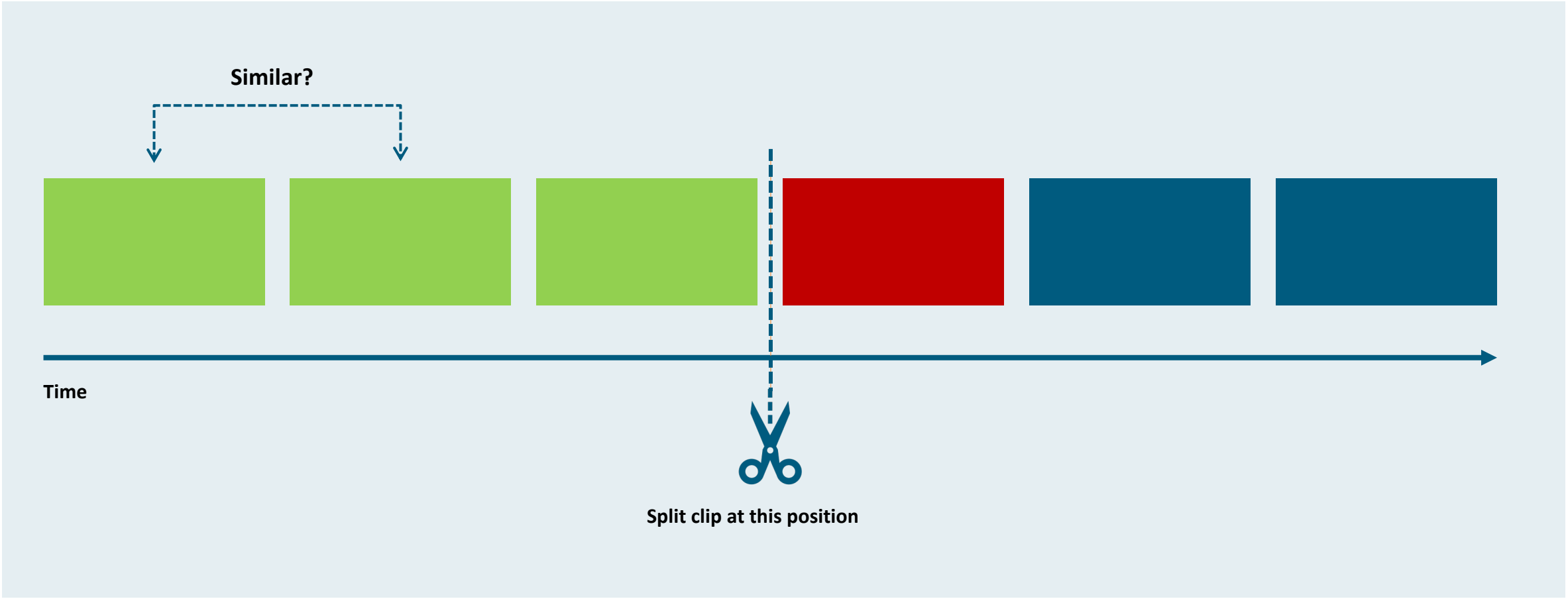


Are the Frames similar?

