Using Video Compression to Exploit Similarities in Biometric Databases

Panteleimon Cheropoulos, Samy Dafir, Kevin Schörgnhofer

Fachbereich Computerwissenschaften - Universität Salzburg

30. Juni 2017

Inhalt

Intro

General incentive of this assignment:

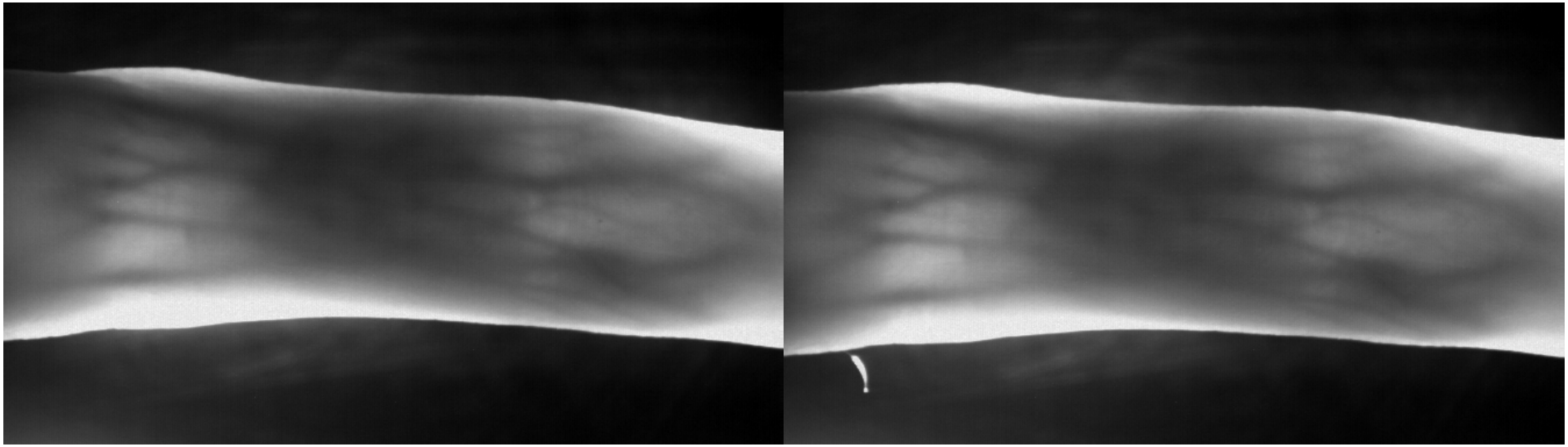
„Is it possible to effectively apply **video compression** for almost identical **pictures**?“

→ can we achieve *better* results with video compression than with image compression?
→ which codec is best suited for our purposes?
→ how does the change of video codec parameters affect the results?
→ how to determine the quality of the results?

Dataset

The database consists of finger vein images of different fingers of different persons

- 6 fingers per person, with 4 pictures per finger $\rightarrow$ 24 pictures per person
- 60 persons at all
- we worked with a subset of those



Video Compression

Why video compression?

Video Compression

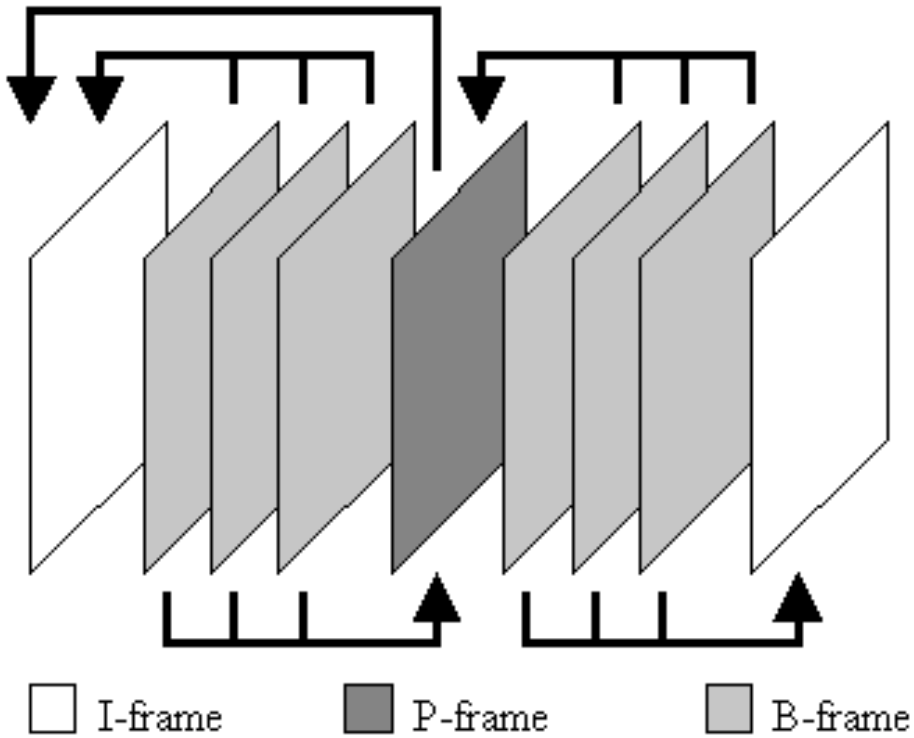
Why video compression?

- Very similar images
- Image compression only compresses individual images
- Video compression does 2 things:
  1. Compresses images
  2. Exploits similarities between images

I,P,B-Frames

3 different types of pictures

- I-Frame: Intra-coded picture
- P-Frame: Predictive-coded picture
- B-Frame: Bidirectional predictive-coded picture



Group Of Pictures

- usually defined with two numbers
  1. defines distance of two I-Frames
  2. defines distance of two anchor frames (I or P)
- we used GOP to adapt the encoding to the database
  - 24 pictures per person: use GOP 24 $\rightarrow$ 1 I-Frame per person
  - 4 pictures per finger: use GOP 4 $\rightarrow$ 1 I-Frame per finger
- P- and B-Frames allow higher compression $\rightarrow$ GOP affects the compression rate

JPEG2000

- used as a baseline for comparison
- standard encoding settings, except number of layers
- ImageMagick (7.0.5-10) with integrated OpenJPEG library
  1. encode pictures with jp2, with different compression rates
  2. determine quality of the pictures
  3. compare with pictures compressed with video codecs

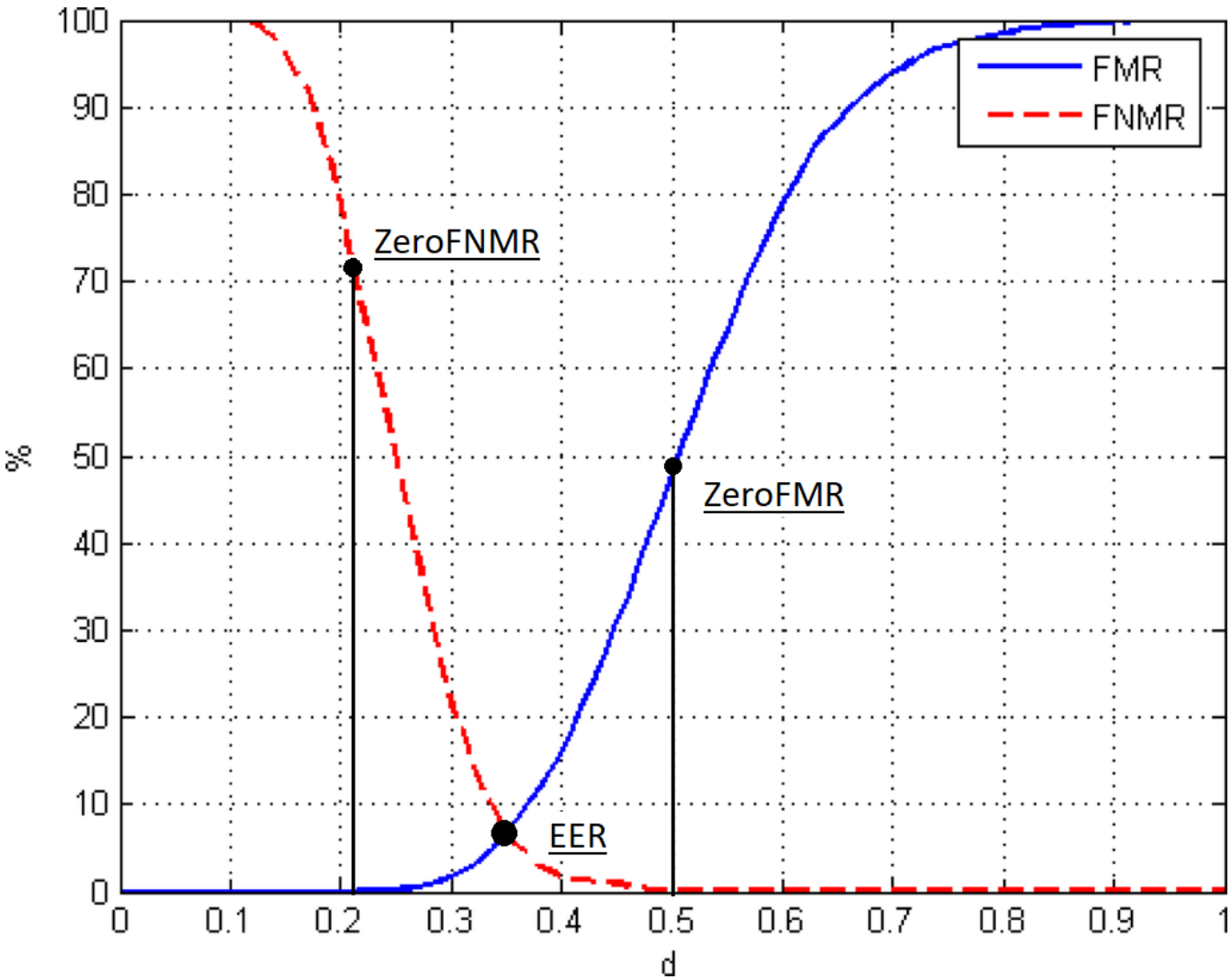
Matcher

- Used as a "black box"
- Compares original and compressed images
  $\rightarrow$ Checks if matches found
- calculates different error metrics
- Target:
  Compare results of jpeg2000 and video compression

Error Metrics

- FMR: False Match Rate
- FNMR: False Non Match Rate
- EER: Equal Error Rate
- Lower values are always better

Error Metrics



http://www.mdpi.com/sensors/sensors-11-09499/article_deploy/html/images/sensors-11-09499f22-1024.png

CRF

- CRF value (Constant Rate Factor)
  1. The range of the quantizer scale is 0-51
  2. A lower value means better quality (0 for best quality, lossless)
  3. default value is 23
  4. A higher value means bad quality (51 for worst quality)

Presets

- presets (they provide a certain encoding speed)
  1. ultrafast , superfast , veryfast , faster , fast
  2. medium (default)
  3. slow, slower, veryslow, placebo
  4. we focused more to the slower presets (medium-veryslow)

Settings

- what are the settings behind them?

	medium	veryslow
–b-adapt	1	2
–partitions	p8x8,b8x8,i8x8,i4x4	all
–bframes	3	8
–ref	3	16

Settings

- quick explanation of the settings :
  - –b-adapt "Mod":
    - algorithm for the adaptive distribution of B-frames
    - values : 0,1,2
  - –bframes "Max":
    - Defines how many B-frames can be positioned directly behind each other
    - values are between 0 and 16 (3 is default)

Settings

- –partitions "partitions":
  - partition size for macroblocks
- –ref "frames":
  - amount of valid reference frames

Settings (qscale mpeg4)

- -qscale:v n
- configure and select a video quality level
- value for n : 1-31
- 1 is the highest quality for largest filesize
- 31 is the lowest quality for smalest filesize

Video Compression

Setup:

- Used ffmpeg v.3.3.2 (latest version)
- Compressed 240 images
- Different crf values (0-50)
- Different qscale values for mpeg4 part 2
- Varying group of pictures (1, 4, 24)
- two presets (medium, veryslow)

Video Compression

Repeat for each (crf, gop, preset) - combination

1. Compress images into single video
2. get videosize (for compression rate)
3. Decompress video $\rightarrow$ get images
4. Put into folder named with settings

Additional steps

- Collect image names $\rightarrow$ parameters for matcher
- rename decompressed videos

Matching

- Used matcher as "black box"
- Input:
  1. original images
  2. compression output folders (crf, gop, preset)
- Match each output folder
- Retrieve error metrics
- Evaluate Error rate dependent on compression rate