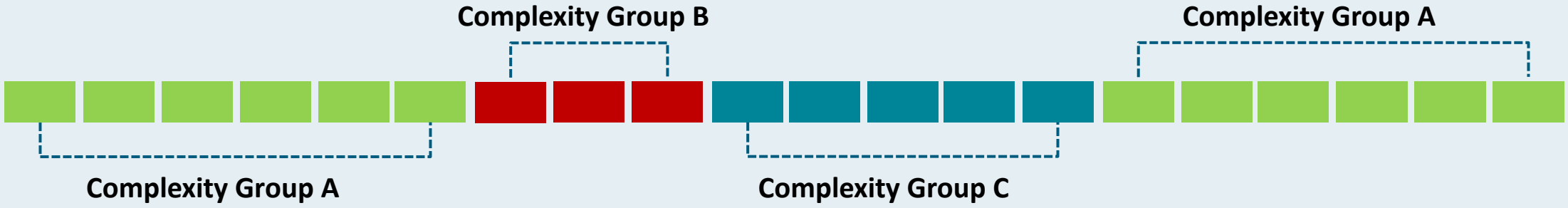
YES → Move on to the next frames

NO → Different complexity detected!
→ Split clip at this position

Similarity Analysis & Shot-Encoding Workflow



Similarity Analysis & Shot-Encoding Workflow



Infer optimized encoding settings

By checking Database for existing, similar shots or using regression (ML) models

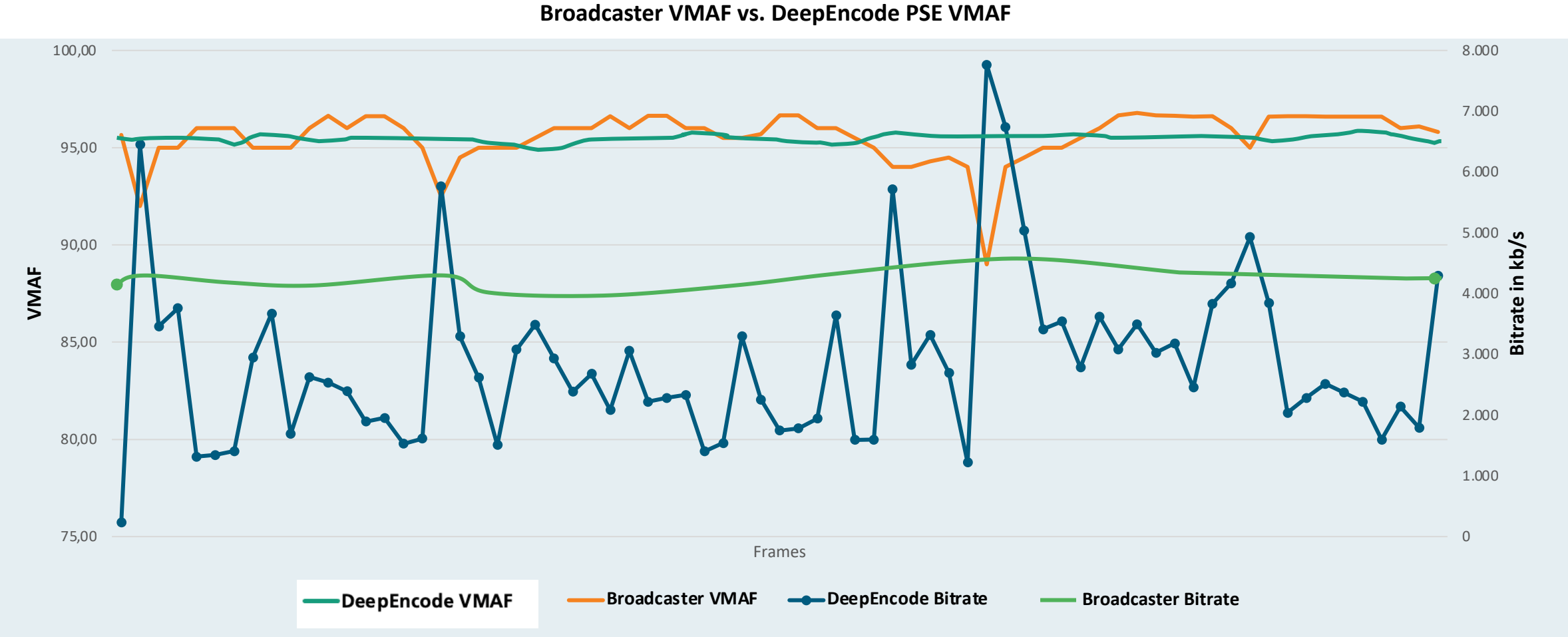
Encode Complexity Groups

And stitch them back together or send to packaging

Shot-based Encoding - VMAF Comparison

Average Bitrates:

Broadcaster: **3880 kb/s**
DeepEncode PSE: **2730 kb/s**



Fraunhofer FOKUS

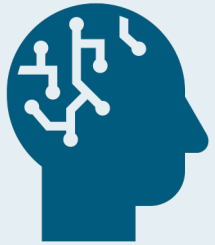
Shot-based Encoding - Results

1/60 of video playtime

to conduct Frame-by-Frames similarity analysis

Up to 68% bitrate reduction

compared to current broadcaster encoding settings



Prediction accurate to ±2 VMAF points

With more training content models become more accurate

Summary

Summary

Video Foundations

- Framerates
- Resolutions & Aspect Ratios
- Scan modes: Progressive vs Interlaced
- Color modes and Chroma Subsampling

Video Encoding

- Video Codecs – what & why
- Video Encoding – How?
- Frame Types
- Group of Pictures

Video Quality

- Subjective / objective testing
- MOS
- PSNR and VMAF

Content Aware Encoding

- Per-Title Encoding
- Per-Shot Encoding

Contact

Daniel Silhavy

- Email: daniel.silhavy@fokus.fraunhofer.de
- LinkedIn: https://www.linkedin.com/in/daniel-silhavy-21650a129/

Fraunhofer FOKUS
Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
info@fokus.fraunhofer.de
www.fokus.fraunhofer.de