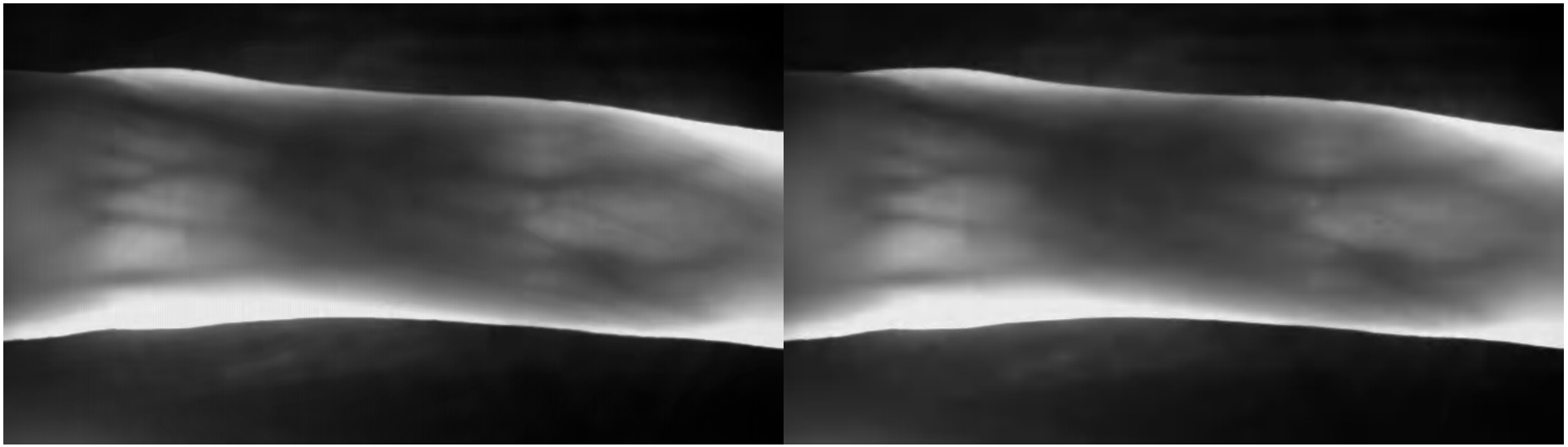
Evaluation

Compression rate vs. matching errors

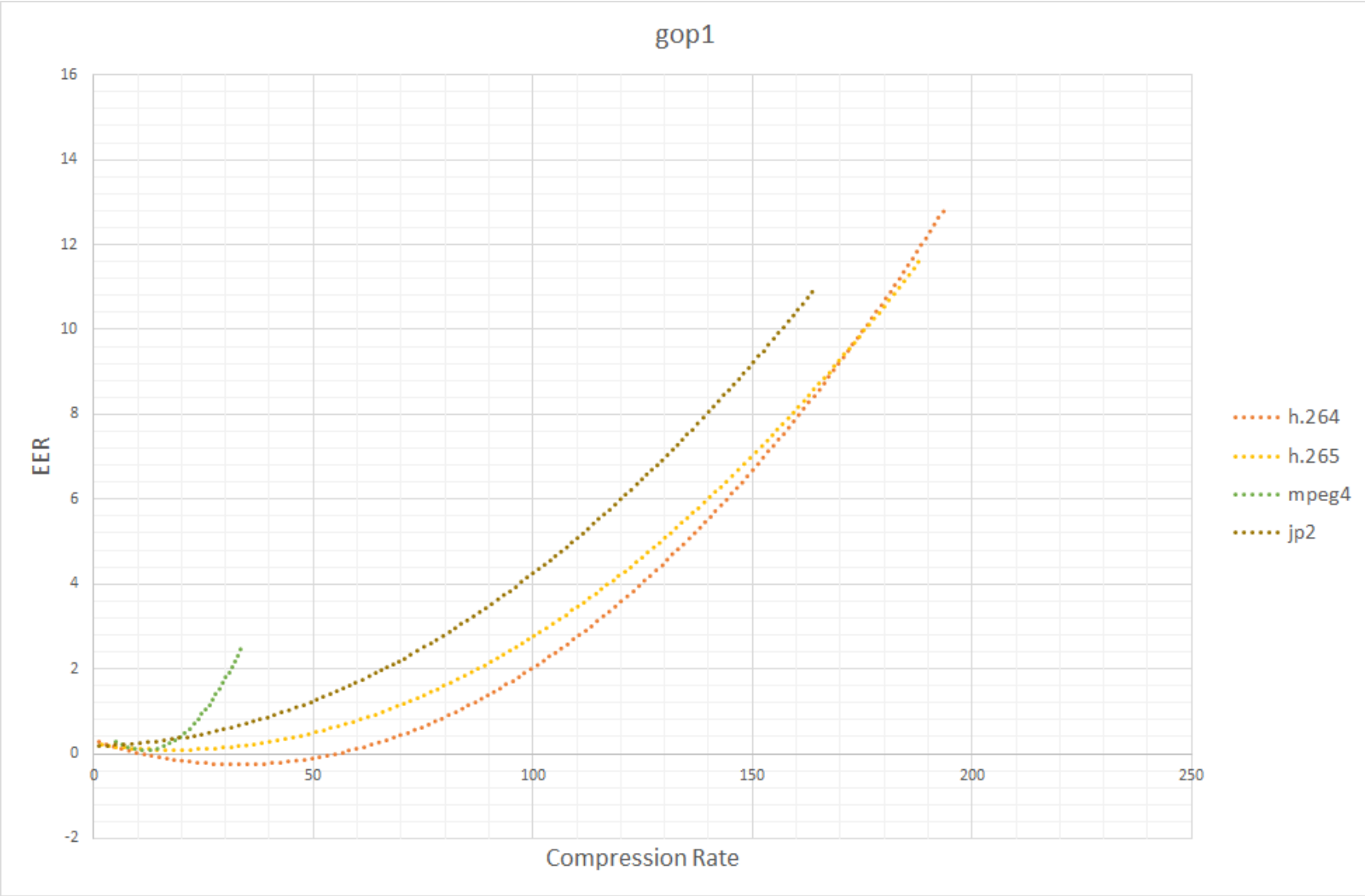
- Group of pictures (1, 4, 24)
- Codecs (mpeg4 part 2, h.264, h.265)
- presets (medium, veryslow)
- Best result of jpeg2000 compression as baseline

Comparing output images

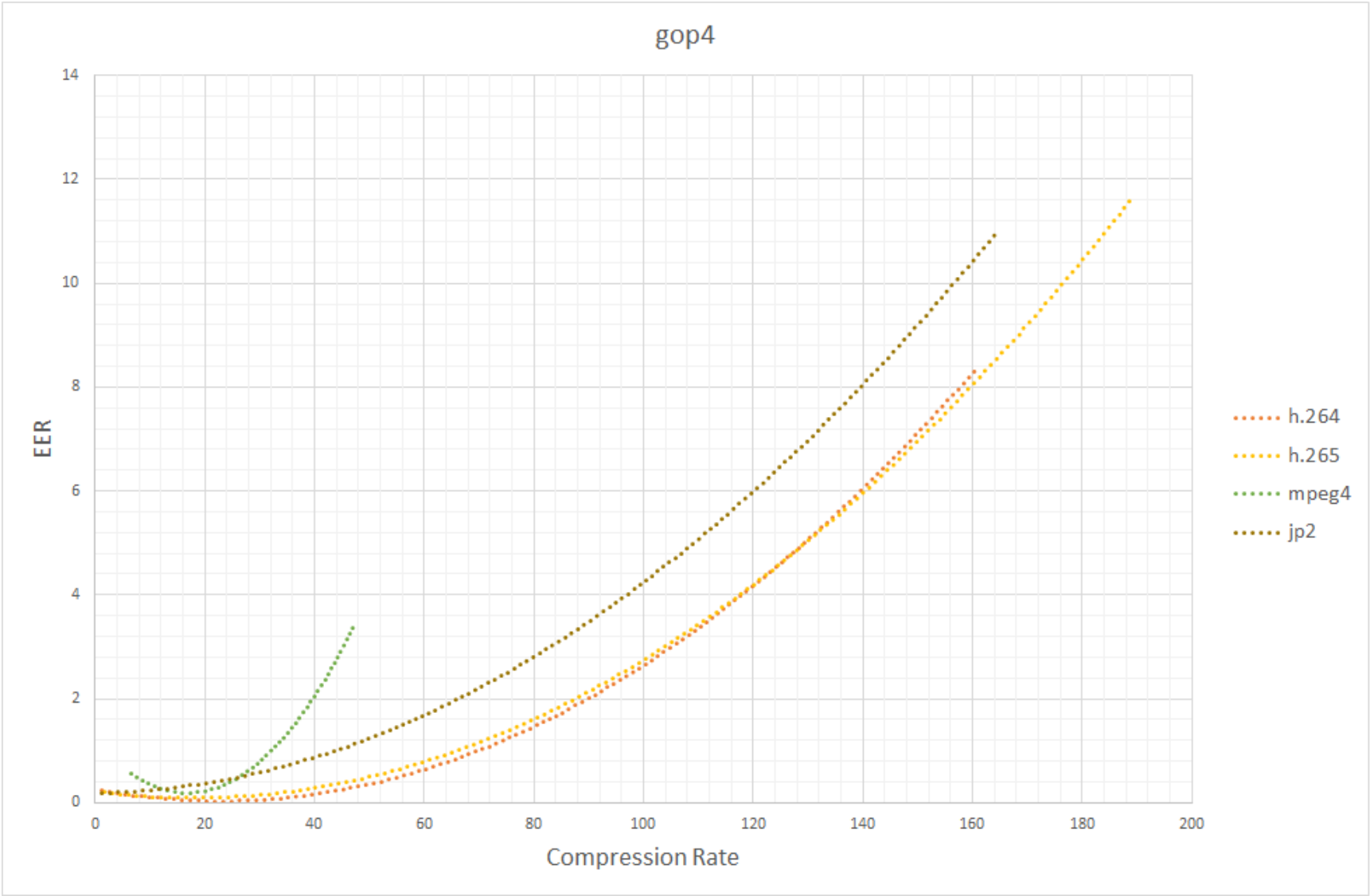
h265 vs. JPEG2000



Comparing codecs

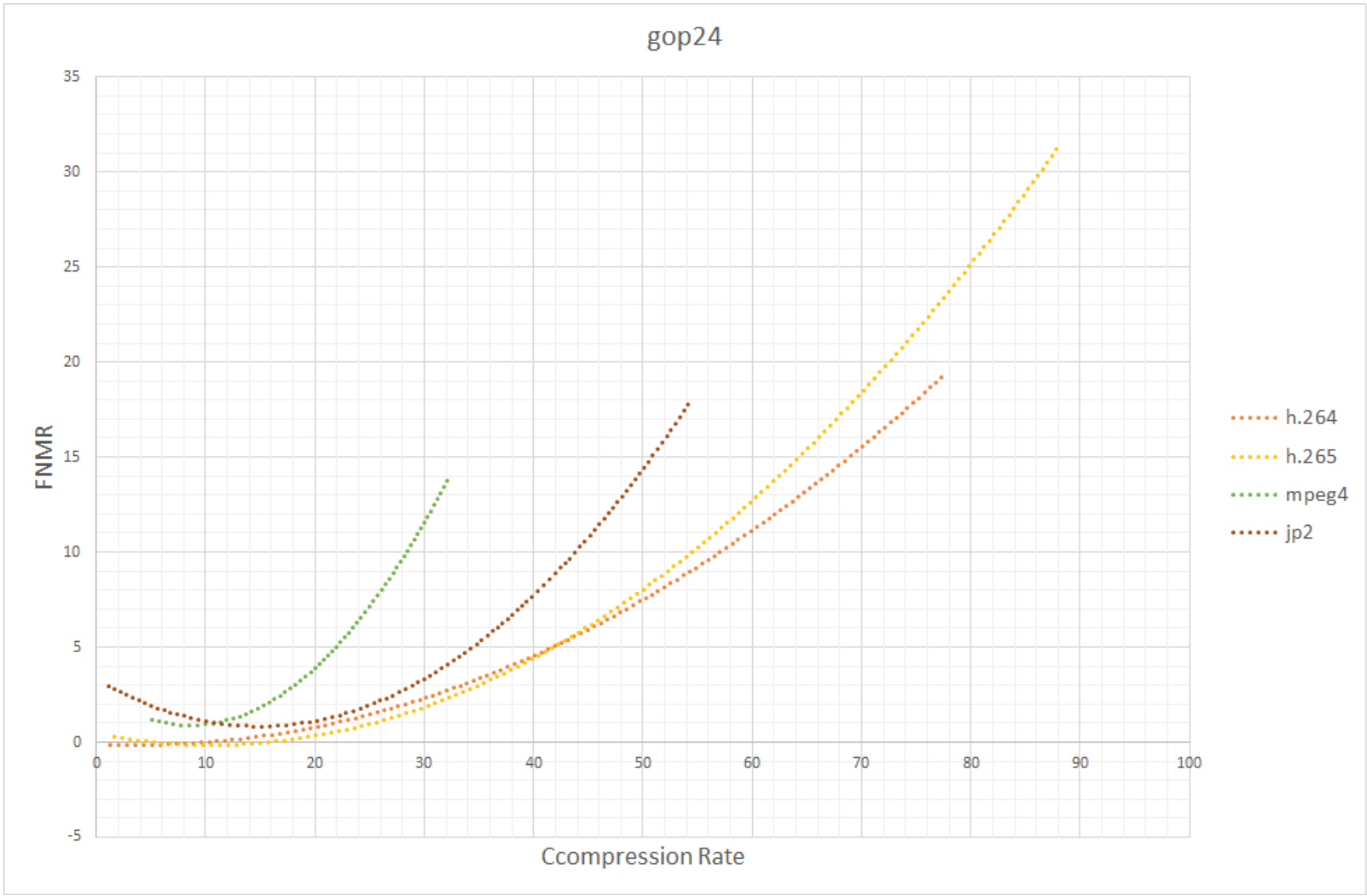


Comparing codecs

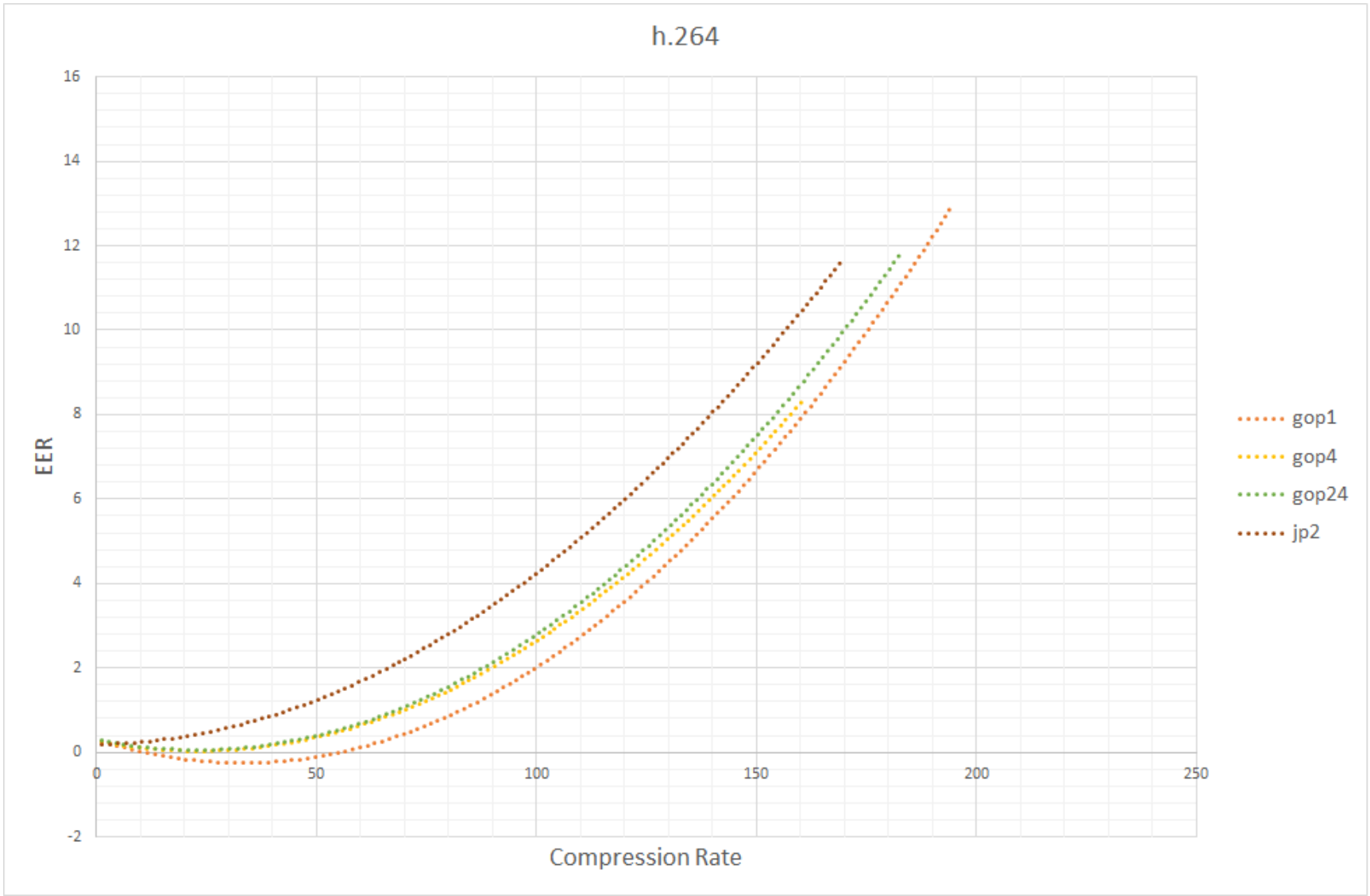


Comparing codecs

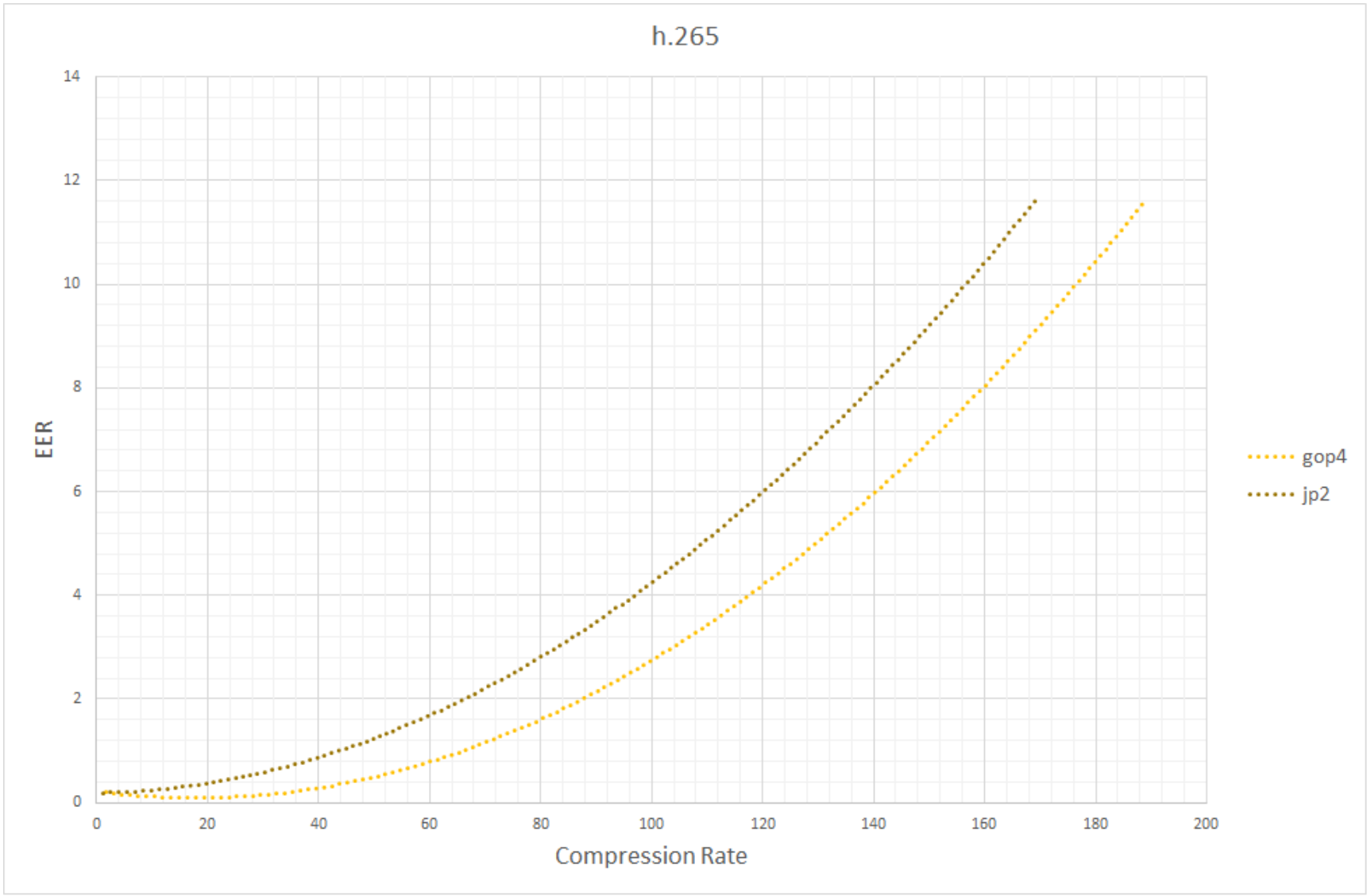
Comparing codecs



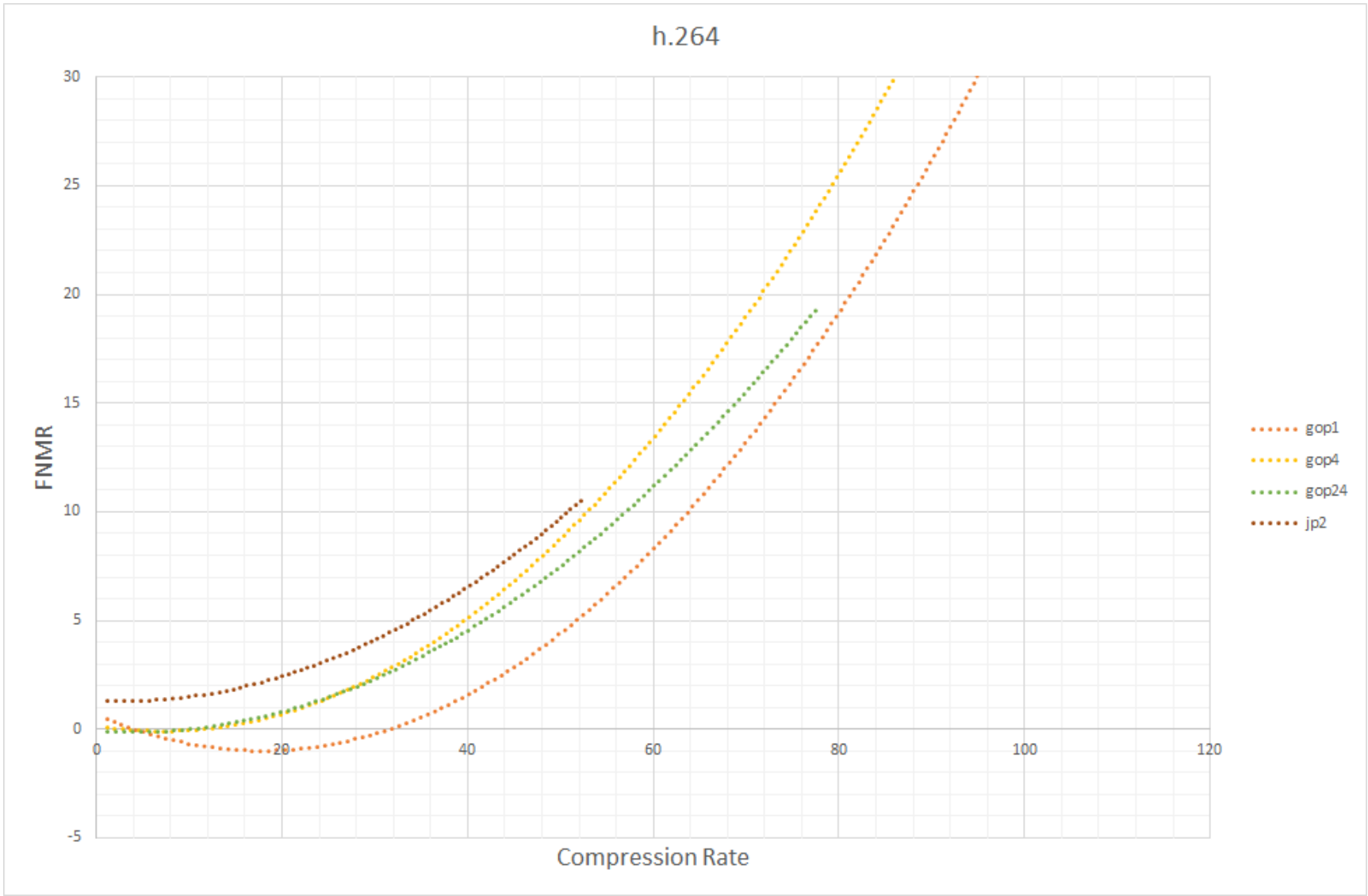
Comparing group of pictures



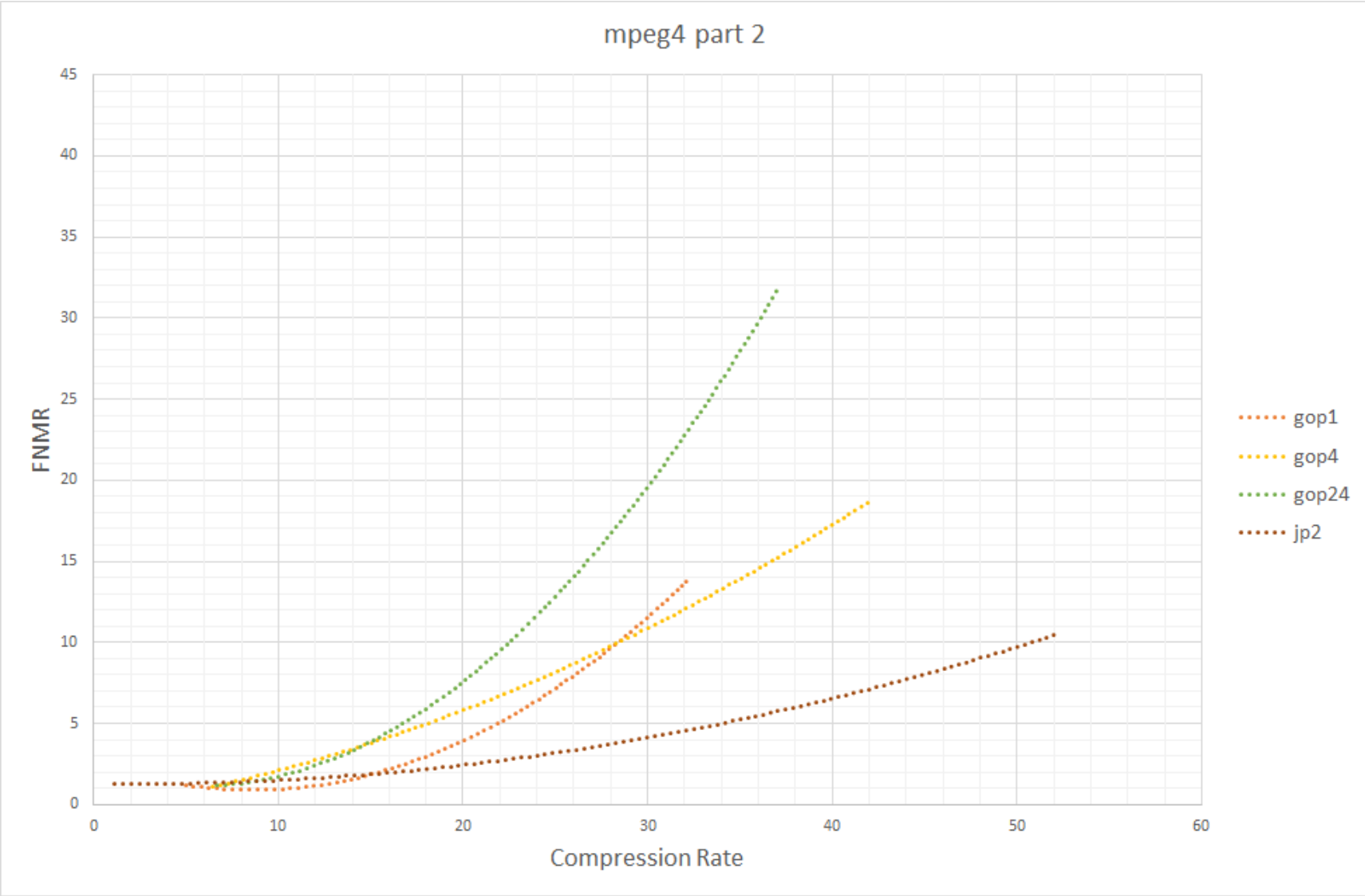
Comparing group of pictures



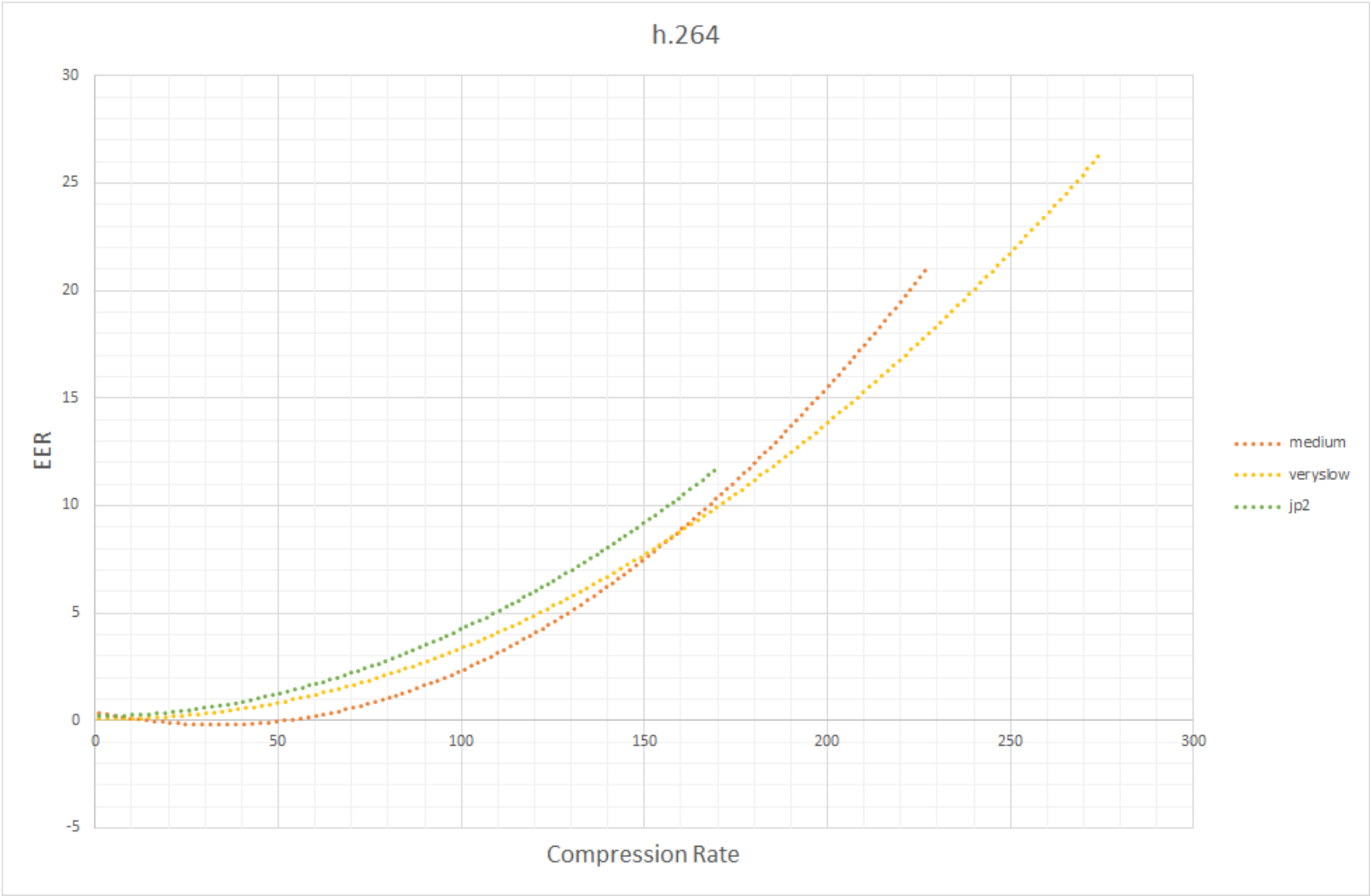
Comparing group of pictures



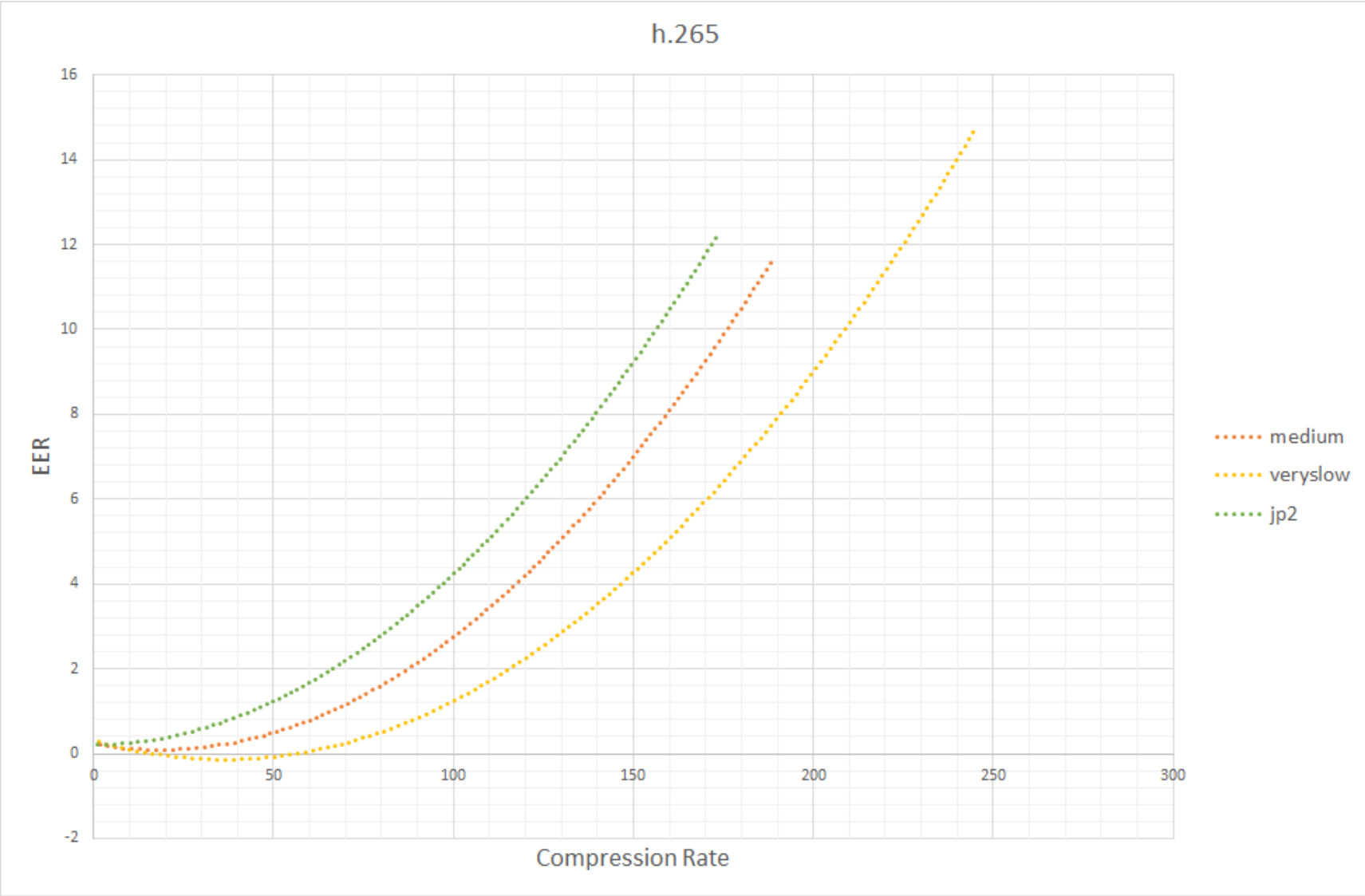
Comparing group of pictures



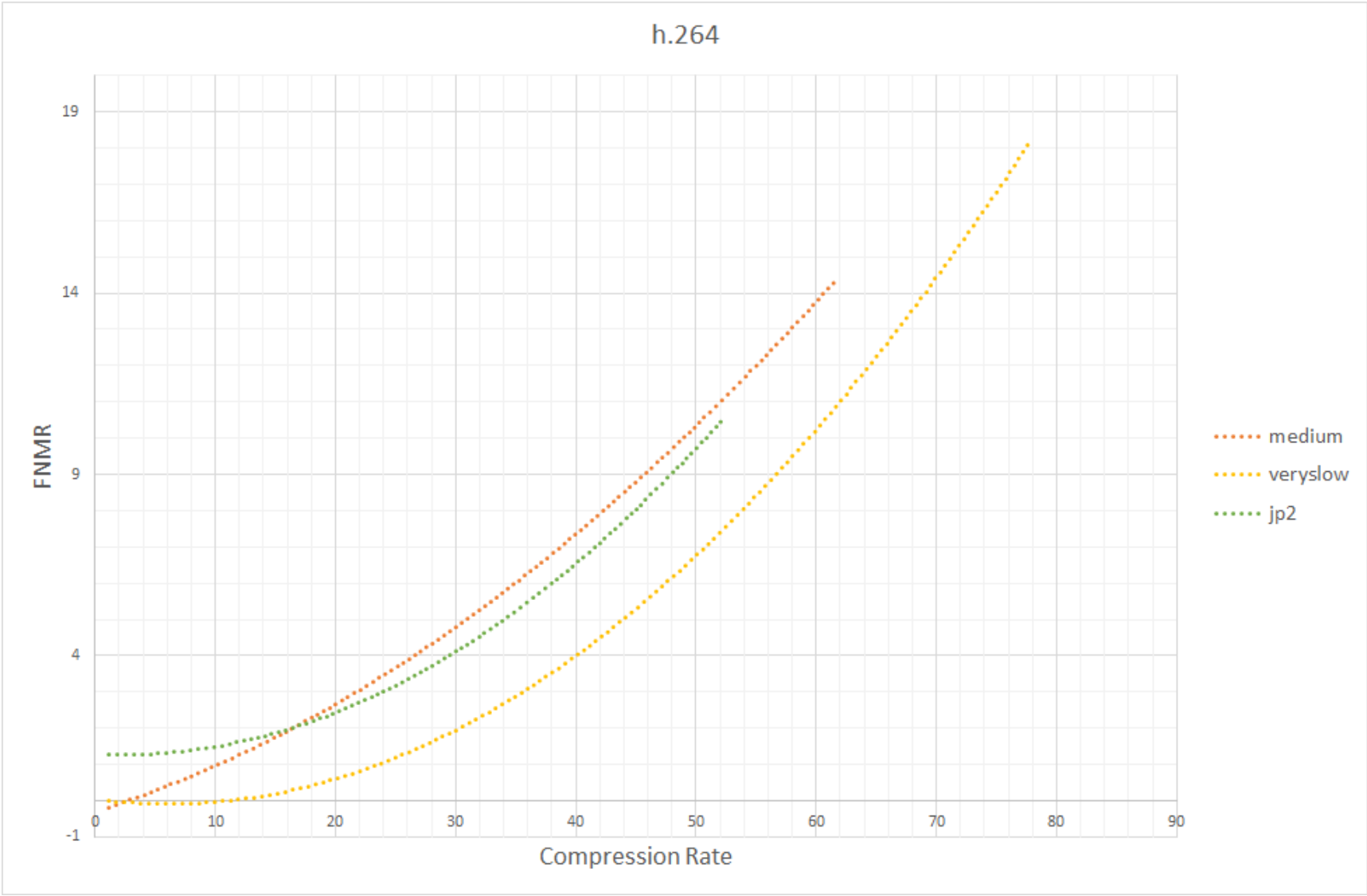
Comparing presets



Comparing presets



Comparing presets



Conclusion

Best Results:

- GOP 1 or GOP 4
- h.264 or h.265
- veryslow

40% higher compression at same EER (h.265 veryslow)
30% higher compression at same FNMR (h.264 veryslow)

Thank you for your attention!